Parallel and Distributed Programming I

Prof. Dr. Reinhard Wilhelm

December 5, 2005

Connection networks offer high-speed communication between processors and/or memory.
They are characterized by

Topology: how nodes are linked

- Static (direct) connection networks connect nodes directly by point-to-point channels,
- Dynamic (indirect) connection networks connect nodes indirectly by several channels and/or switches (bus-based or switch-based networks)

Routing Method: how transmission of messages through the network works

- Routing Algorithm: how a path is chosen
- Switching Strategy: how transmission is performed

Characteristics of Connection Networks

Network G with n nodes	Degree $g(G)$	Diameter $\delta(G)$	Edge connectivity $ec(G)$	Width of bisection $B(G)$
complete graph	$n-1$	1	$n-1$	$\left(\frac{n}{2}\right)^2$
linear array	2	$n-1$	1	1
ring	2	$\left\lfloor \frac{n}{2} \right\rfloor$	2	2
d-dimensional grid $(n = r^d)$	$2d$	$d(\sqrt[d]{n} - 1)$	d	$n^{\frac{d-1}{d}}$
d-dimensional torus $(n = r^d)$	$2d$	$d\left\lfloor \frac{\sqrt[d]{n}}{2} \right\rfloor$	$2d$	$2n^{\frac{d-1}{d}}$
k-dimensional hypercube $(n = 2^k)$	$\log n$	$\log n$	$\log n$	$\frac{n}{2}$
k-dimensional CCC-network $(n = k \cdot 2^k$ for $k \geq 3)$	3	$2k - 1 + \left\lfloor \frac{k}{2} \right\rfloor$	3	$\frac{n}{2k}$
complete binary tree $(n = 2^k - 1)$	3	$2\log \frac{n+1}{2}$	1	1
k-folded d-cube	$2d$	$d\left\lfloor \frac{k}{2} \right\rfloor$	$2d$	$2k^{d-1}$

Dynamic Connection Networks

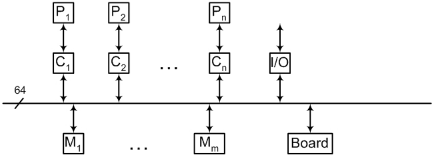
- Processors and memory modules are connected indirectly via a connection network.
- Connection networks are built from **switches** connected by physical **wires**.
- Connections for individual message transfers are **built dynamically** by setting switches correspondingly.
- Dynamic connection networks are mainly used for computer with **shared memory**.

Topological instantations of dynamic connection networks:

- Bus-networks
- Crossbar-networks
- Multi-stage switching networks

Bus-networks

- A bus-network consists of a **set of wires** over which data can be transported from a source to a target.
- **Example**: Bus with 64-bit wires, processors $P_1, \ldots, P_n$ with caches $C_1, \ldots, C_n$, memory modules $M_1, \ldots, M_m$:

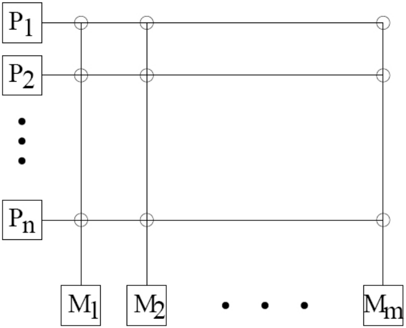


Characteristics/Limitations:

- At any time, only **one data transfer** can occur (**time sharing**).
- A **bus-arbiter** coordinates synchronous data transfer requests (**contention-bus**).
- Bus-networks are used for a **small number of processors** (32 up to maximal 64). Application in **SMP-Systems**, e.g. Sun Fire, IBM Regatta.

Crossbar-networks

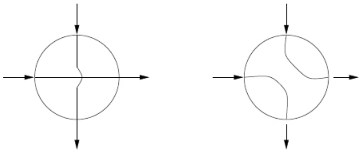
- An $(n \times m)$-crossbar-network consists of n **inputs**, m **outputs** and $n \cdot m$ **switches**.
  Example: n processors $P_1, \ldots, P_n$, m memory modules $M_1, \cdots, M_m$:



- For **every data transfer** (e.g. **memory request**) a **connection** from one input to one output is set up.

