# Part IV

## Low-Diameter Architectures

<table>
<thead>
<tr>
<th>Part I: Fundamental Concepts</th>
<th>Background and Motivation</th>
<th>1. Introduction to Parallelism</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Complexity and Models</td>
<td>2. A Taste of Parallel Algorithms</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3. Parallel Algorithm Complexity</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4. Models of Parallel Processing</td>
</tr>
<tr>
<td>Part II: Extreme Models</td>
<td>Abstract View of Shared Memory</td>
<td>5. PRAM and Basic Algorithms</td>
</tr>
<tr>
<td>Part III: Mesh-Based Architectures</td>
<td>Data Movement on 2D Arrays</td>
<td>7. Sorting and Selection Networks</td>
</tr>
<tr>
<td></td>
<td>Mesh Algorithms and Variants</td>
<td>8. Other Circuit-Level Examples</td>
</tr>
<tr>
<td>Part IV: Low-Diameter Architectures</td>
<td>The Hypercube Architecture</td>
<td>9. Sorting on a 2D Mesh or Torus</td>
</tr>
<tr>
<td></td>
<td>Hypercubic and Other Networks</td>
<td>10. Routing on a 2D Mesh or Torus</td>
</tr>
<tr>
<td></td>
<td></td>
<td>11. Numerical 2D Mesh Algorithms</td>
</tr>
<tr>
<td></td>
<td></td>
<td>12. Other Mesh-Related Architectures</td>
</tr>
<tr>
<td>Part V: Some Broad Topics</td>
<td>Coordination and Data Access</td>
<td>13. Hypercubes and Their Algorithms</td>
</tr>
<tr>
<td></td>
<td>Robustness and Ease of Use</td>
<td>14. Sorting and Routing on Hypercubes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>15. Other Hypercubic Architectures</td>
</tr>
<tr>
<td></td>
<td></td>
<td>16. A Sampler of Other Networks</td>
</tr>
<tr>
<td>Part VI: Implementation Aspects</td>
<td>Control-Parallel Systems</td>
<td>17. Emulation and Scheduling</td>
</tr>
<tr>
<td></td>
<td>Data Parallelism and Conclusion</td>
<td>18. Data Storage, Input, and Output</td>
</tr>
<tr>
<td></td>
<td></td>
<td>19. Reliable Parallel Processing</td>
</tr>
<tr>
<td></td>
<td></td>
<td>20. System and Software Issues</td>
</tr>
<tr>
<td></td>
<td></td>
<td>21. Shared-Memory MIMD Machines</td>
</tr>
<tr>
<td></td>
<td></td>
<td>22. Message-Passing MIMD Machines</td>
</tr>
<tr>
<td></td>
<td></td>
<td>23. Data-Parallel SIMD Machines</td>
</tr>
<tr>
<td></td>
<td></td>
<td>24. Past, Present, and Future</td>
</tr>
</tbody>
</table>
About This Presentation

This presentation is intended to support the use of the textbook *Introduction to Parallel Processing: Algorithms and Architectures* (Plenum Press, 1999, ISBN 0-306-45970-1). It was prepared by the author in connection with teaching the graduate-level course ECE 254B: Advanced Computer Architecture: Parallel Processing, at the University of California, Santa Barbara. Instructors can use these slides in classroom teaching and for other educational purposes. Any other use is strictly prohibited. © Behrooz Parhami

<table>
<thead>
<tr>
<th>Edition</th>
<th>Released</th>
<th>Revised</th>
<th>Revised</th>
<th>Revised</th>
</tr>
</thead>
<tbody>
<tr>
<td>First</td>
<td>Spring 2005</td>
<td>Spring 2006</td>
<td>Fall 2008</td>
<td>Fall 2010</td>
</tr>
<tr>
<td></td>
<td>Winter 2013</td>
<td>Winter 2014</td>
<td>Winter 2016</td>
<td>Winter 2019</td>
</tr>
</tbody>
</table>
Low-Diameter Architectures

Study the hypercube and related interconnection schemes:
- Prime example of low-diameter (logarithmic) networks
- Theoretical properties, realizability, and scalability
- Complete our view of the “sea of interconnection nets”

Topics in This Part

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>Hypercubes and Their Algorithms</td>
</tr>
<tr>
<td>14</td>
<td>Sorting and Routing on Hypercubes</td>
</tr>
<tr>
<td>15</td>
<td>Other Hypercubic Architectures</td>
</tr>
<tr>
<td>16</td>
<td>A Sampler of Other Networks</td>
</tr>
</tbody>
</table>
# 13 Hypercubes and Their Algorithms

Study the hypercube and its topological/algorithmic properties:
- Develop simple hypercube algorithms (more in Ch. 14)
- Learn about embeddings and their usefulness

## Topics in This Chapter

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>13.1</td>
<td>Definition and Main Properties</td>
</tr>
<tr>
<td>13.2</td>
<td>Embeddings and Their Usefulness</td>
</tr>
<tr>
<td>13.3</td>
<td>Embedding of Arrays and Trees</td>
</tr>
<tr>
<td>13.4</td>
<td>A Few Simple Algorithms</td>
</tr>
<tr>
<td>13.5</td>
<td>Matrix Multiplication</td>
</tr>
<tr>
<td>13.6</td>
<td>Inverting a Lower-Triangular Matrix</td>
</tr>
</tbody>
</table>
13.1 Definition and Main Properties

Begin studying networks that are intermediate between diameter-1 complete network and diameter-$p^{1/2}$ mesh

