High Performance Computing – Programming Paradigms and Scalability

PD Dr. rer. nat. habil. Ralf-Peter Mundani
Computation in Engineering / BGU
Scientific Computing in Computer Science / INF

Summer Term 2018

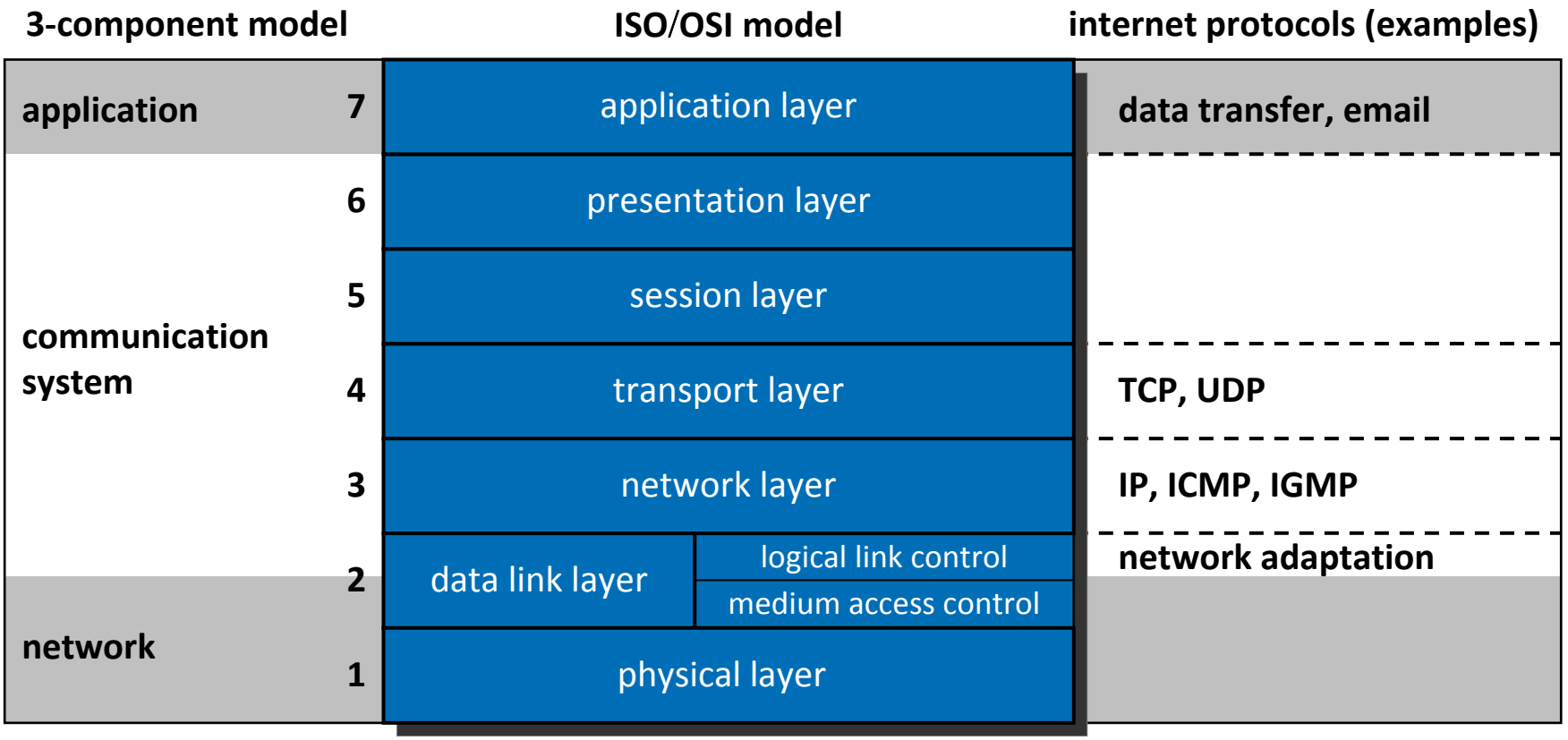
Part 2: High-Performance Networks

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

- overview
  - definitions
  - static topologies
  - dynamic topologies
  - examples

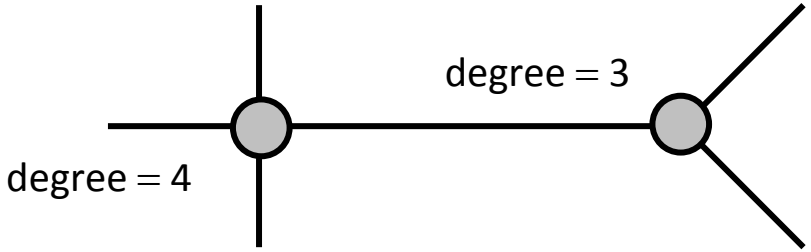
Definitions

- reminder: protocols



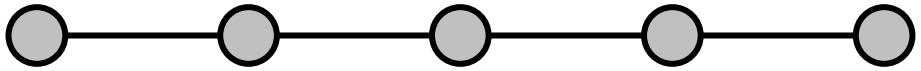
Definitions

- 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



Definitions

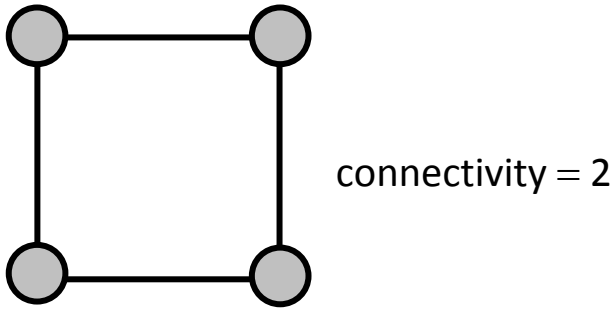
- diameter
  - distance of a pair of nodes (length of the shortest path between a pair of nodes), i.e. the number of nodes a message has to pass on its way from the sender to the receiver
  - diameter of a network := max. distance of all pair of nodes in the network
  - higher diameters (between two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



diameter = 4

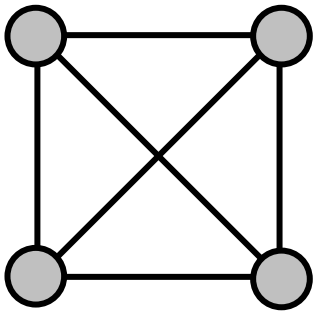
Definitions

- connectivity
  - min. number 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



Definitions

- bisection width
  - min. number of edges (cables) that have to be removed to separate the network into two equal parts (bisection width ≠ connectivity, see below)
  - important for determining the number 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: number of nodes/2)



bisection width = 4

(connectivity = 3)

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

Definitions

- bandwidth
  - max. transmission performance of a network for a certain amount of time
  - bandwidth 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

Definitions

- 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

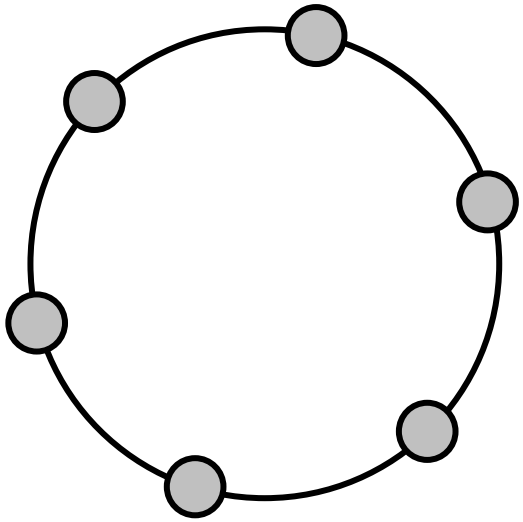
- overview
  - definitions ✓
  - static topologies
  - dynamic topologies
  - examples