Crossbar-networks

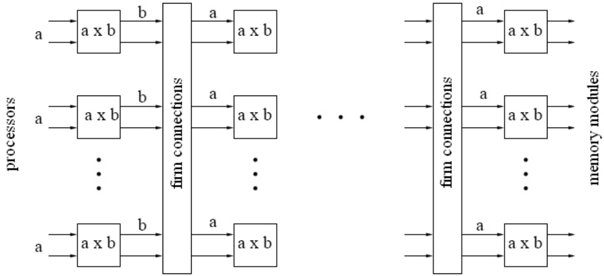
- **Possible settings of switches** correspond to required paths for data transfer or memory request:



- At any time, each memory module can satisfy **at most one memory request**;
  $\rightarrow$ in each column **at most one switch** may divert the direction
- A processor can perform **several memory requests simultaneously** on different memory modules.
- **Example** of crossbar-network:
  Fujitsu VPP500 (1992) uses a 224×224 crossbar-network.

Multi-stage Switching Networks

- Construction in **several layers (stages)** of **switches** and connecting wires
- Frequently, $(a \times b)$-crossbars are used as switches.
  Switches of neighboring stages are connected via **fixed connecting wires**:



- A memory request of a processor P to a memory module M is made by a **selection of a path** from P to M and the **setting of the switches** on this path.

Multi-stage Switching Networks

Criteria for the design of Multi-stage Networks:

- Costs: number of switches,
- HW Complexity: degree of the switches,
- Speed: depth of the network,
- Throughput: maximal number of messages that can be sent in parallel

Multi-stage Switching Networks

Representation as a graph:

- **directed acyclic graph**, nodes represent switches (and processors/memory modules), edges represent channels between switches.
- Processors and memory modules are designated nodes.
- The **connection of neighboring stages** is described by a **glueing function**, a **permutation** $\pi : \{1, \ldots, n\} \rightarrow \{1, \ldots, n\}$, so that outgoing wires of stage i are connected with incoming wires $(\pi(1), \ldots, \pi(n))$ of stage $i + 1$.

Mostly, multi-stage networks are built as follows:

- $n = 2^{k+1}$ **sources** and $n = 2^{k+1}$ **targets**;
- $k + 1$ **stages** of 2^k nodes each (thus, $(k + 1)2^k$ nodes in total);
- (2×2)-crossbar switches as switches.

Settings of switches in (2×2)-crossbar switches:

straight crossover upper broadcast lower broadcast

Examples: Omega-, Baseline- or Butterfly-Network

Multi-stage Switching Networks

Commonalities between $(n \times n)$ (k-dimensional) Omega-/Butterfly- and Baseline-Networks

- They connect $n = 2^{k+1}$ **sources** with $n = 2^{k+1}$ **targets**;
- They possess $k + 1$ **stages** of 2^k nodes each (thus, $(k + 1)2^k$ nodes in total);
- They use (2×2)-crossbar switches as switches.
- **Naming of switches** by pairs (α, i), where
  - $\alpha \in \{0, 1\}^{\log n-1}$ is a $(\log n - 1)$-bit word for selecting one of the $n/2$ switches on every stage;
  - $i \in \{0, \ldots, \log n - 1\}$ is the **number of the stage** of the switch.
  - Processor or memory-module nodes are named by (α, in) and (α, out), resp., with $\alpha \in \{0, 1\}^{\log n}$

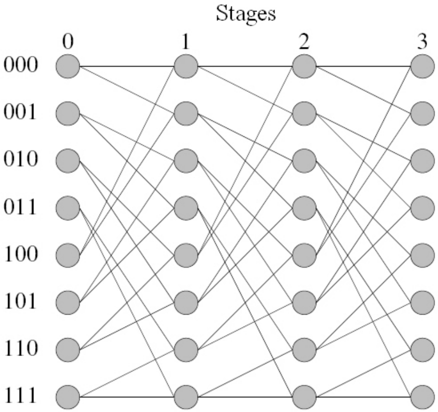
The **Glueing functions** are different for the three network types, the same for all stages in Omega-networks, and stage-dependent in the other two.

Multi-stage Switching Networks

Glueing function of an $(n \times n)$ **(k-dimensional)Omega-Network** (same for all stages): There exists an edge from switch (α, i) in stage i to **two switches** $(\beta_1, i+1)$ and $(\beta_2, i+1)$ in stage $i+1$, iff

1. β_1 is obtained from α by **cyclic left shift** or
2. β_2 is obtained from β_1 by inverting the **least significant (rightmost) bit**.