<table>
<thead>
<tr>
<th>Sublogarithmic diameter</th>
<th>Superlogarithmic diameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$\sqrt{n}$</td>
</tr>
<tr>
<td>2</td>
<td>$n/2$</td>
</tr>
<tr>
<td>log $n$ / log log $n$</td>
<td>$n - 1$</td>
</tr>
</tbody>
</table>

- Complete network
- PDN
- Star, pancake
- Binary tree, hypercube
- Torus
- Ring
- Linear array
Very-High-Dimensional Meshes and Tori

$qD$ mesh with $m$ processors along each dimension: $p = m^q$

- **Diameter**: $D = q(m - 1) = q(p^{1/q} - 1)$
- **Bisection width**: $B = p^{1 - 1/q}$ when $m = p^{1/q}$ is even
- **Node degree**: $d = 2q$

$qD$ torus with $m$ processors along each dimension = $m$-ary $q$-cube

What happens when $q$ becomes as large as $\log_2 p$?

- **Diameter**: $D = q(p^{1/q} - 1) = \log_2 p \ast = O(\log p)$
- **Bisection width**: $B = p^{1 - 1/q} = p / p^{1/q} = p/2 \ast = O(p)$
- **Node degree**: $d = 2q = \log_2 p \ast\ast = O(\log p)$

$qD$ torus with 2 processors along each dimension same as mesh

---

* What is the value of $m = p^{1/q} = p^{1/\log_2 p}$?

  - $m = p^{1/\log_2 p}$
  - $m^{\log_2 p} = p$
  - $m = 2$

** When $m = 2$, node degree becomes $q$ instead of $2q$
**Hypercube and Its History**

Binary tree has logarithmic diameter, but small bisection
Hypercube has a much larger bisection
Hypercube is a mesh with the maximum possible number of dimensions

\[ 2 \times 2 \times 2 \times \ldots \times 2 \]

\[ q = \log_2 p \]

We saw that increasing the number of dimensions made it harder to design and visualize algorithms for the mesh
Oddly, at the extreme of \( \log_2 p \) dimensions, things become simple again!

**Brief history of the hypercube (binary \( q \)-cube) architecture**

Concept developed: early 1960s [Squi63]
Direct (single-stage) and indirect (multistage) versions: mid 1970s
    Initial proposals [Peas77], [Sull77] included no hardware
Caltech’s 64-node Cosmic Cube: early 1980s [Seit85]
    Introduced an elegant solution to routing (wormhole switching)
Several commercial machines: mid to late 1980s
    Intel PSC (personal supercomputer), CM-2, nCUBE (Section 22.3)
Basic Definitions

Hypercube is generic term; 3-cube, 4-cube, . . . , $q$-cube in specific cases

Fig. 13.1
The recursive structure of binary hypercubes.

Parameters:
$p = 2^q$
$B = \frac{p}{2} = 2^{q-1}$
$D = q = \log_2 p$
$d = q = \log_2 p$
The 64-Node Hypercube

Only sample wraparound links are shown to avoid clutter

Isomorphic to the $4 \times 4 \times 4$ 3D torus (each has $64 \times 6/2$ links)
Neighbors of a Node in a Hypercube

$x_{q-1}x_{q-2} \ldots x_2x_1x_0$ ID of node $x$

$x_{q-1}x_{q-2} \ldots x_2x_1x_0'$ dimension-0 neighbor; $N_0(x)$

$x_{q-1}x_{q-2} \ldots x_2x_1'x_0'$ dimension-1 neighbor; $N_1(x)$

. . .

$x_{q-1}'x_{q-2} \ldots x_2x_1x_0$ dimension-$(q-1)$ neighbor; $N_{q-1}(x)$

Nodes whose labels differ in $k$ bits (at Hamming distance $k$) connected by shortest path of length $k$

Both node- and edge-symmetric

Strengths: symmetry, log diameter, and linear bisection width

Weakness: poor scalability
13.2 Embeddings and Their Usefulness

**Dilation:** Longest path onto which an edge is mapped (routing slowdown)

**Congestion:** Max number of edges mapped onto one edge (contention slowdown)

**Load factor:** Max number of nodes mapped onto one node (processing slowdown)

Fig. 13.2 Embedding a seven-node binary tree into 2D meshes of various sizes.

Expansion: ratio of the number of nodes (9/7, 8/7, and 4/7 here)
13.3 Embedding of Arrays and Trees

Alternate inductive proof: Hamiltonicity of the $q$-cube is equivalent to the existence of a $q$-bit Gray code

Basis: $q$-bit Gray code beginning with the all-0s codeword and ending with $10^{q-1}$ exists for $q = 2$: 00, 01, 11, 10

Fig. 13.3   Hamiltonian cycle in the $q$-cube.
Mesh/Torus Embedding in a Hypercube

Fig. 13.5  The $4 \times 4$ mesh/torus is a subgraph of the 4-cube.

Is a mesh or torus a subgraph of the hypercube of the same size?

We prove this to be the case for a torus (and thus for a mesh)
A tool used in our proof

Product graph $G_1 \times G_2$:
- Has $n_1 \times n_2$ nodes
- Each node is labeled by a pair of labels, one from each component graph
- Two nodes are connected if either component of the two nodes were connected in the component graphs

The $2^a \times 2^b \times 2^c \ldots$ torus is the product of $2^a$-, $2^b$-, $2^c$-, \ldots node rings
The $(a + b + c + \ldots)$-cube is the product of $a$-cube, $b$-cube, $c$-cube, \ldots
The $2^q$-node ring is a subgraph of the $q$-cube
If a set of component graphs are subgraphs of another set, the product graphs will have the same relationship
Embedding Trees in the Hypercube

The \((2^q - 1)\)-node complete binary tree is not a subgraph of the \(q\)-cube.