Static 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

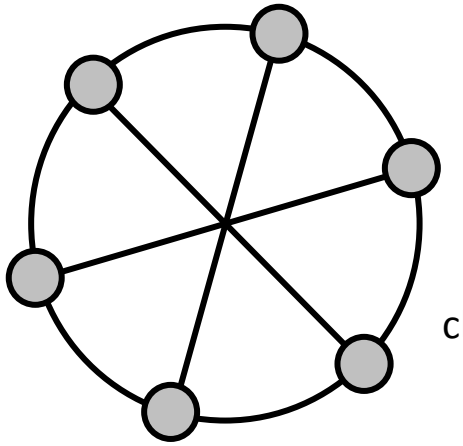
Static Topologies

- 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



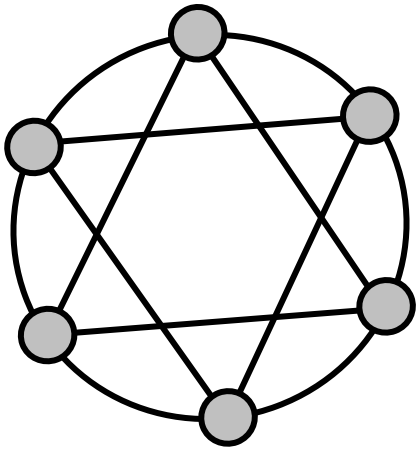
Static Topologies

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



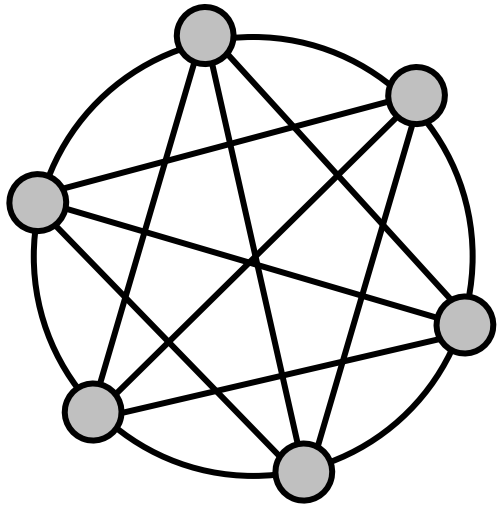
chordal ring of degree = 3

chordal ring of degree = 4



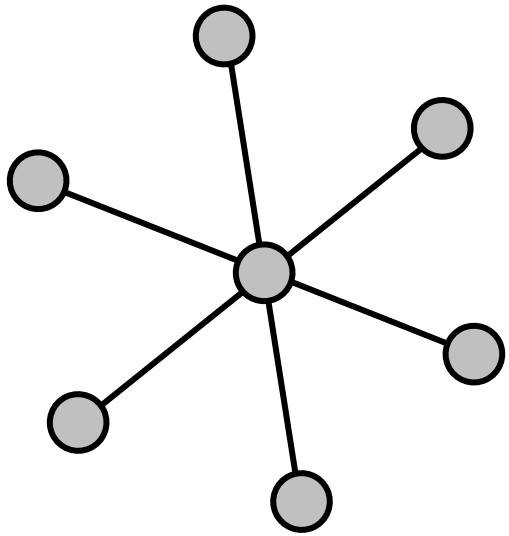
Static Topologies

- 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



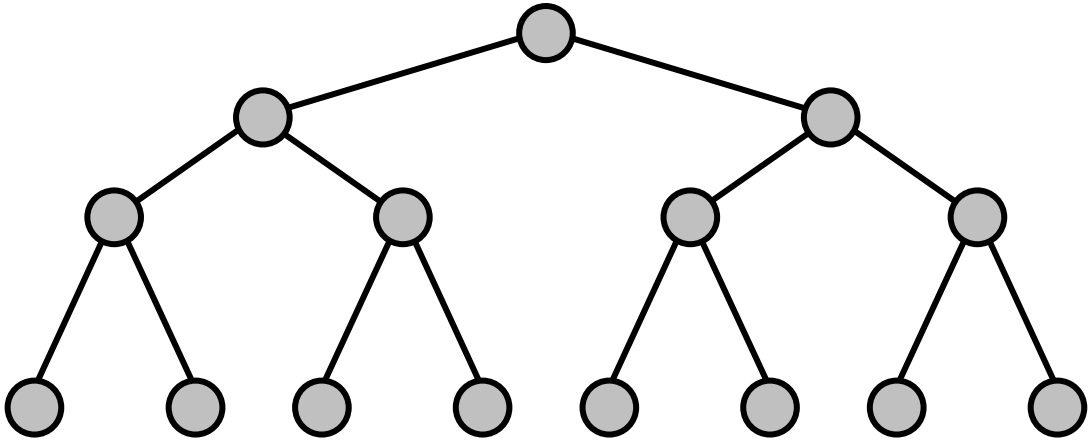
Static Topologies

- 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



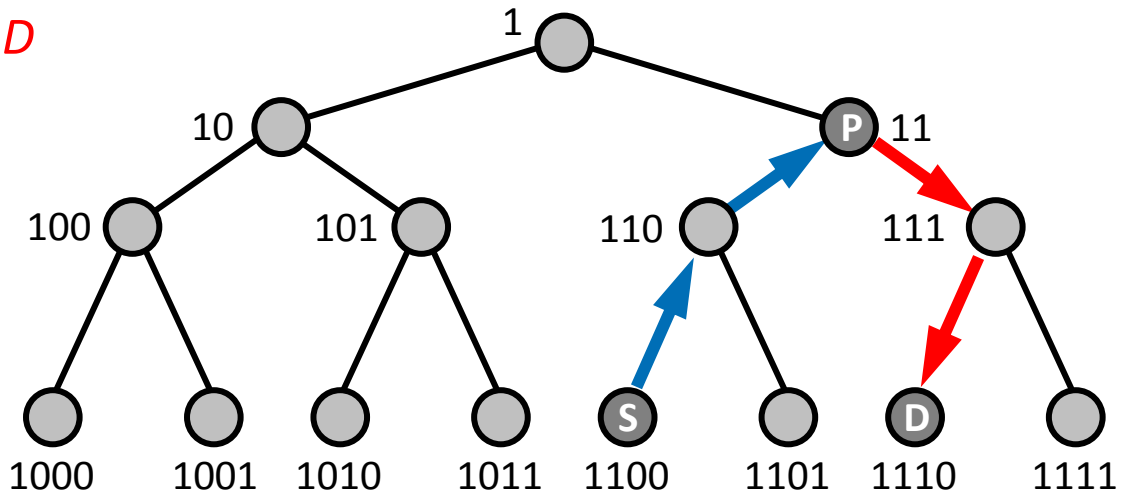
Static Topologies

- binary tree
  - two-dimensional network
  - N nodes and N–1 edges (tree height $h = \lfloor \mathrm{ld}\, N \rfloor$)
  - degree $= 3$
  - diameter $= 2h$
  - bisection width $= 1$
  - drawback: bottleneck in direction of root (➔ blocking)



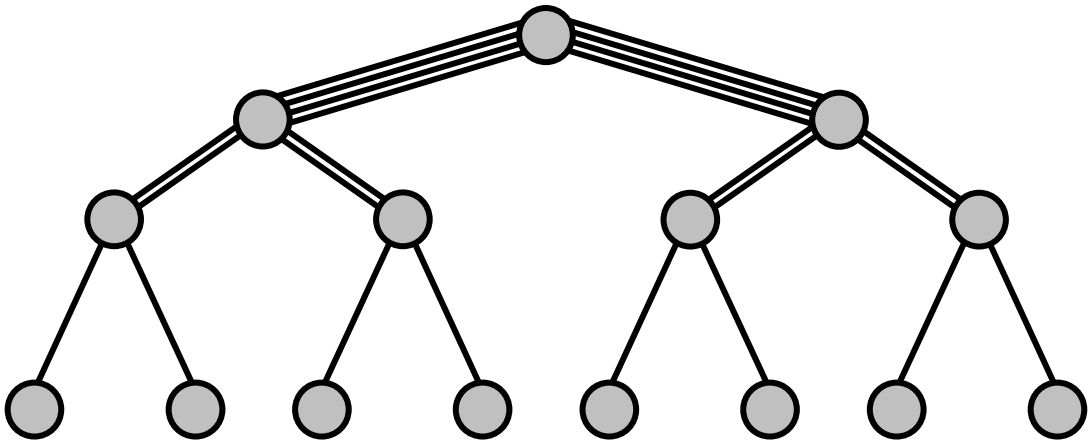
Static Topologies

- 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 $S \rightarrow P$
    - descend from $P \rightarrow D$



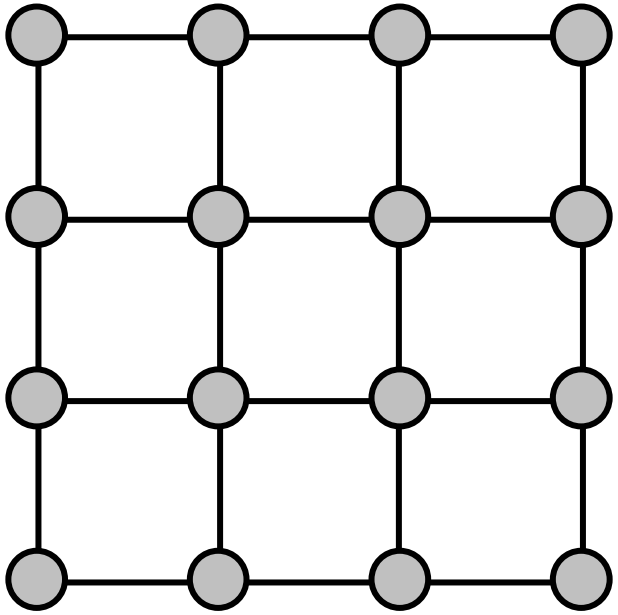
Static Topologies

- 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 Topologies

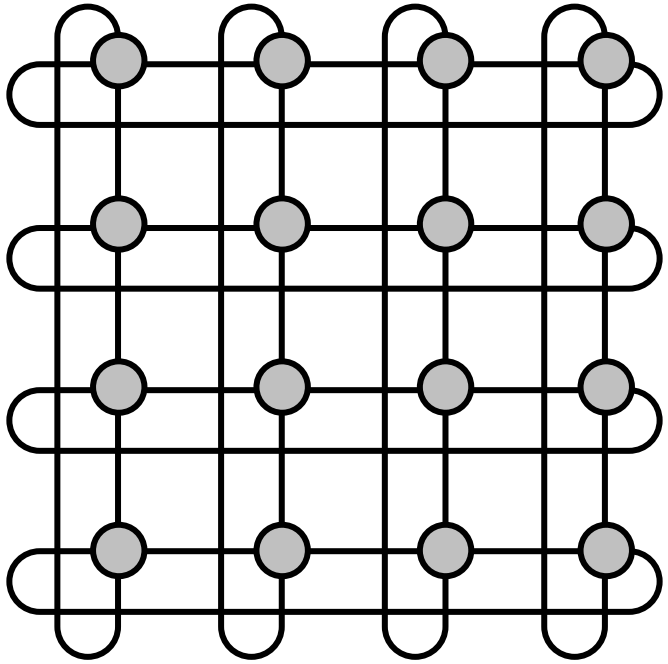
- mesh / torus
  - k-dimensional network
  - N nodes and $k \cdot (N - r^{k-1})$ edges ($r \times r$ mesh, $r = \sqrt[k]{N}$)
  - degree $= 2k$
  - diameter $= k \cdot (r - 1)$
  - bisection width $= r^{k-1}$
  - high fault tolerance
  - drawback
    - large diameter
    - too expensive for $k > 3$



2D mesh

Static Topologies

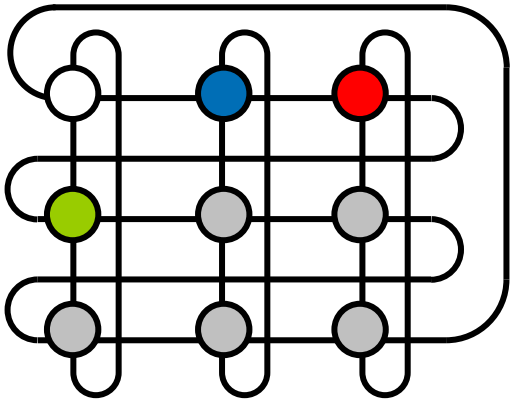
- mesh / torus (cont'd)
  - k-dimensional network
  - N nodes and $k \cdot N$ edges ($r \times r$ mesh, $r = \sqrt[k]{N}$)
  - diameter $= k \cdot \lfloor r/2 \rfloor$
  - bisection width $= 2r^{k-1}$
  - frequently used: BlueGene/L, e.g.
  - drawback: too expensive for $k > 3$

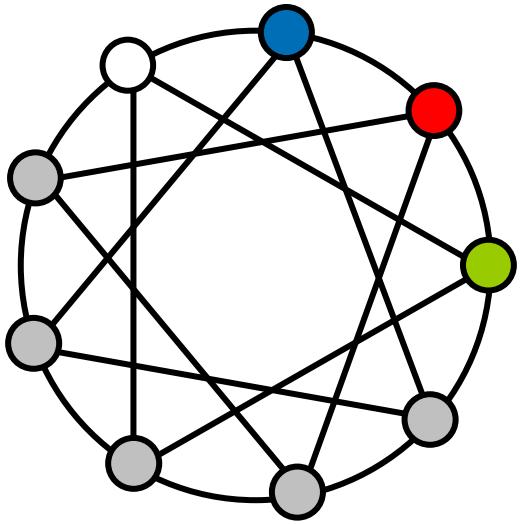


2D torus

Static Topologies

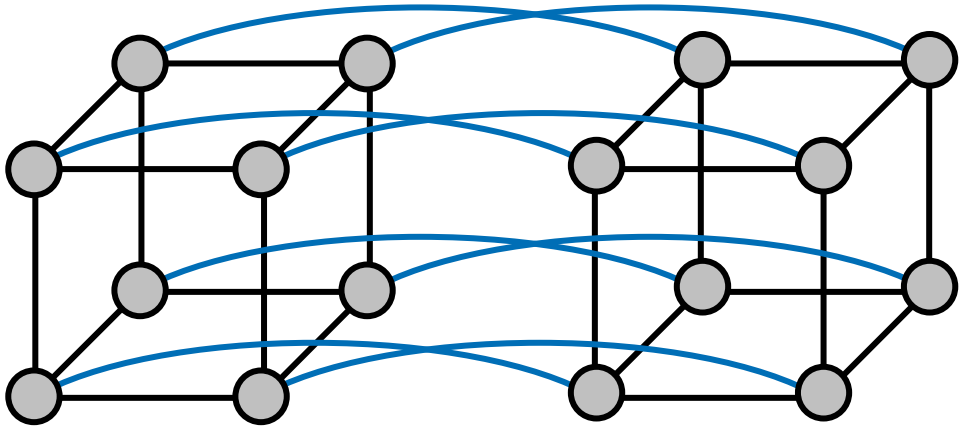
- ILLIAC mesh
  - two-dimensional network
  - N nodes and $2N$ edges ($r \times r$ mesh, $r = \sqrt{N}$)
  - degree $= 4$
  - diameter $= r - 1$
  - bisection width $= 2r$
  - conforms to a chordal ring of degree $= 4$