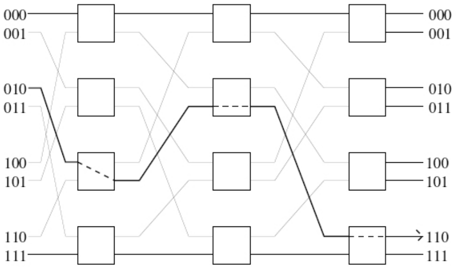
Example: (16×16)-Omega-Network with 4 stages and 8 switches at every stage:



Multi-stage Switching Networks

Routing in the Omega-Network

- In order to transfer a message from source α to target β the switch in stage k, $k = \{0, \ldots, \log n - 1\}$, considers the k-th bit β_k (counting from the left) and selects the output channel based on the following rule:
  1. if the k-th bit $\beta_k = 0$, the message is transferred via the **upper output channel of the switch**;
  2. if the k-th bit $\beta_k = 1$, the message is transferred via the **lower output channel of the switch**.
- the last $\log n - 1$ bits of α determine the switch in stage 0 to which α is connected and the first bit of α to which of its input channels.
- **Example**: (8×8)-Omega-Network with path from **(010,in)** to **(110,out)**:

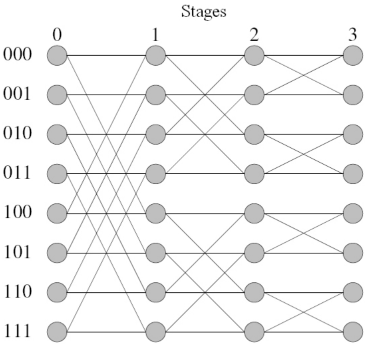


Multi-stage Switching Networks

Glueing function of the $(n \times n)$ (k**-dimensional) Butterfly-Network** (Banyan-Network) (depending on the stage):
There is an edge from switch (α, i) on stage i to switch $(\beta, i+1)$ on stage $i+1$, if

1. either α and β are identical (**straight edge**) or
2. α and β differ exactly in the $(i+1)$st bit position (counting from the left) (**cross edge**).

Example: (16×16) (3-dimensional) Butterfly-Network with 4 stages:



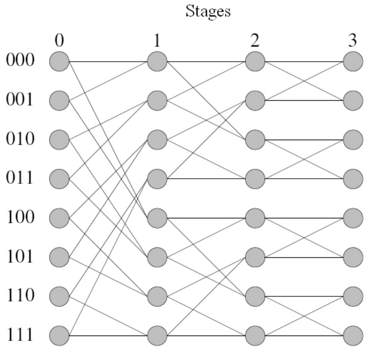
Multi-stage Switching Networks

Glueing function of the **$(n \times n)$ (k-dimensional) Baseline-Network** (depending on the stage):

Nodes (α, i) for $0 \leq i \leq k$ are connected with nodes $(\alpha', i+1)$, if:

1. the k-bit word α' is obtained from α using a **cyclic right shift** of the last $(k - i)$ bits of α, **or**
2. the k-bit word α' is obtained from α by inverting first the **rightmost** bit of α and then doing a cyclic right shift of the last $(k - i)$ bits.

Example: A 3-dimensional Baseline-Network with 4 stages:

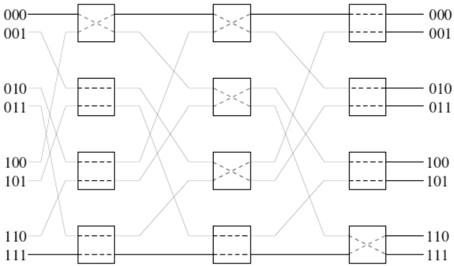


Multi-stage Switching Networks

- How many messages can be sent through the network from different inputs to different outputs **in parallel**?
- **Maximally** n for n sources and n targets.
  Example: message transfer with $n = 8$ in (8×8)-Omega-Network:

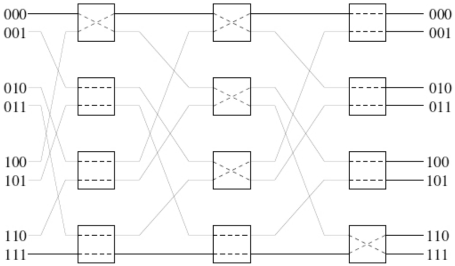
$$\pi^8 = \begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 7 & 3 & 0 & 1 & 2 & 5 & 4 & 6 \end{pmatrix}$$

- corresponding **setting of switches**:



Multi-stage Switching Networks

- Many such parallel transfers defined by permutations $\pi^n : \{0, \ldots, n-1\} \to \{0, \ldots, n-1\}$ cannot be realized, because they lead to **conflicts in the network**.
  Example: the two transmissions from (010,in) to (110,out) and from (000,in) to (111,out) lead to a conflict in an (8×8)-Omega-Network.