Proof by contradiction based on the parity of node label weights (number of 1s is the labels).

The \(2^q\)-node double-rooted complete binary tree is a subgraph of the \(q\)-cube.

Fig. 13.6 The \(2^q\)-node double-rooted complete binary tree is a subgraph of the \(q\)-cube.
A Useful Tree Embedding in the Hypercube

The \((2^q - 1)\)-node complete binary tree can be embedded into the \((q - 1)\)-cube.

Despite the load factor of \(q\), many tree algorithms entail no slowdown.

Fig. 13.7 Embedding a 15-node complete binary tree into the 3-cube.
13.4 A Few Simple Algorithms

Semigroup computation on the $q$-cube
Processor $x$, $0 \leq x < p$ do $t[x] := v[x]$
   {initialize “total” to own value}
for $k=0$ to $q-1$ processor $x$, $0 \leq x < p$, do
  get $y := t[N_k(x)]$
  set $t[x] := t[x] \otimes y$
endfor

Commutativity of the operator $\otimes$ is implicit here.
How can we remove this assumption?

Fig. 13.8 Semigroup computation on a 3-cube.
Parallel Prefix Computation

Parallel prefix computation on the $q$-cube
Processor $x$, $0 \leq x < p$, do $t[x] := u[x] := v[x]$
   {initialize subcube “total” and partial prefix}
for $k=0$ to $q-1$ processor $x$, $0 \leq x < p$, do
   get $y := t[N_k(x)]$
   set $t[x] := t[x] \otimes y$
   if $x > N_k(x)$ then set $u[x] := y \otimes u[x]$
endfor

Commutativity of the operator $\otimes$ is implicit in this algorithm as well.
How can we remove this assumption?

Fig. 13.8 Semigroup computation on a 3-cube.
Sequence Reversal on the Hypercube

Reversing a sequence on the $q$-cube
for $k = 0$ to $q - 1$ Processor $x$, $0 \leq x < p$, do
get $y := v[N_k(x)]$
set $v[x] := y$
endfor

Fig. 13.11  Sequence reversal on a 3-cube.
Ascend, Descend, and Normal Algorithms

Graphical depiction of ascend, descend, and normal algorithms.
13.5 Matrix Multiplication

\[ p = m^3 = 2^q \] processors, indexed as \( ijk \) (with three \( q/3 \)-bit segments)

1. Place elements of \( A \) and \( B \) in registers \( R_A \) & \( R_B \) of \( m^2 \) processors with the IDs \( 0jk \)

2. Replicate inputs: communicate across 1/3 of the dimensions

3, 4. Rearrange the data by communicating across the remaining 2/3 of dimensions so that processor \( ijk \) has \( A_{ji} \) and \( B_{ik} \)

5. Move \( C_{jk} \) to processor \( 0jk \)

Fig. 13.12 Multiplying two \( 2 \times 2 \) matrices on a 3-cube.
The algorithm involves communication steps in three loops, each with $q/3$ iterations (in one of the 4 loops, 2 values are exchanged per iteration)

$$T_{\text{mul}}(m, m^3) = O(q) = O(\log m)$$

Analysis in the case of block matrix multiplication ($m \times m$ matrices):
Matrices are partitioned into $p^{1/3} \times p^{1/3}$ blocks of size $(m/p^{1/3}) \times (m/p^{1/3})$
Each communication step deals with $m^2/p^{2/3}$ block elements
Each multiplication entails $2m^3/p$ arithmetic operations

$$T_{\text{mul}}(m, p) = m^2/p^{2/3} \times O(\log p) + 2m^3/p$$

Communication

Computation

Dim 0

Dim 1

Dim 2
13.6 Inverting a Lower-Triangular Matrix

For $A = \begin{pmatrix} B & 0 \\ C & D \end{pmatrix}$ we have $A^{-1} = \begin{pmatrix} B^{-1} & 0 \\ -D^{-1}CB^{-1} & D^{-1} \end{pmatrix}$.

Because $B$ and $D$ are both lower triangular, the same algorithm can be used recursively to invert them in parallel.

$$T_{\text{inv}}(m) = T_{\text{inv}}(m/2) + 2T_{\text{mul}}(m/2) = T_{\text{inv}}(m/2) + O(\log m) = O(\log^2 m)$$
Recursive Lower-Triangular Matrix Inversion Algorithm

For $A = \begin{bmatrix} B & 0 \\ C & D \end{bmatrix}$ we have $A^{-1} = \begin{bmatrix} B^{-1} & 0 \\ -D^{-1}CB^{-1} & D^{-1} \end{bmatrix}$

Invert lower-triangular matrices $B$ and $D$

Send $B^{-1}$ and $D^{-1}$ to the subcube holding $C$

Compute $-D^{-1}CB^{-1}$ to in the subcube

$q$-cube and its four $(q-2)$-subcubes
14 Sorting and Routing on Hypercubes

Study routing and data movement problems on hypercubes:
- Learn about limitations of oblivious routing algorithms
- Show that bitonic sorting is a good match to hypercube

Topics in This Chapter

<table>
<thead>
<tr>
<th>14.1</th>
<th>Defining the Sorting Problem</th>
</tr>
</thead>
<tbody>
<tr>
<td>14.2</td>
<td>Bitonic Sorting on a Hypercube</td>
</tr>
<tr>
<td>14.3</td>
<td>Routing Problems on a Hypercube</td>
</tr>
<tr>
<td>14.4</td>
<td>Dimension-Order Routing</td>
</tr>
<tr>
<td>14.5</td>
<td>Broadcasting on a Hypercube</td>
</tr>
<tr>
<td>14.6</td>
<td>Adaptive and Fault-Tolerant Routing</td>
</tr>
</tbody>
</table>
14.1 Defining the Sorting Problem