Static Topologies

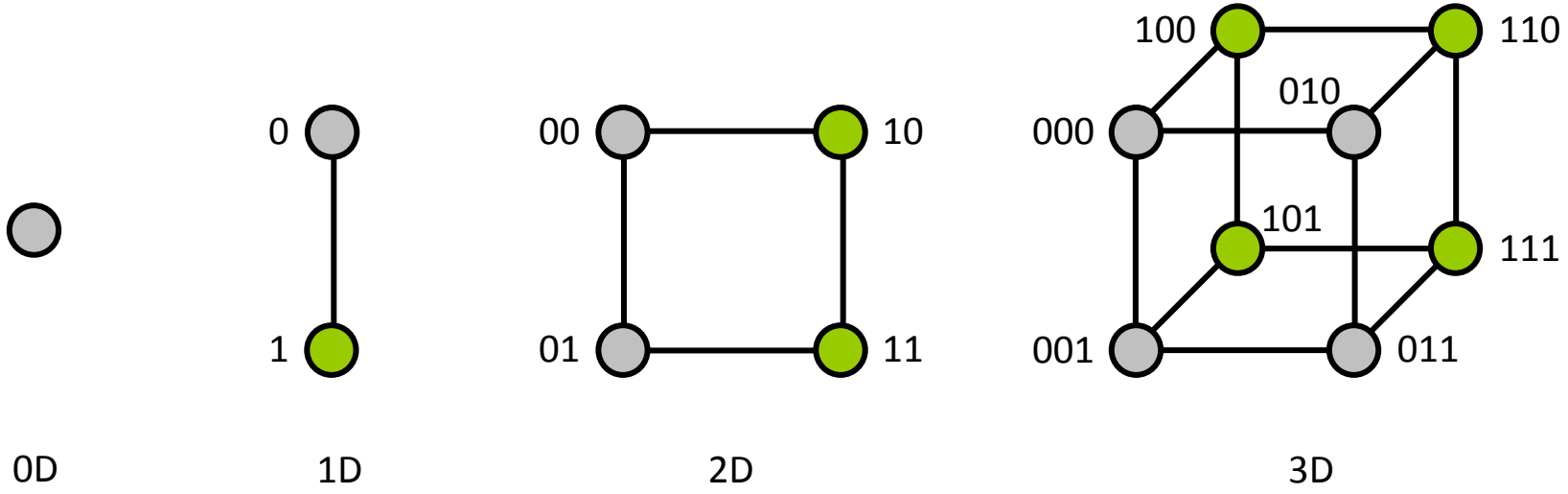
- hypercube
  - k-dimensional network
  - 2^k nodes and $k \cdot 2^{k-1}$ edges
  - degree $= k$
  - diameter $= k$
  - bisection width $= 2^{k-1}$
  - drawback: scalability (only doubling of nodes allowed)



4D hypercube

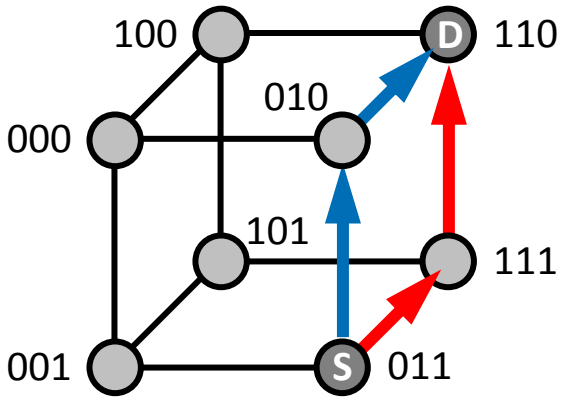
Static Topologies

- 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 Topologies

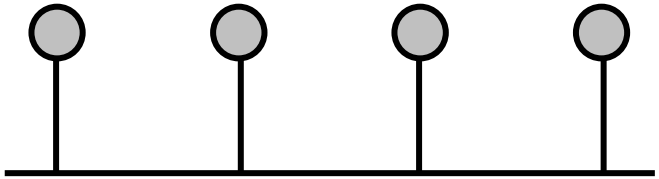
- hypercube (cont'd)
  - nodes are directly connected for a HAMMING distance of 1 only
  - routing
    - compute $S \otimes D$ (XOR) for possible ways between nodes S and D
    - route in increasing / decreasing order until final destination is reached
  - example
    - $S =$ '011', $D =$ '110'
    - $S \otimes D =$ '101'
    - decreasing: '011' → '010' → '110'
    - increasing: '011' → '111' → '110'



- overview
  - definitions ✓
  - static topologies ✓
  - dynamic topologies
  - examples

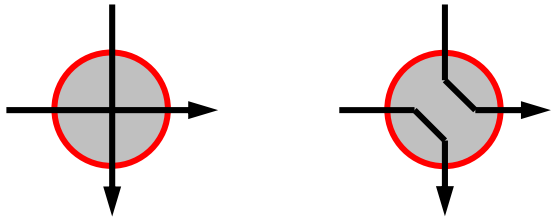
Dynamic Topologies

- 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
  - example: CSMACD

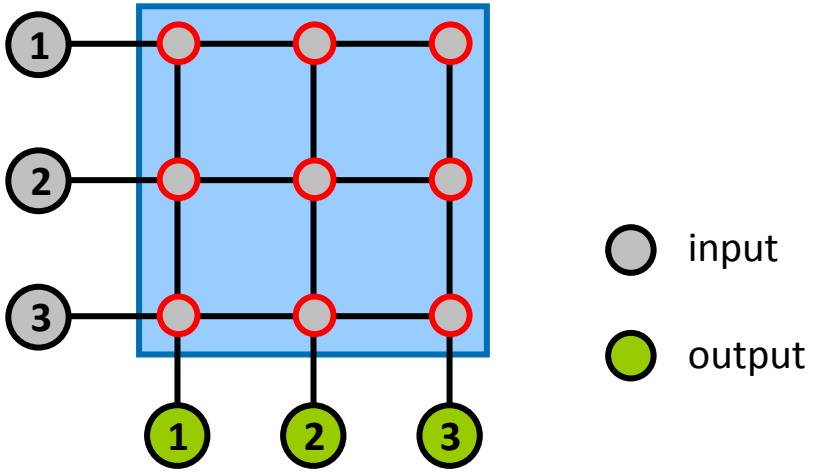


Dynamic Topologies

- 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

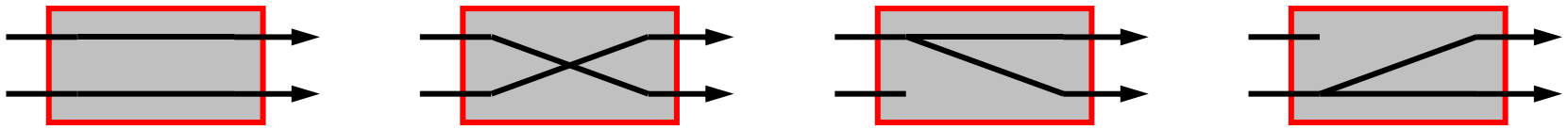


switch element



Dynamic Topologies

- permutation networks
  - tradeoff between low performance of buses and high costs of crossbars
  - based on 2×2 switch elements with four switching possibilities
    - straight
    - crossed
    - upper / lower broadcast

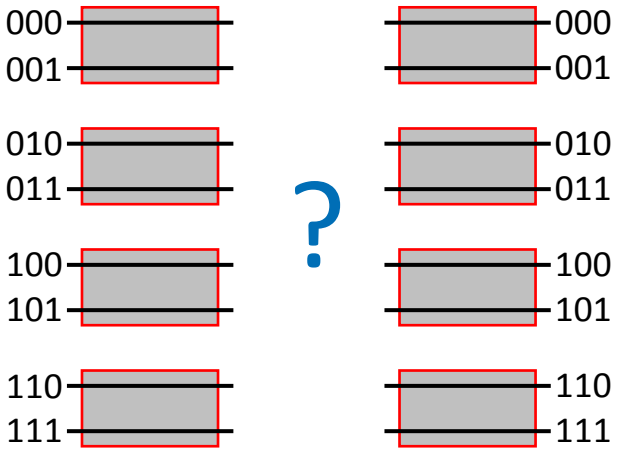


straight | crossed | upper broadcast | lower broadcast

  - switching *N* inputs to *N* outputs ➔ permutation of inputs (to outputs)
    - single stage: one column with *N*/2 of 2×2 switch elements
    - multistage: several of those columns

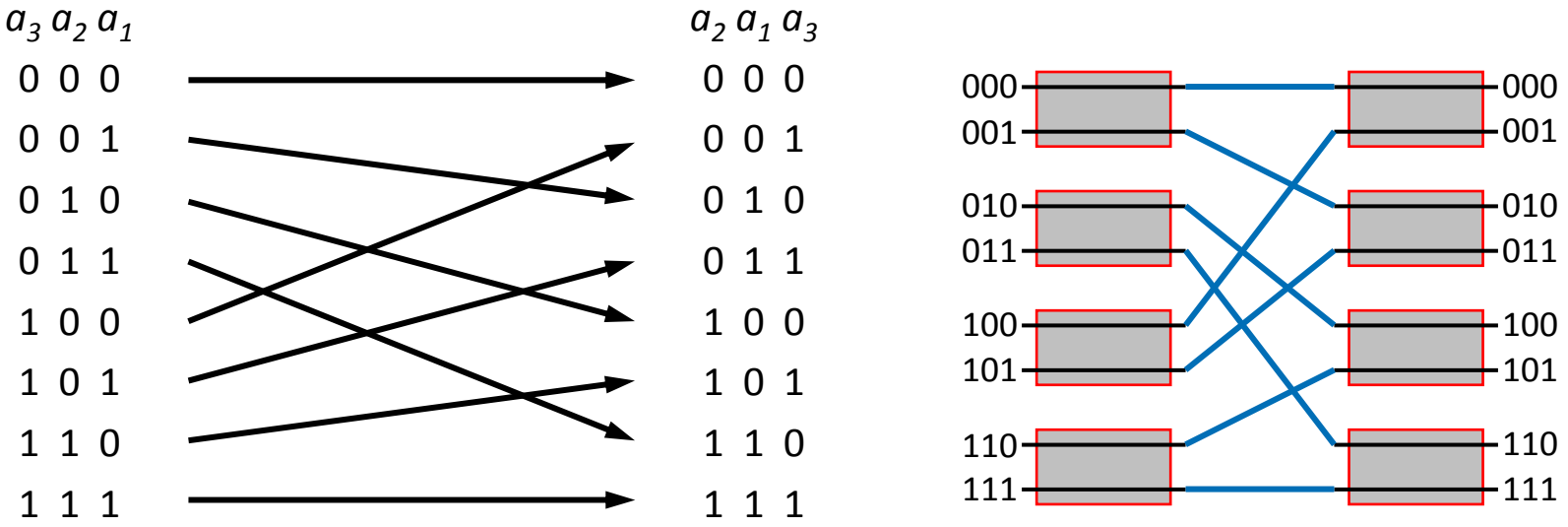
Dynamic Topologies

- 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}, …, a_2, a_1)
  - permutations can now be expressed as simple bit manipulation
  - typical permutations
    - perfect shuffle
    - butterfly
    - exchange