Multi-stage Switching Networks

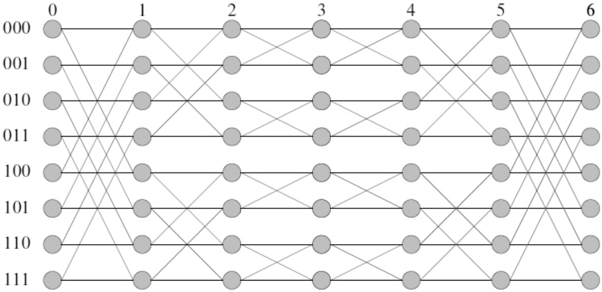
- Conflicts of this kind **cannot be resolved**, because for any pair of source and target there is only one path connecting them.
  Networks with this property are called **blocking networks**.
  Omega-, Butterfly-, and Baseline-Networks are all blocking networks.
- Conflicts in blocking networks can be resolved by distributing the message transmissions onto several runs through the network.
- In general, there are $n!$ permutations corresponding to combinations of (source,target).
- For $n/2 \log n$ switches, there are $2^{n/2 \log n} = n^{n/2}$ combinations of switch settings.

Multi-stage Switching Networks

k-dimensional Beneš-Network

- A k-dimensional Beneš-Network is built by cascading **two k-dimensional Butterfly-Networks**:
  - the **first** $k+1$ stages are a **Butterfly-Network**;
  - the **last** $k+1$ stages are a Butterfly-Network with the stages turned around;
  - $(k+1)$-th stage of the first Butterfly-Network coincides with the first stage of the inverted Butterfly-Network;

    $\rightarrow$ In total: $2k+1$ **stages** with $N = 2^k$ **switches per stage**.
- **Example**: Beneš-Network with 8 input nodes:



Multi-stage Switching Networks

- k-**dimensional Beneš-Network** is **non-blocking**.
- Each permutation $\pi^n : \{0, 1, \ldots, n-1\} \rightarrow \{0, 1, \ldots, n-1\}$ can be realized by n parallel communications.
- Example:

$$\pi^8 = \begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 5 & 3 & 4 & 7 & 0 & 1 & 2 & 6 \end{pmatrix}$$

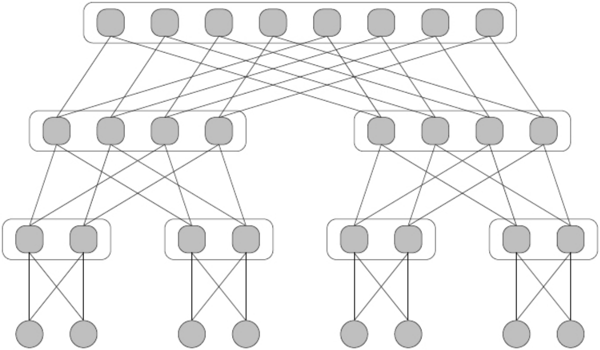
Multi-stage Switching Networks

Fat-Tree or dynamic tree

- The **basic structure** of a Fat-Tree is a complete binary tree:
  - The n **processors** are placed in **leaves** of the tree (level 0)
  - Trees in general don't provide big throughput. Why?
  - The **inner nodes** are **switches**, whose shapes depend on the level in the tree.
- **Construction of switches** in level i, $i = 1, \ldots, \log n$:
  - every switch has 2^i **incoming edges** and 2^i **outgoing edges**, $i = 1, \ldots, \log n$
  - internal realization, e.g. from 2^{i-1} switches with two incoming and outgoing edges each
    $\rightarrow$ every level i has $n/2$ switches that are **grouped** into $2^{\log n - i}$ nodes

Multi-stage Switching Networks

- **Example**: Fat-Tree with 16 processors and 4 levels:



(Processors are real leaves and are not shown on this picture)