Review of the hypercube:

Fully symmetric with respect to dimensions

Typical computations involve communication across all dimensions

Dimension-order communication is known as “ascend” or “descend” (0 up to \(q-1\), or \(q-1\) down to 0)

Due to symmetry, any hypercube dimension can be labeled as 0, any other as 1, and so on
Hypercube Sorting: Goals and Definitions

Arrange data in order of processor ID numbers (labels)

The ideal parallel sorting algorithm:

\[ T(p) = \Theta((n \log n)/p) \]

This ideal has not been achieved in all cases for the hypercube

1-1 sorting (\(p\) items to sort, \(p\) processors)

Batcher’s odd-even merge or bitonic sort: \(O(\log^2 p)\) time
\(O(\log p)\)-time deterministic algorithm not known

\(k\)-\(k\) sorting (\(n = kp\) items to sort, \(p\) processors)

Optimal algorithms known for \(n >> p\) or when average running time is considered (randomized)
Hypercube Sorting: Attempts and Progress

No bull’s eye yet!

There are three categories of practical sorting algorithms:

1. Deterministic 1-1, $O(\log^2 p)$-time

2. Deterministic $k$-$k$, optimal for $n >> p$ (that is, for large $k$)

3. Probabilistic (1-1 or $k$-$k$)

Pursuit of $O(\log p)$-time algorithm is of theoretical interest only
Bitonic Sequences

Bitonic sequence:

1 3 3 4 6 6 6 2 2 1 0 0

Rises, then falls

8 7 7 6 6 6 5 4 6 8 8 9

Falls, then rises

8 9 8 7 7 6 6 6 5 4 6 8

The previous sequence, right-rotated by 2

In Chapter 7, we designed bitonic sorting nets. Bitonic sorting is ideally suited to hypercube.

Fig. 14.1 Examples of bitonic sequences.
Sorting a Bitonic Sequence on a Linear Array

Time needed to sort a bitonic sequence on a $p$-processor linear array:

$$B(p) = p + p/2 + p/4 + \ldots + 2 = 2p - 2$$

Not competitive, because we can sort an arbitrary sequence in $2p - 2$ unidirectional communication steps using odd-even transposition.

Fig. 14.2 Sorting a bitonic sequence on a linear array.
Bitonic Sorting on a Linear Array

Sorting an arbitrary sequence of length $p$:
\[ T(p) = T(p/2) + B(p) = T(p/2) + 2p - 2 = 4p - 4 - 2 \log_2 p \]

Alternate derivation:
\[ T(p) = B(2) + B(4) + \ldots + B(p) = 2 + 6 + \ldots + (2p - 2) = 4p - 4 - 2 \log_2 p \]

Recall that
\[ B(p) = 2p - 2 \]
Visualizing Bitonic Sorting on a Linear Array

Initial data sequence, stored one per processor

Phase 1: Sort half-arrays in opposite directions

Phase 2: Shift data leftward to compare half-arrays

Phase 3: Send larger item in each pair to the right

Phase 4: Sort each bitonic half-sequence separately
14.2 Bitonic Sorting on a Hypercube

For linear array, the 4p-step bitonic sorting algorithm is inferior to odd-even transposition which requires p compare-exchange steps (or 2p unidirectional communications)

The situation is quite different for a hypercube

**Sorting a bitonic sequence on a hypercube:** Compare-exchange values in the upper subcube (nodes with $x_{q-1} = 1$) with those in the lower subcube ($x_{q-1} = 0$); sort the resulting bitonic half-sequences

$$B(q) = B(q - 1) + 1 = q$$

**Complexity:** 2q communication steps

Sorting a bitonic sequence of size n on q-cube, $q = \log_2 n$

for $l = q - 1$ downto 0 processor $x$, $0 \leq x < p$, do

if $x_l = 0$

then get $y := v[N_l(x)]$; keep min($v(x), y$); send max($v(x), y$) to $N_l(x)$

endif

defor

This is a “descend” algorithm
Bitonic Sorting on a Hypercube

Fig. 14.4 Sorting a bitonic sequence of size 8 on the 3-cube.

\[ T(q) = T(q - 1) + B(q) = T(q - 1) + q = q(q + 1)/2 = O(\log^2 p) \]
14.3 Routing Problems on a Hypercube

Recall the following categories of routing algorithms:

- **Off-line**: Routes precomputed, stored in tables
- **On-line**: Routing decisions made on the fly
- **Oblivious**: Path depends only on source & destination
- **Adaptive**: Path may vary by link and node conditions

**Good news for routing on a hypercube:**
Any 1-1 routing problem with $p$ or fewer packets can be solved in $O(\log p)$ steps, using an off-line algorithm; this is a consequence of there being many paths to choose from.

**Bad news for routing on a hypercube:**
Oblivious routing requires $\Omega(p^{1/2}/\log p)$ time in the worst case
(only slightly better than mesh)
In practice, actual routing performance is usually much closer to the log-time best case than to the worst case.
Theorem 14.1: Let $G = (V, E)$ be a $p$-node, degree-$d$ network. Any oblivious routing algorithm for routing $p$ packets in $G$ needs $\Omega(p^{1/2}/d)$ worst-case time.