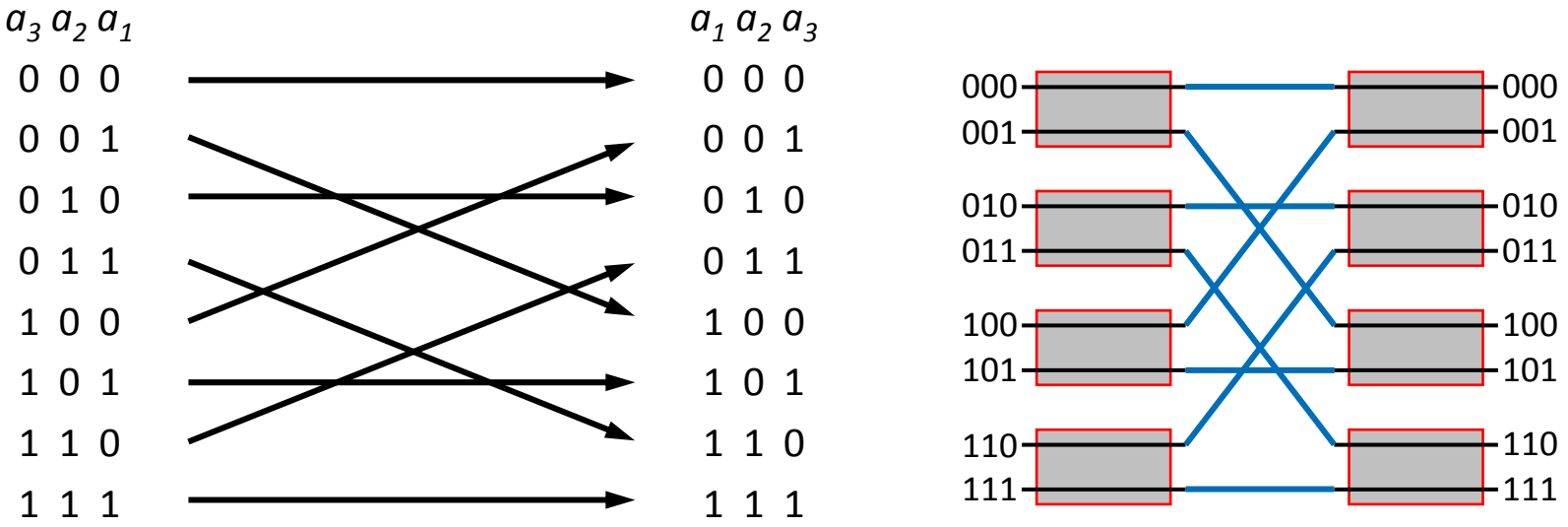
Dynamic Topologies

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



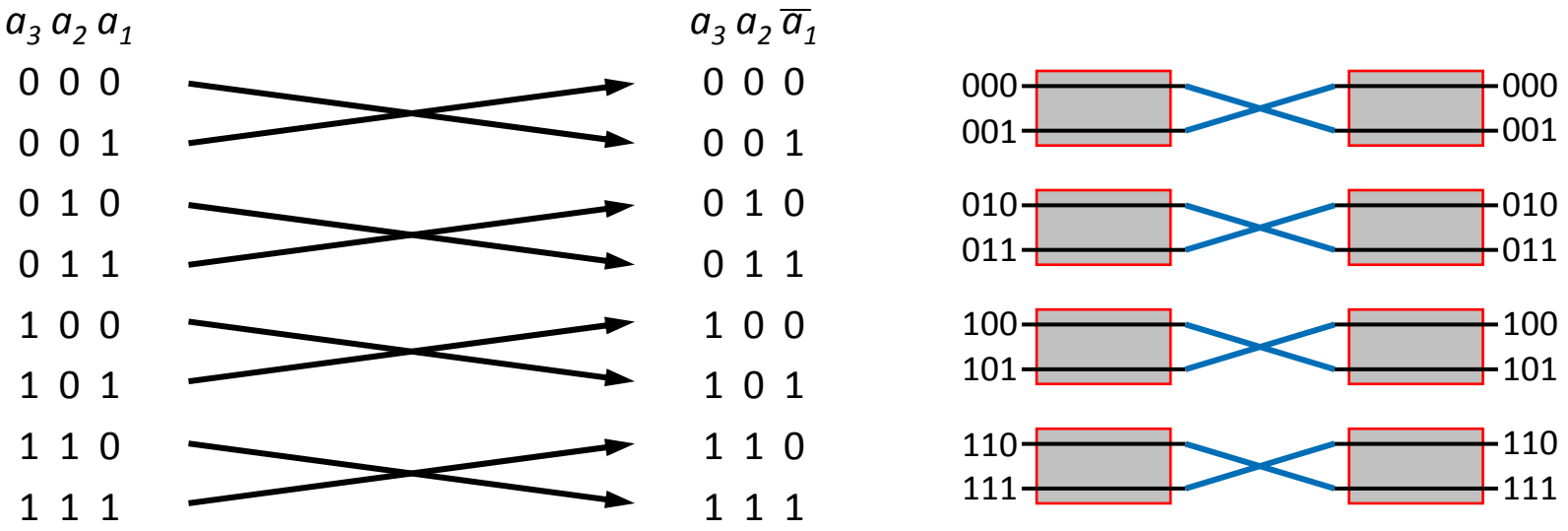
Dynamic Topologies

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



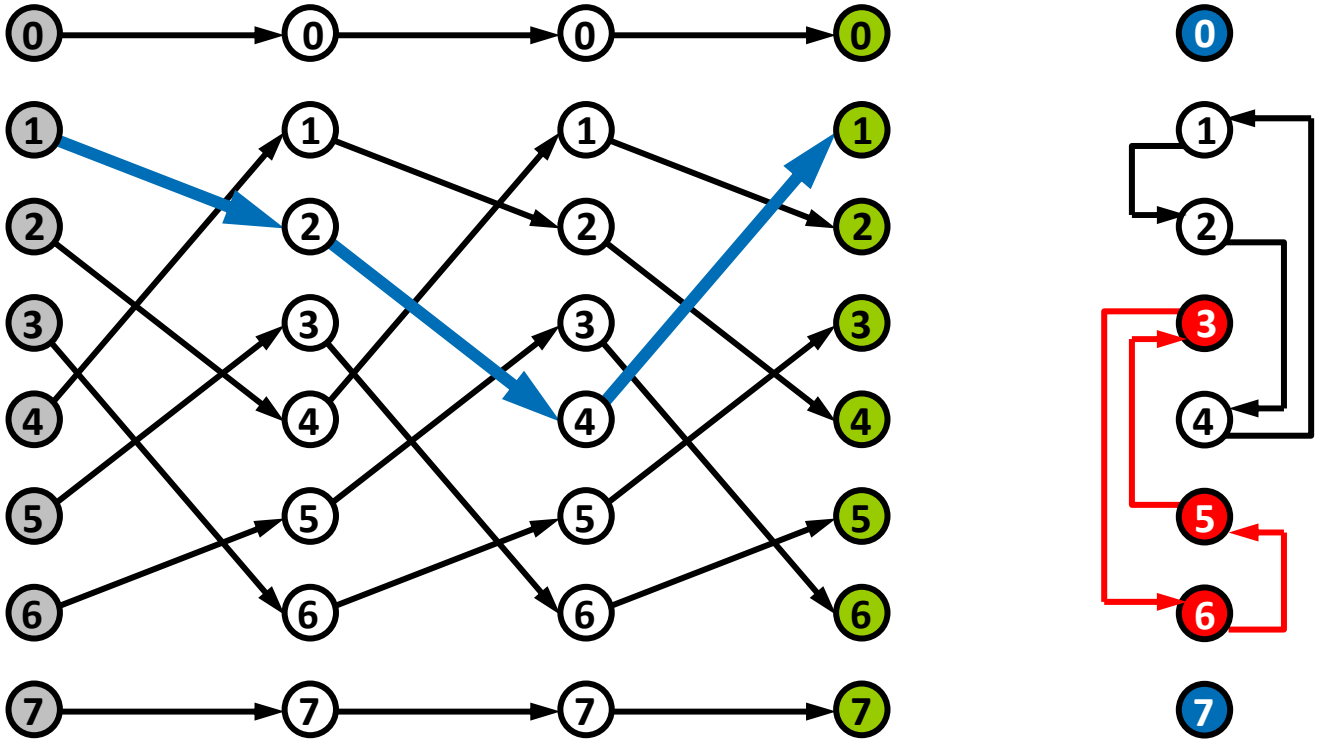
Dynamic Topologies

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



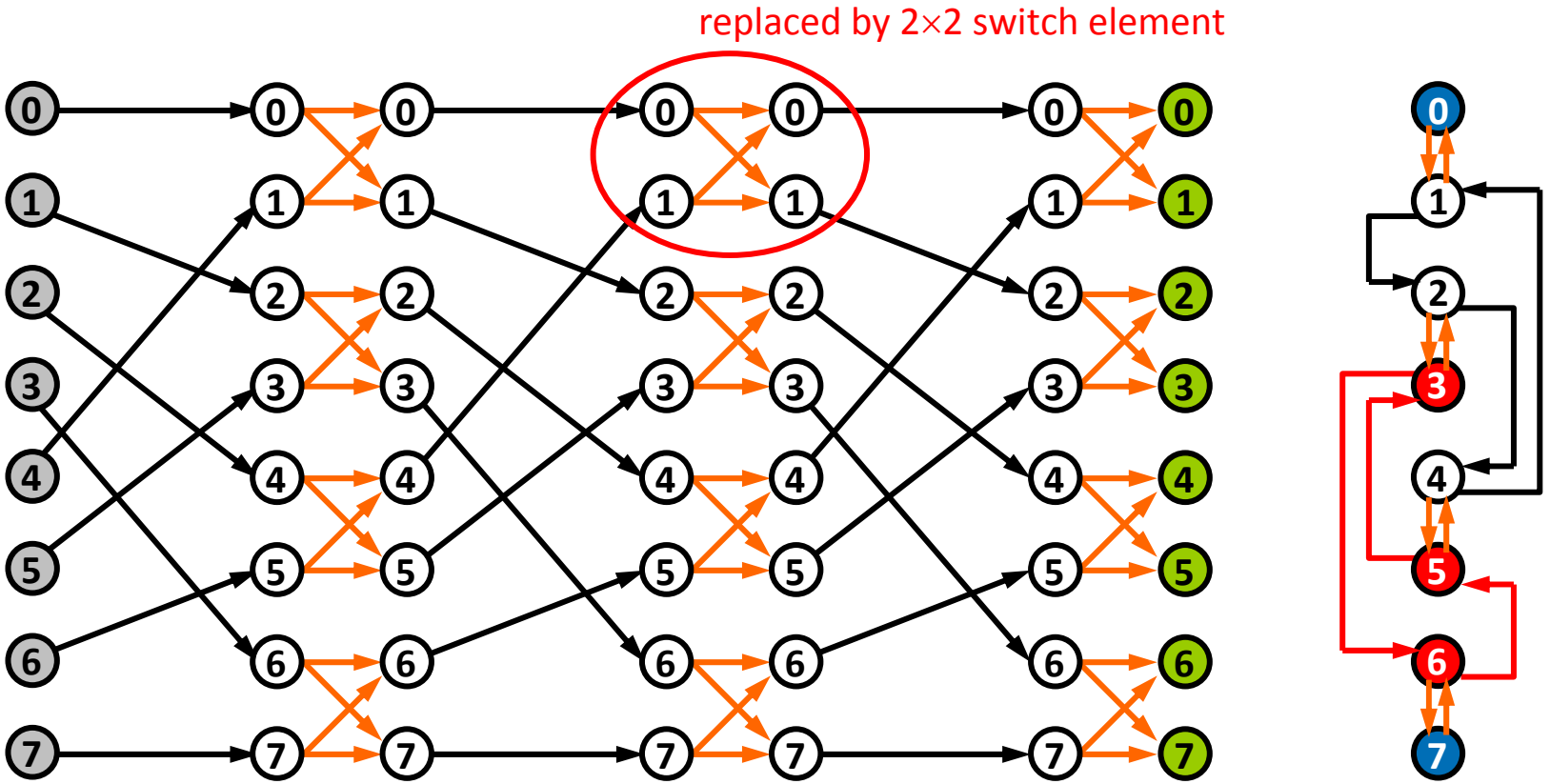
Dynamic Topologies

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



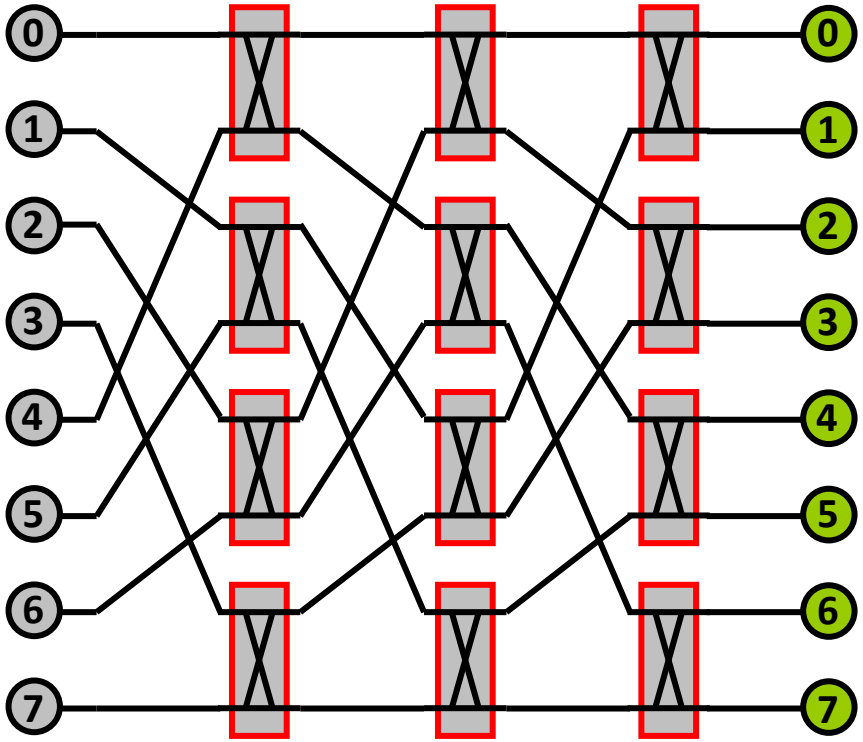
Dynamic Topologies

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



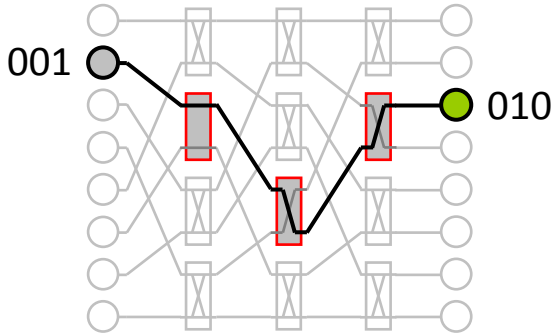
Dynamic Topologies

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



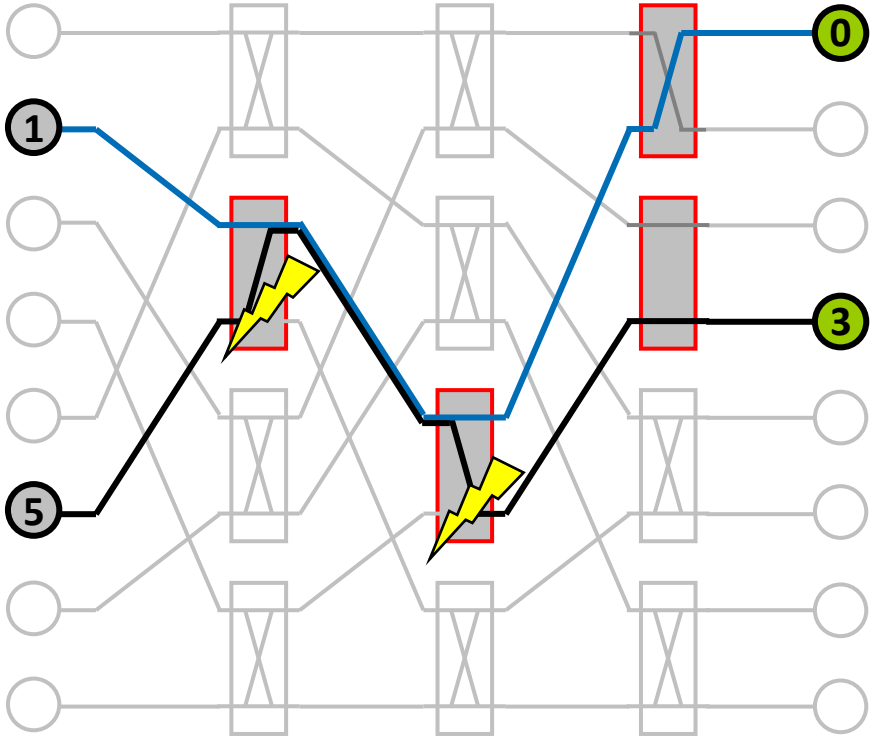
Dynamic Topologies

- omega (cont'd)
  - multistage network (for N nodes ➔ ld N stages)