Proof Sketch: Let $P_{u,v}$ be the unique path used for routing messages from $u$ to $v$.

There are $p(p-1)$ possible paths for routing among all node pairs.

These paths are predetermined and do not depend on traffic within the network.

Our strategy: find $k$ node pairs $u_i, v_i$ ($1 \leq i \leq k$) such that $u_i \neq u_j$ and $v_i \neq v_j$ for $i \neq j$, and $P_{u_i,v_i}$ all pass through the same edge $e$.

Because $\leq 2$ packets can go through a link in one step, $\Omega(k)$ steps will be needed for some 1-1 routing problem.

The main part of the proof consists of showing that $k$ can be as large as $p^{1/2}/d$. 
### 14.4 Dimension-Order Routing

Source: 01011011  
Destination: 11010110  
Differences: \( ^1 \) \( ^2 \) \( ^1 \)  
Path: 01011011  
\( \overline{1} \)011011  
11010111  
11010111  
11010110

**Unfolded hypercube** (indirect cube, butterfly) facilitates the discussion, visualization, and analysis of routing algorithms.

*Fig. 14.5* Unfolded 3-cube or the 32-node butterfly network.

Dimension-order routing between nodes \( i \) and \( j \) in \( q \)-cube can be viewed as routing from node \( i \) in column 0 (\( q \)) to node \( j \) in column \( q \) (0) of the butterfly.
Self-Routing on a Butterfly Network

From node 3 to 6: routing tag = 011 ⊕ 110 = 101 “cross-straight-cross”
From node 3 to 5: routing tag = 011 ⊕ 101 = 110 “cross-cross-straight”
From node 6 to 1: routing tag = 110 ⊕ 001 = 111 “cross-cross-cross”

Fig. 14.6 Example dimension-order routing paths.

Number of cross links taken = length of path in hypercube
Butterfly Is Not a Permutation Network

Fig. 14.7 Packing is a “good” routing problem for dimension-order routing on the hypercube.

Fig. 14.8 Bit-reversal permutation is a “bad” routing problem for dimension-order routing on the hypercube.
Why Bit-Reversal Routing Leads to Conflicts?

Consider the \((2a + 1)\)-cube and messages that must go from nodes

\[
\begin{align*}
0 & 0 0 \ldots 0 \\
\_ & \_ \_ x_1 x_2 \ldots x_{a-1} x_a \_ & \_ \_ \\
\_ & \_ \_ \_ & \_ & \_ \_ \_ \_ \_ \_ 0 0 0 \ldots 0 \\
\_ & \_ \_ \_ \_ \_ \_ \_ \_ \_ a + 1 \text{ zeros} & \_ & \_ \_ \_ \_ \_ \_ \_ \_ \_ a + 1 \text{ zeros}
\end{align*}
\]

If we route messages in dimension order, starting from the right end, all of these \(2^a = \Theta(p^{1/2})\) messages will pass through node 0.

**Consequences of this result:**

1. The \(\Theta(p^{1/2})\) delay is even worse than \(\Omega(p^{1/2}/d)\) of Theorem 14.1.
2. Besides delay, large buffers are needed within the nodes.

**True or false?** If we limit nodes to a constant number of message buffers, then the \(\Theta(p^{1/2})\) bound still holds, except that messages are queued at several levels before reaching node 0.

**Bad news (false):** The delay can be \(\Theta(p)\) for some permutations.

**Good news:** Performance usually much better; i.e., \(\log_2 p + o(\log p)\)
Wormhole Routing on a Hypercube

Good/bad routing problems are good/bad for wormhole routing as well.
Dimension-order routing is deadlock-free.
14.5 Broadcasting on a Hypercube

Flooding: applicable to any network with all-port communication

00000
00001, 00010, 00100, 01000, 10000
00011, 00101, 01001, 10001, 00110, 01010, 10010, 01100, 10100, 11000
00111, 01011, 10011, 01101, 10101, 11001, 01110, 10110, 11010, 11100
01111, 10111, 11011, 11101, 11110
11111

Source node
Neighbors of source
Distance-2 nodes
Distance-3 nodes
Distance-4 nodes
Distance-5 node

Binomial broadcast tree with single-port communication

Fig. 14.9 The binomial broadcast tree for a 5-cube.
Hypercube Broadcasting Algorithms

Fig. 14.10
Three hypercube broadcasting schemes as performed on a 4-cube.

Binomial-tree scheme (nonpipelined)

Pipelined binomial-tree scheme

Johnsson & Ho’s method

To avoid clutter, only A shown
14.6 Adaptive and Fault-Tolerant Routing

There are up to $q$ node-disjoint and edge-disjoint shortest paths between any node pairs in a $q$-cube.

Thus, one can route messages around congested or failed nodes/links.

A useful notion for designing adaptive wormhole routing algorithms is that of virtual communication networks.

Each of the two subnetworks in Fig. 14.11 is acyclic.

Hence, any routing scheme that begins by using links in subnet 0, at some point switches the path to subnet 1, and from then on remains in subnet 1, is deadlock-free.
The fault diameter of the $q$-cube is $q + 1$.

Robustness of the Hypercube

Rich connectivity provides many alternate paths for message routing.

The node that is furthest from S is not its diametrically opposite node in the fault-free hypercube.

The fault diameter of the $q$-cube is $q + 1$. 
15 Other Hypercubic Architectures

Learn how the hypercube can be generalized or extended:
• Develop algorithms for our derived architectures
• Compare these architectures based on various criteria

<table>
<thead>
<tr>
<th>Topics in This Chapter</th>
</tr>
</thead>
<tbody>
<tr>
<td>15.1 Modified and Generalized Hypercubes</td>
</tr>
<tr>
<td>15.2 Butterfly and Permutation Networks</td>
</tr>
<tr>
<td>15.3 Plus-or-Minus-$2^i$ Network</td>
</tr>
<tr>
<td>15.4 The Cube-Connected Cycles Network</td>
</tr>
<tr>
<td>15.5 Shuffle and Shuffle-Exchange Networks</td>
</tr>
<tr>
<td>15.6 That’s Not All, Folks!</td>
</tr>
</tbody>
</table>
15.1 Modified and Generalized Hypercubes

Fig. 15.1 Deriving a twisted 3-cube by redirecting two links in a 4-cycle.

Diameter is one less than the original hypercube
Folded Hypercubes

**Fig. 15.2** Deriving a folded 3-cube by adding four diametral links.

Diameter is half that of the original hypercube

![Folded 3-cube with Dim-0 links removed](image)

![Rotate 180 degrees](image)

![After renaming, diametral links replace dim-0 links](image)

**Fig. 15.3** Folded 3-cube viewed as 3-cube with a redundant dimension.
Generalized Hypercubes

A hypercube is a power or homogeneous product network

\( q \)-cube = \((0-0)^q\); \( q \) th power of \( K_2 \)

Generalized hypercube = \( q \)th power of \( K_r \)
(node labels are radix-\( r \) numbers)
Node \( x \) is connected to \( y \) iff \( x \) and \( y \) differ in one digit
Each node has \( r - 1 \) dimension-\( k \) links

Example: radix-4 generalized hypercube
Node labels are radix-4 numbers
Bijective Connection Graphs

Beginning with a $c$-node seed network, the network size is recursively doubled in each step by linking nodes in the two halves via an arbitrary one-to-one mapping. Number of nodes = $c \, 2^q$

Hypercube is a special case, as are many hypercube variant networks (twisted, crossed, mobius, . . . , cubes)

Special case of $c = 1$:
Diameter upper bound is $q$
Diameter lower bound is an open problem (it is better than $\lceil q + 1 \rceil/2$)
15.2 Butterfly and Permutation Networks

Fig. 7.4 Butterfly and wrapped butterfly networks.

- Butterfly networks: $2^q$ rows, $q + 1$ columns
- Wrapped butterfly networks: $2^q$ rows, $q$ columns
Structure of Butterfly Networks

Switching these two row pairs converts this to the original butterfly network. Changing the order of stages in a butterfly is thus equivalent to a relabeling of the rows (in this example, row xyz becomes row xzy).

Fig. 15.5 Butterfly network with permuted dimensions.

The 16-row butterfly network.
Fat Trees

Fig. 15.6 Two representations of a fat tree.

Skinny tree?

Front view:
Binary tree

Side view:
Inverted binary tree

Fig. 15.7 Butterfly network redrawn as a fat tree.
Butterfly as Multistage Interconnection Network

Generalization of the butterfly network
- High-radix or \( m \)-ary butterfly, built of \( m \times m \) switches
- Has \( m^q \) rows and \( q + 1 \) columns (\( q \) if wrapped)

Fig. 6.9 Example of a multistage memory access network

Fig. 15.8 Butterfly network used to connect modules that are on the same side
Beneš Network

A $2^q$-row Beneš network:
- Can route any $2^q \times 2^q$ permutation
- It is “rearrangeable”

Fig. 15.9  Beneš network formed from two back-to-back butterflies.
Routing Paths in a Beneš Network

Fig. 15.10 Another example of a Beneš network.

To which memory modules can we connect proc 4 without rearranging the other paths?

What about proc 6?
15.3 Plus-or-Minus-2^i Network

Fig. 15.11 Two representations of the eight-node PM2I network.

The hypercube is a subgraph of the PM2I network
Unfolded PM2I Network

Data manipulator network was used in Goodyear MPP, an early SIMD parallel machine.

“Augmented” means that switches in a column are independent, as opposed to all being set to same state (simplified control).

Fig. 15.12 Augmented data manipulator network.
15.4 The Cube-Connected Cycles Network

The cube-connected cycles network (CCC) is the earliest example of what later became known as X-connected cycles, with X being an arbitrary network.

Transform a $p$-node, degree-$d$ network into a $pd$-node, degree-3 network by replacing each of the original network nodes with a $d$-node cycle.

Original degree-8 node in a network, with its links labeled $L_0$ through $L_7$.

Replacement 8-node cycle, with each of its 8 nodes accommodating one of the links $L_0$ through $L_7$. 
Replacing each node of a high-dimensional $q$-cube by a cycle of length $q$ is how CCC was originally proposed.
Another View of the Cube-Connected Cycles Network

The cube-connected cycles network (CCC) can be viewed as a simplified wrapped butterfly whose node degree is reduced from 4 to 3.

Fig. 15.13 A wrapped butterfly (left) converted into cube-connected cycles.
Emulation of a 6-Cube by a 64-Node CCC

With proper node mapping, dim-0 and dim-1 neighbors of each node will map onto the same cycle.

Suffices to show how to communicate along other dimensions of the 6-cube.
Emulation of Hypercube Algorithms by CCC

Node \((x, j)\) is communicating along dimension \(j\); after the next rotation, it will be linked to its dimension-\((j + 1)\) neighbor.

Ascend, descend, and normal algorithms.