  - N nodes and $E = N/2 \cdot (\text{ld } N)$ switch elements
  - $N!$ permutations possible, but only 2^E ($< N!$) different switch states
  - (self configuring) routing
    - compare addresses from S and D bitwise from left to right ➔ stage i evaluates address bits s_i and d_i
    - if equal switch straight (–), otherwise switch crossed (×)
  - example
    - S = '001', D = '010'
    - switch states: – × ×



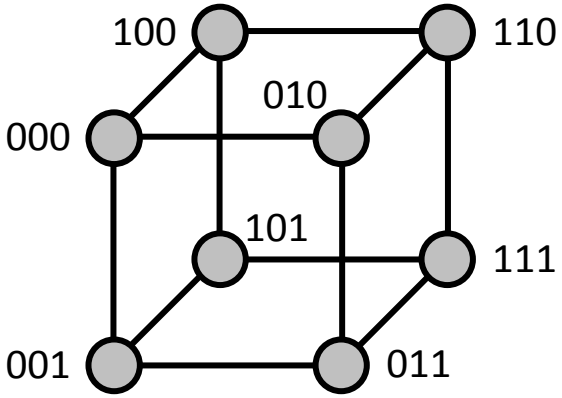
Dynamic Topologies

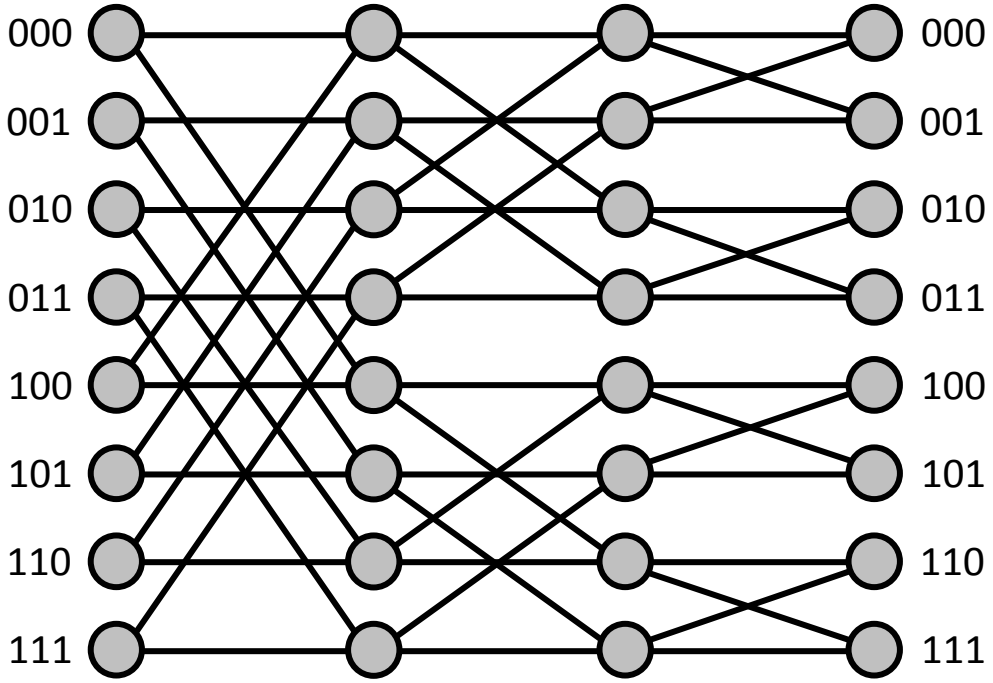
- omega (cont'd)
  - problem: there exists exactly one route from each input to each output
    ➔ risk of blocking
  - example: simultaneous connections 1 → 0 and 5 → 3
    - 1 → 0: *S* = '001', *D* = '000'
      ➔ switch states: – – ×
    - 5 → 3: *S* = '101', *D* = '011'
      ➔ switch states: × × –
    - conflicting switch states



Dynamic Topologies

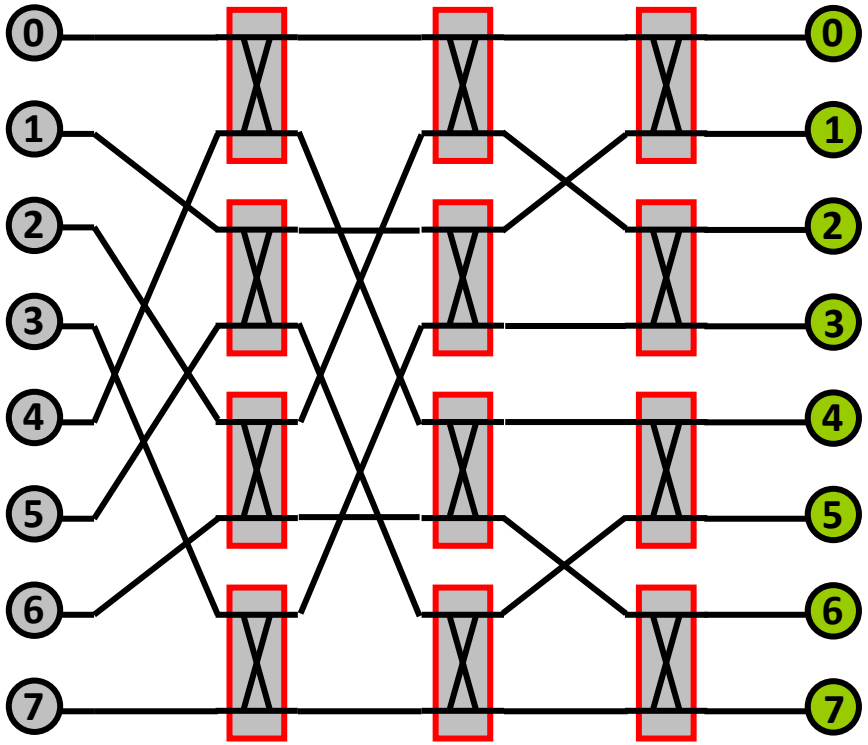
- banyan / butterfly
  - idea: unrolling of a static hypercube
  - bitwise processing of address bits a_i from left to right ➔ dynamic hypercube a.k.a. butterfly (known from FFT flow diagram)





Dynamic Topologies

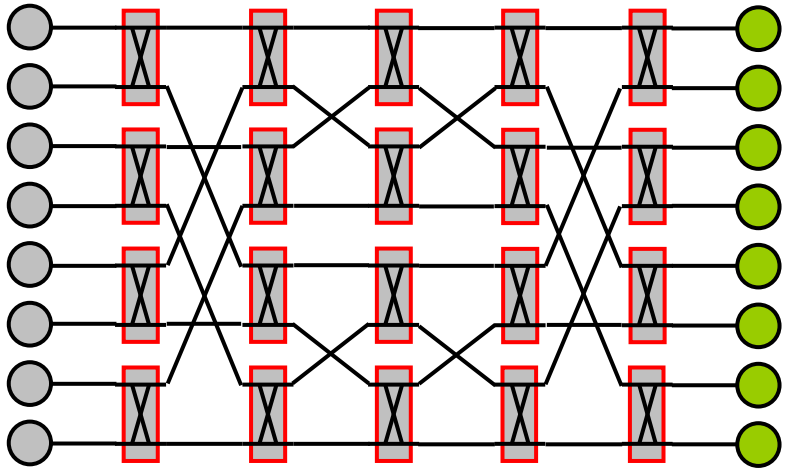
- banyan / butterfly (cont'd)
  - replace crossed connections by 2×2 switch elements
  - introduced by GOKE and LIPOVSKI in 1973; blocking still possible



banyan tree

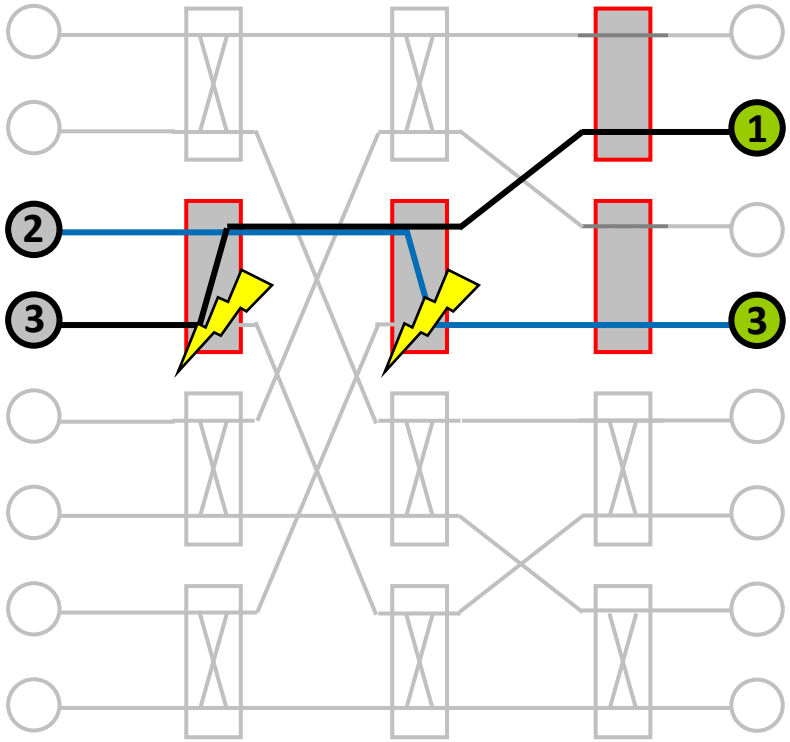
Dynamic Topologies

- BENEŠ
  - multistage network
  - built via merging butterfly network with its copied mirror
  - N nodes and $N \cdot (\text{ld } N) - N/2$ switch elements
  - $N!$ permutations possible, all can be switched
  - *key property*: for any permutation of inputs to outputs there is a contention-free routing