Fig. 15.15 CCC emulating a normal hypercube algorithm.
15.5 Shuffle and Shuffle-Exchange Networks

Fig. 15.16 Shuffle, exchange, and shuffle-exchange connectivities.
Shuffle-Exchange Network

Fig. 15.17 Alternate views of an eight-node shuffle–exchange network.
Routing in Shuffle-Exchange Networks

In the $2^q$-node shuffle network, node $x = x_{q-1}x_{q-2} \ldots x_2x_1x_0$ is connected to $x_{q-2} \ldots x_2x_1x_0x_{q-1}$ (cyclic left-shift of $x$)

In the $2^q$-node shuffle-exchange network, node $x$ is additionally connected to $x_{q-2} \ldots x_2x_1x_0x_{q-1}'$

01011011 Source
11010110 Destination
^ ^ ^ Positions that differ

01011011 Shuffle to 10110110 Exchange to 10110111
10110111 Shuffle to 01101111
01101111 Shuffle to 11011110
11011110 Shuffle to 10111101
10111101 Shuffle to 01111011 Exchange to 01111010
01111010 Shuffle to 11101010 Exchange to 11101011
11101010 Shuffle to 11110100
11101011 Shuffle to 11010111 Exchange to 11010110
Diameter of Shuffle-Exchange Networks

For $2^q$-node shuffle-exchange network: $D = q = \log_2 p$, $d = 4$

With shuffle and exchange links provided separately, as in Fig. 15.18, the diameter increases to $2q - 1$ and node degree reduces to $3$

![Diagram of Eight-node network with separate shuffle and exchange links.](image)

Fig. 15.18 Eight-node network with separate shuffle and exchange links.
Fig. 15.19
Multistage shuffle–exchange network (omega network) is the same as butterfly network.
When $q$ is a power of 2, the $2^q$-node cube-connected cycles network derived from the $q$-cube, by replacing each node with a $q$-node cycle, is a subgraph of the $(q + \log_2 q)$-cube \(\Rightarrow\) CCC is a pruned hypercube

Other pruning strategies are possible, leading to interesting tradeoffs

**Fig. 15.20** Example of a pruned hypercube.

\[
D = \log_2 p + 1
\]

\[
d = \frac{\left(\log_2 p + 1\right)}{2}
\]

\[
B = \frac{p}{4}
\]
Möbius Cubes

Dimension-\(i\) neighbor of \(x = x_{q-1}x_{q-2} \ldots x_{i+1}x_i \ldots x_1 x_0\) is:

\[
x_{q-1}x_{q-2} \ldots 0x_i' \ldots x_1 x_0 \quad \text{if} \quad x_{i+1} = 0 \text{ (}x_i\text{ complemented, as in } q\text{-cube)}
\]

\[
x_{q-1}x_{q-2} \ldots 1x_i' \ldots x_1' x_0' \quad \text{if} \quad x_{i+1} = 1 \text{ (}x_i\text{ and bits to its right complemented)}
\]

For dimension \(q - 1\), since there is no \(x_q\), the neighbor can be defined in two possible ways, leading to 0- and 1-Möbius cubes.

A Möbius cube has a diameter of about 1/2 and an average internode distance of about 2/3 of that of a hypercube.

Fig. 15.21
Two 8-node Möbius cubes.
16 A Sampler of Other Networks

Complete the picture of the “sea of interconnection networks”:
- Examples of composite, hybrid, and multilevel networks
- Notions of network performance and cost-effectiveness

<table>
<thead>
<tr>
<th>Topics in This Chapter</th>
</tr>
</thead>
<tbody>
<tr>
<td>16.1 Performance Parameters for Networks</td>
</tr>
<tr>
<td>16.2 Star and Pancake Networks</td>
</tr>
<tr>
<td>16.3 Ring-Based Networks</td>
</tr>
<tr>
<td>16.4 Composite or Hybrid Networks</td>
</tr>
<tr>
<td>16.5 Hierarchical (Multilevel) Networks</td>
</tr>
<tr>
<td>16.6 Multistage Interconnection Networks</td>
</tr>
</tbody>
</table>
16.1 Performance Parameters for Networks

A wide variety of direct interconnection networks have been proposed for, or used in, parallel computers.

They differ in topological, performance, robustness, and realizability attributes.

Fig. 4.8 (expanded)
The sea of direct interconnection networks.
Diameter and Average Distance

Diameter $D$ (indicator of worst-case message latency)
Routing diameter $D(R)$; based on routing algorithm $R$

Average internode distance $\Delta$ (indicator of average-case latency)
Routing average internode distance $\Delta(R)$

For the $3 \times 3$ mesh:
$\Delta = \frac{(4 \times 18 + 4 \times 15 + 12)}{(9 \times 8)} = 2$
[or $144/81 = 16/9$]

For the $3 \times 3$ torus:
$\Delta = \frac{(4 \times 1 + 4 \times 2)}{8} = 1.5$ [or $12/9 = 4/3$]
Bisection Width

Indicator or random communication capacity

Node bisection and link bisection

Hard to determine; Intuition can be very misleading

Fig. 16.2 A network whose bisection width is not as large as it appears.
Determining the Bisection Width

Establish upper bound by taking a number of trial cuts. Then, try to match the upper bound by a lower bound.

Establishing a lower bound on $B$:
Embed $K_p$ into $p$-node network
Let $c$ be the maximum congestion

$B \geq \lceil p^2/4 \rceil / c$

An embedding of $K_9$ into $3 \times 3$ mesh

Improved, corrected version of this diagram on next slide

Bisection width $= 4 \times 5 = 20$
Example for Bounding the Bisection Width

Embed $K_9$ into $3 \times 3$ mesh

Observe the max congestion of 7

$\lceil p^2/4 \rceil = 20$

Must cut at least 3 bundles to sever 20 paths

Bisection width of a $3 \times 3$ mesh is at least 3

Given the upper bound of 4:

$3 \leq B \leq 4$
Degree-Diameter Relationship

**Age-old question:** What is the best way to interconnect $p$ nodes of degree $d$ to minimize the diameter $D$ of the resulting network?