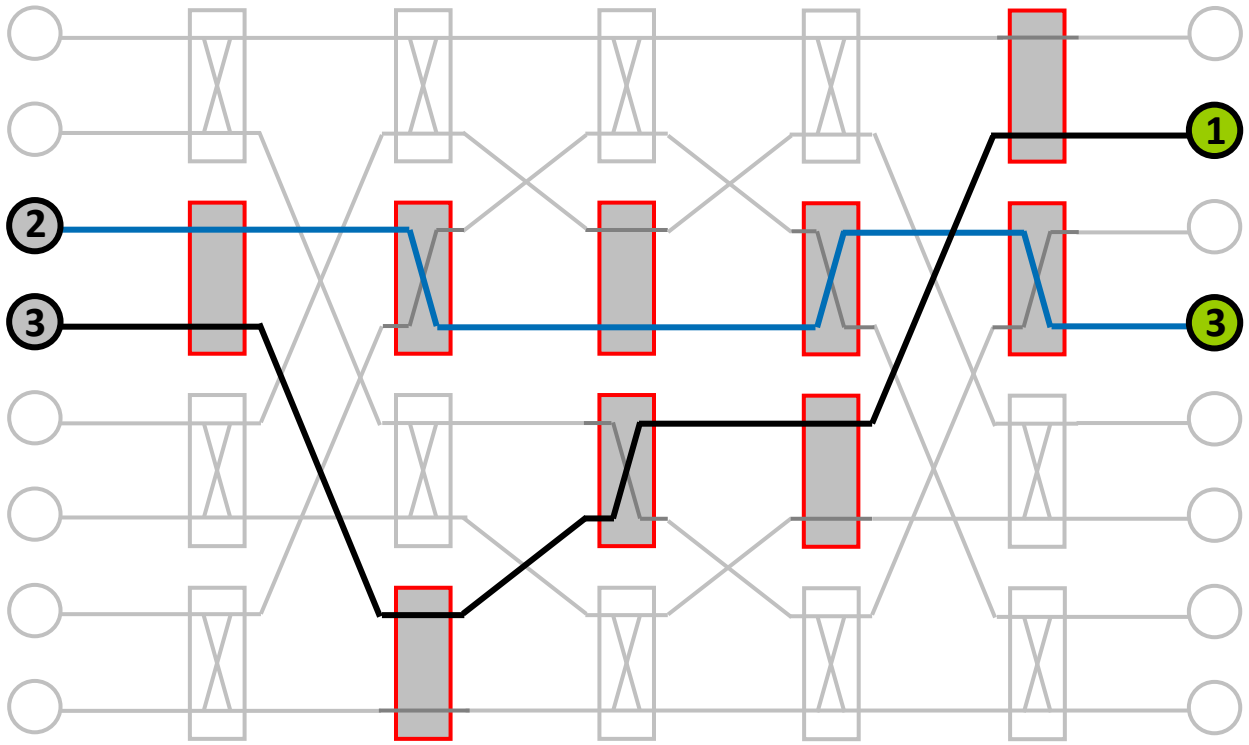
Dynamic Topologies

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



Dynamic Topologies

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



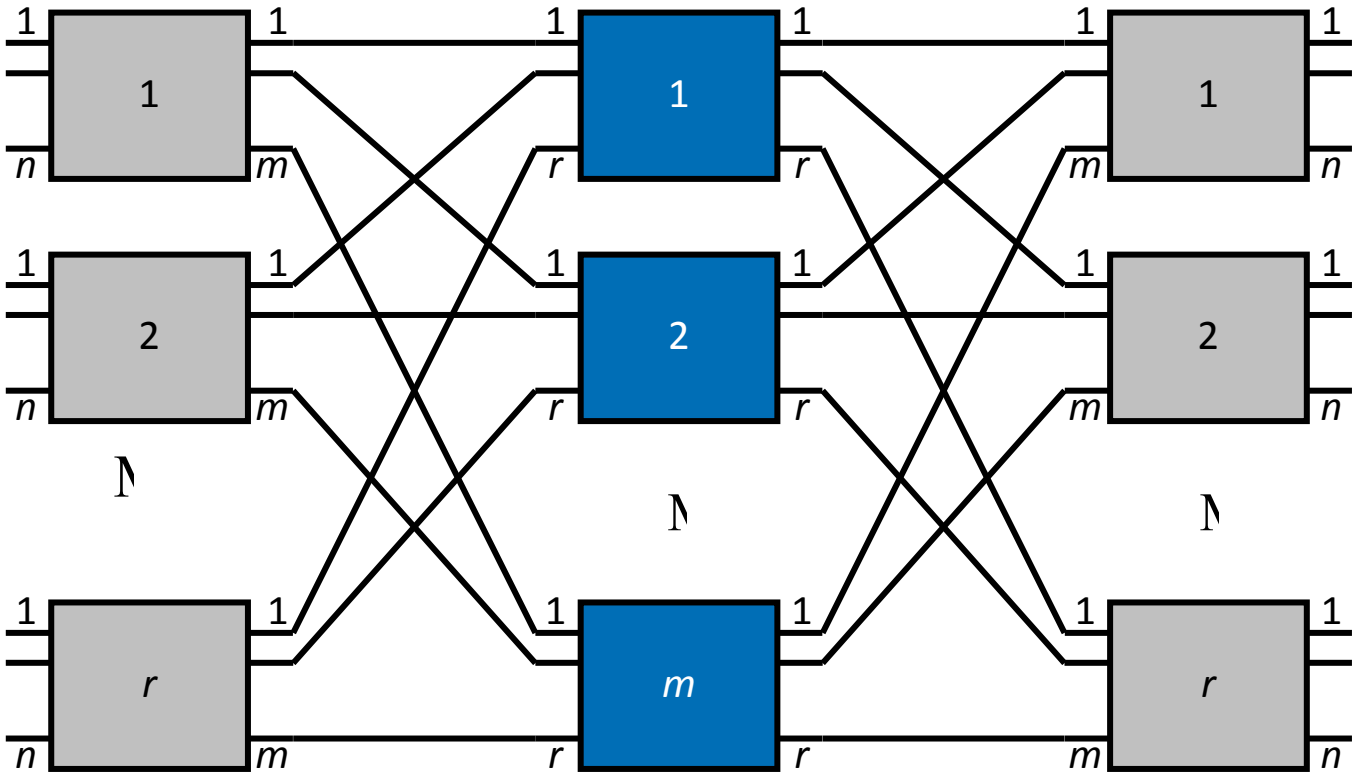
one possibility of routing

Dynamic Topologies

- CLOS
  - proposed by CLOS in 1953 for telephone switching systems
  - objective: to 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 Topologies

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

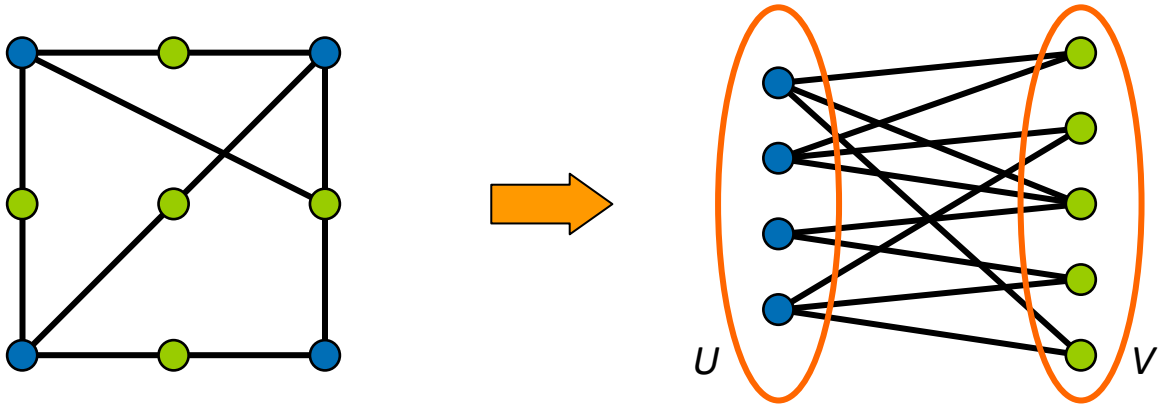


Dynamic Topologies

- 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 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

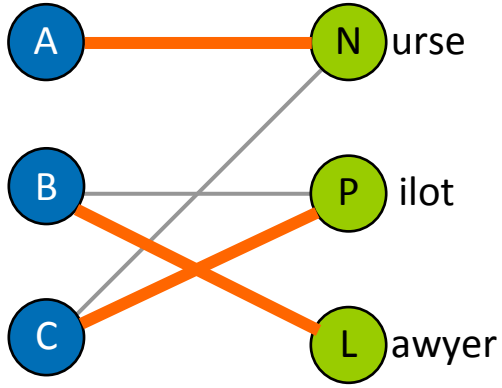


division of vertices in *U* and *V*, i.e. there are no edges within *U* and *V*, only between *U* and *V*

Dynamic 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 Topologies

- CLOS (cont'd)
  - proof for $M \geq N$ via HALL's "Marriage Theorem"
  - 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 S of boys (where $0 \leq S \leq N$) knows S or more girls, each boy can be married with a girl he knows.

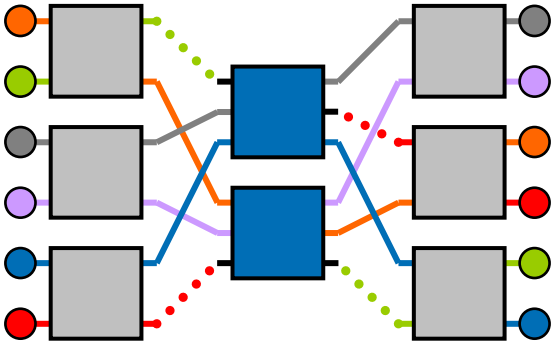
Dynamic Topologies

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

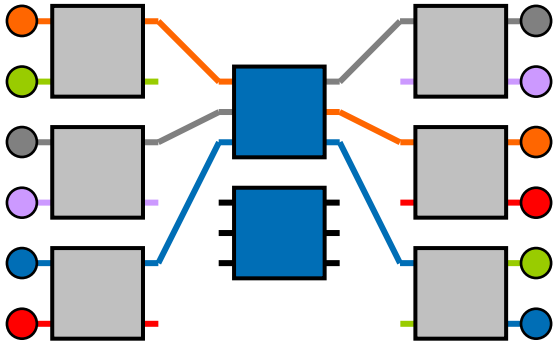
1) for $0 \leq S \leq R$ boys there are $S \cdot N$ connections ➔ at least S girls
2) thus, HALL's theorem states there exists a perfect matching
3) R connections can be handled by one middle stage crossbar
4) bundle these connections and delete the middle stage crossbar
5) repeat from step 1) until $M = 1$
6) new connection can be handled

Dynamic Topologies

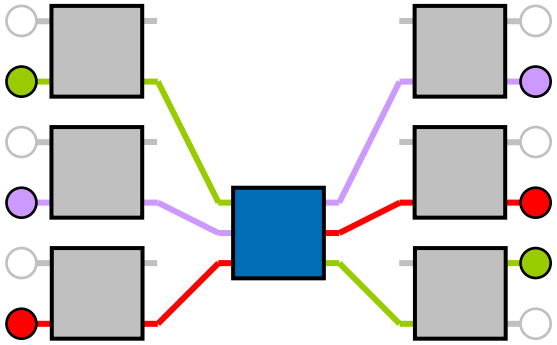
- CLOS (cont'd)
  - proof for $M \geq N$ via HALL's "Marriage Theorem"
  - example: $M = N = 2$



initial situation: two connections cannot be established



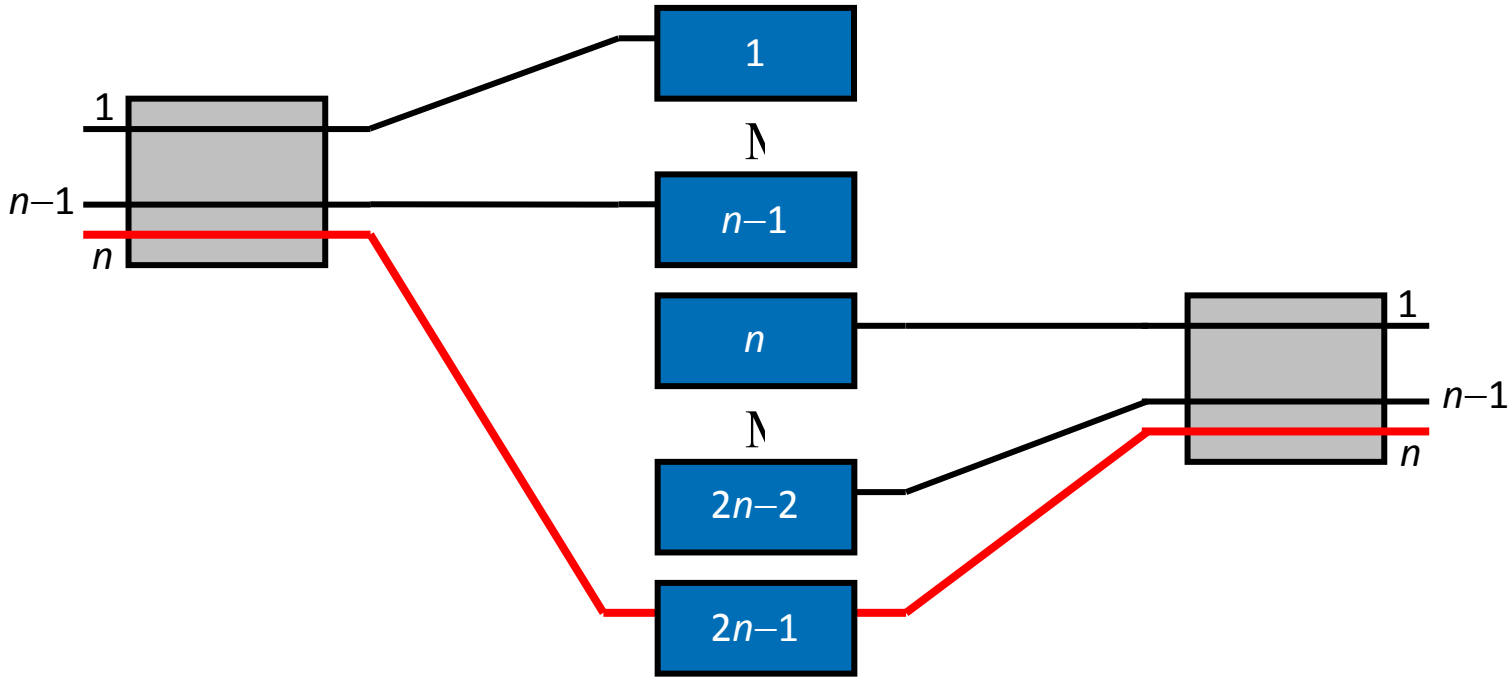
bundle connections to one middle stage crossbar and delete it afterwards ➔ maybe rearrangements are necessary



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

Dynamic 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

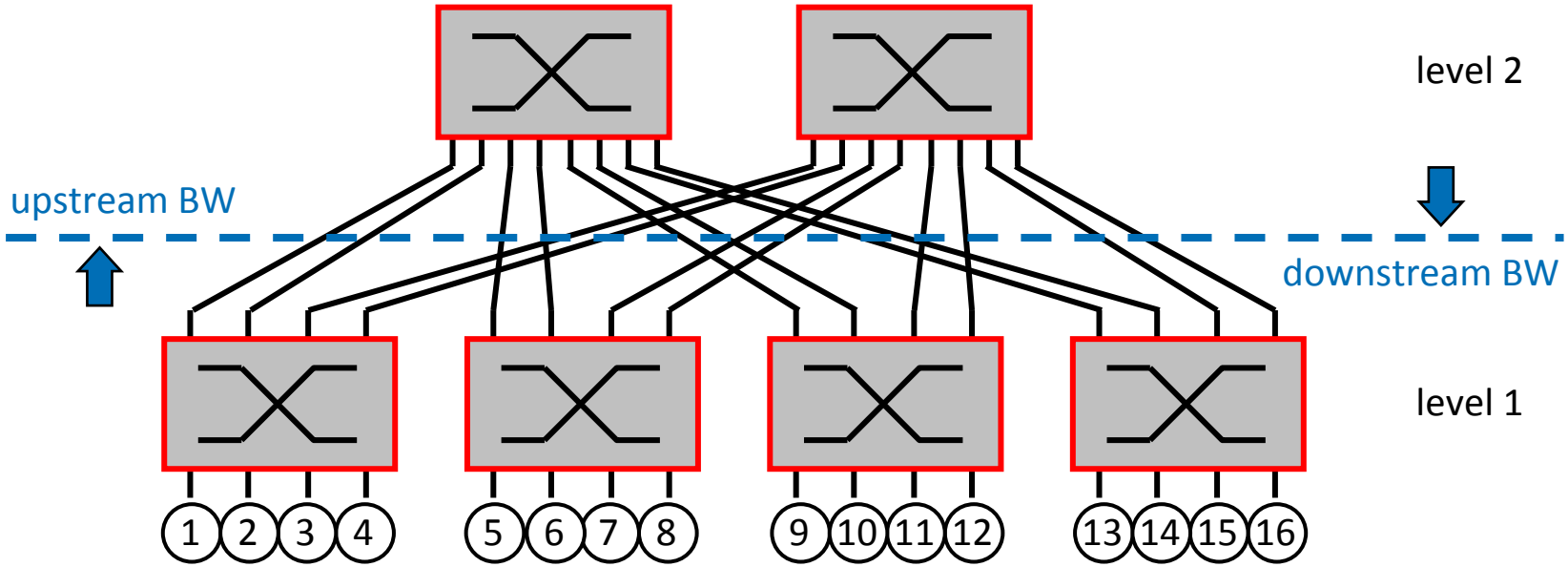


Dynamic Topologies

- constant bisection bandwidth
  - more general concept of CLOS and fat tree networks
  - construction of a non-blocking network connecting *M* nodes
    - using multiple levels of basic *N*×*N* switch elements (*M* > *N*)
    - for any given level, the downstream bandwidth (to nodes) is identical to the upstream bandwidth (from nodes)
  - key for non-blocking: always preserve identical bandwidth (upstream and downstream) between any two levels
  - observation
    - two-stage CBB network connecting *M* nodes always needs 3*M* ports
      ➔ each node needs two ports in first and one port in second stage

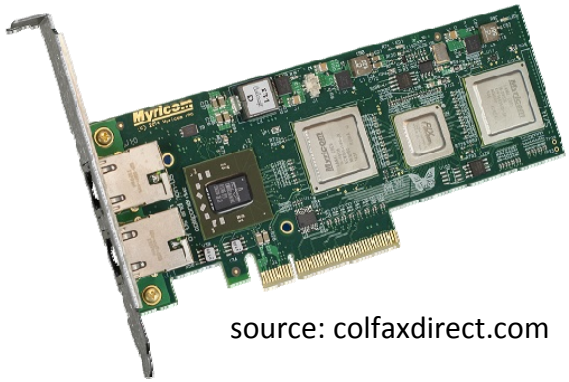
Dynamic Topologies

- constant bisection bandwidth (cont'd)
  - example: two-stage CBB
    - connecting $M = 16$ nodes with 4×4 switch elements
    - hence, in total $3M = 48$ ports (i.e. 6 switch elements) necessary
    - upstream bandwidth = downstream bandwidth



- overview
  - definitions ✓
  - static topologies ✓
  - dynamic topologies ✓
  - examples

Examples

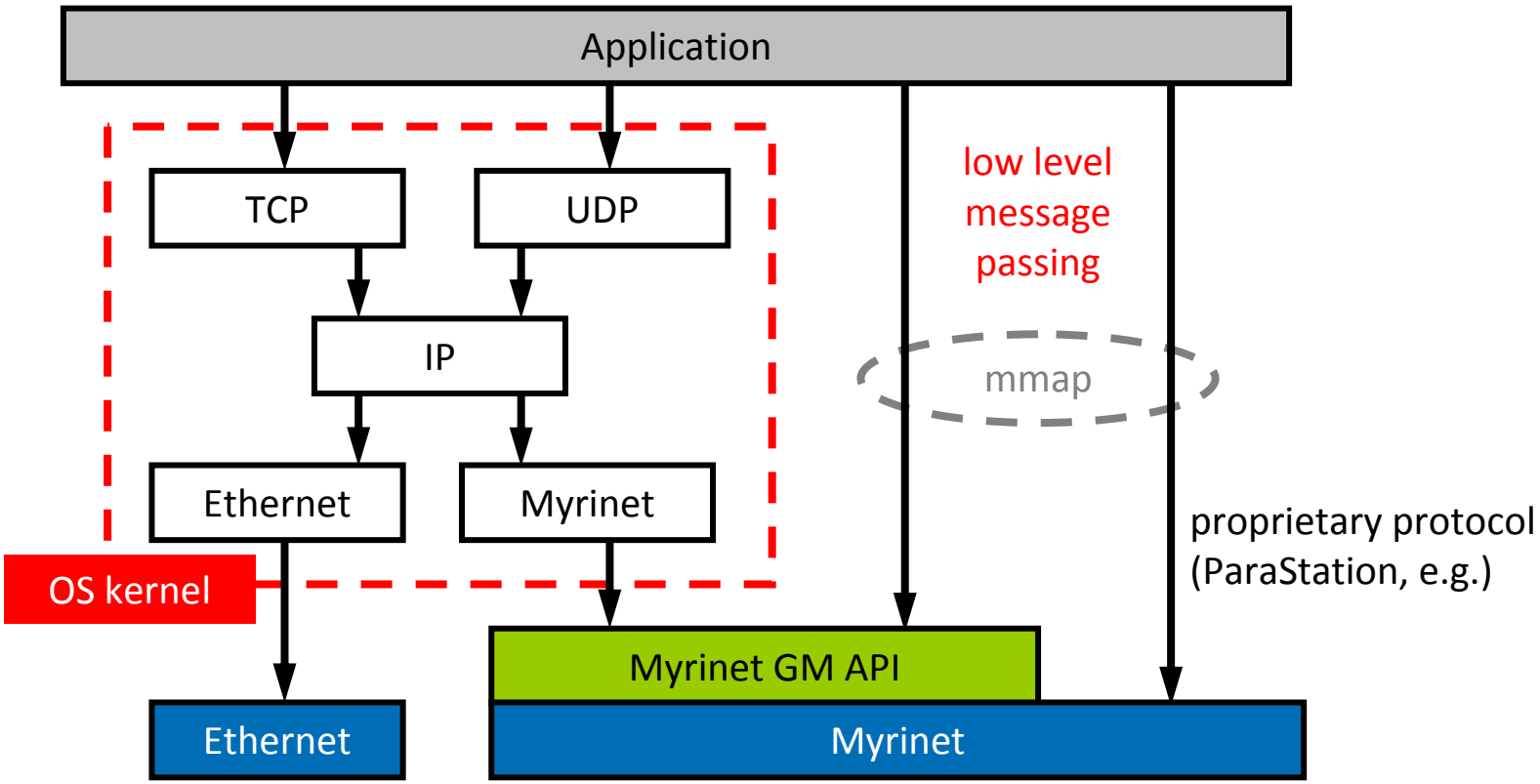


source: colfaxdirect.com

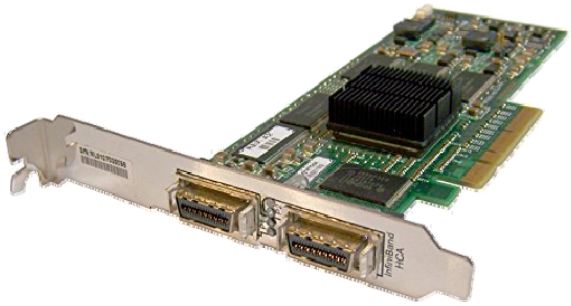
- Myrinet
  - developed by Myricom (1994) for clusters
  - particularly efficient due to
    - usage of onboard (NIC) processors for protocol offload and low-latency, kernel-bypass operations (ParaStation, e.g.)
    - highly scalable, cut-through switching
  - switches
    - consist of 256-port CLOS network
    - based on 32-port crossbar switch chipset
    - can be configured to support as many as 8,192 hosts
    - according to Myricom: used in nearly 38% of Top 500 supercomputers
    - NICs up to 2000 USD per card and switches >300 USD per port

Examples

- Myrinet (cont'd)
  - programming model



Examples

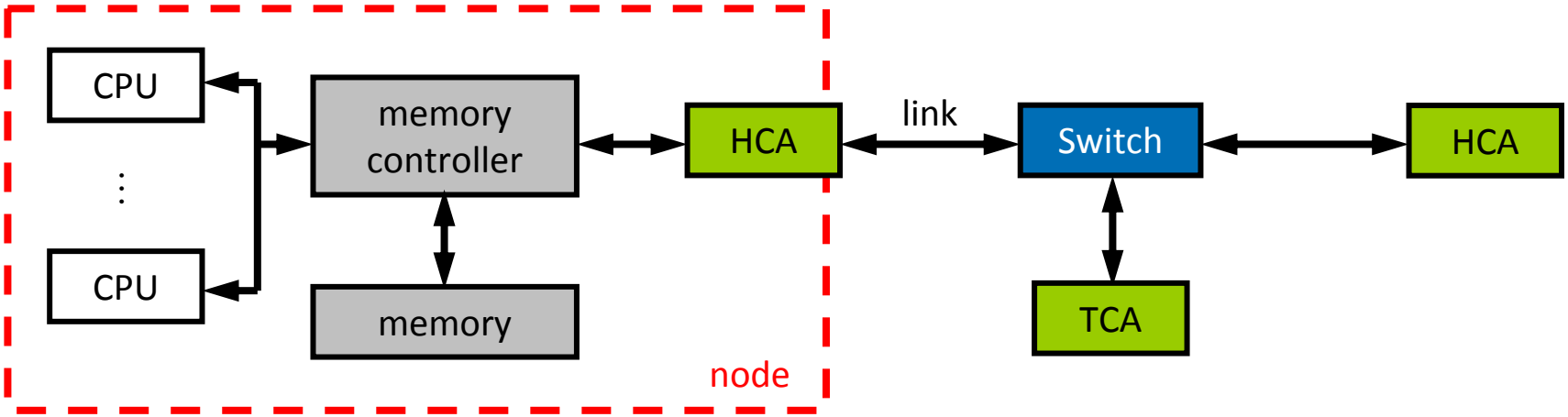


source: serversupply.com

- 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



- overview
  - definitions ✓
  - static topologies ✓
  - dynamic topologies ✓
  - examples ✓