**Alternatively:** Given a desired diameter $D$ and nodes of degree $d$, what is the max number of nodes $p$ that can be accommodated?

Moore bounds (digraphs)

$p \leq 1 + d + d^2 + \ldots + d^D = \frac{(d^{D+1} - 1)}{(d - 1)}$

$D \geq \log_d [p(d - 1) + 1] - 1$

Only ring and $K_p$ match these bounds

Moore bounds (undirected graphs)

$p \leq 1 + d + d(d - 1) + \ldots + d(d - 1)^{D-1}$

$\quad = 1 + d[(d - 1)^D - 1]/(d - 2)$

$D \geq \log_{d-1}[(p - 1)(d - 2)/d + 1]$  

Only ring with odd size $p$ and a few other networks match these bounds
Moore Graphs

A Moore graph matches the bounds on diameter and number of nodes.

For $d = 2$, we have $p \leq 2D + 1$
Odd-sized ring satisfies this bound

For $d = 3$, we have $p \leq 3 \times 2^D - 2$
$D = 1$ leads to $p \leq 4$ ($K_4$ satisfies the bound)
$D = 2$ leads to $p \leq 10$ and the first nontrivial example (Petersen graph)

Fig. 16.1 The 10-node Petersen graph.
How Good Are Meshes and Hypercubes?

For $d = 4$, we have $D \geq \log_3[(p + 1)/2]$  
So, 2D mesh and torus networks are far from optimal in diameter, whereas butterfly is asymptotically optimal within a constant factor.

For $d = \log_2 p$ (as for $d$-cube), we have $D = \Omega(d / \log d)$  
So the diameter $d$ of a $d$-cube is a factor of $\log d$ over the best possible  
We will see that star graphs match this bound asymptotically

Summary:

For node degree $d$, Moore’s bounds establish the lowest possible diameter $D$ that we can hope to achieve with $p$ nodes, or the largest number $p$ of nodes that we can hope to accommodate for a given $D$.

Coming within a constant factor of the bound is usually good enough; the smaller the constant factor, the better.
Layout Area and Longest Wire

The VLSI layout area required by an interconnection network is intimately related to its bisection width $B$.

If $B$ wires must cross the bisection in 2D layout of a network and wire separation is 1 unit, the smallest dimension of the VLSI chip will be $\geq B$.

The chip area will thus be $\Omega(B^2)$ units.

- $p$-node 2D mesh needs $O(p)$ area.
- $p$-node hypercube needs at least $\Omega(p^2)$ area.

The longest wire required in VLSI layout affects network performance.

For example, any 2D layout of a $p$-node hypercube requires wires of length $\Omega((p/\log p)^{1/2})$; wire length of a mesh does not grow with size.

When wire length grows with size, the per-node performance is bound to degrade for larger systems, thus implying sublinear speedup.
Measures of Network Cost-Effectiveness

Composite measures, that take both the network performance and its implementation cost into account, are useful in comparisons.

One such measure is the degree-diameter product, $dD$

- Mesh / torus: $\Theta(p^{1/2})$
- Binary tree: $\Theta(\log p)$
- Pyramid: $\Theta(\log p)$
- Hypercube: $\Theta(\log^2 p)$

Not quite similar in cost-performance

However, this measure is somewhat misleading, as the node degree $d$ is not an accurate measure of cost; e.g., VLSI layout area also depends on wire lengths and wiring pattern and bus based systems have low node degrees and diameters without necessarily being cost-effective.

Robustness must be taken into account in any practical comparison of interconnection networks (e.g., tree is not as attractive in this regard)
16.2 Star and Pancake Networks

Fig. 16.3 The four-dimensional star graph.

Has $p = q!$ nodes

Each node labeled with a string $x_1x_2...x_q$ which is a permutation of \{1, 2, ... , q\}

Node $x_1x_2...x_i...x_q$ is connected to $x_ix_2...x_1...x_q$ for each $i$ (note that $x_1$ and $x_i$ are interchanged)

When the $i$th symbol is switched with $x_1$, the corresponding link is called a dimension-$i$ link

$d = q - 1; \ D = \left\lfloor 3(q - 1)/2 \right\rfloor$

$D, \ d = O(\log p / \log \log p)$
Routing in the Star Graph

Source node

- Dimension-2 link to
- Dimension-6 link to

Last symbol now adjusted
- Dimension-2 link to
- Dimension-5 link to

Last 2 symbols now adjusted
- Dimension-2 link to
- Dimension-4 link to

Last 3 symbols now adjusted
- Dimension-2 link to
- Dimension-3 link to

Last 4 symbols now adjusted
- Dimension-2 link (Dest’n)

The diameter of star is in fact somewhat less
$D = \lceil 3(q-1)/2 \rceil$

Clearly, this is not a shortest-path routing algorithm.

We need a maximum of two routing steps per symbol, except that last two symbols need at most 1 step for adjustment $\Rightarrow D \leq 2q - 3$
### Star’s Sublogarithmic Degree and Diameter

\[ d = \Theta(q) \text{ and } D = \Theta(q); \text{ but how is } q \text{ related to the number } p \text{ of nodes?} \]

\[ p = q! \approx e^{-q}q^q (2\pi q)^{1/2} \quad [\text{using Striling’s approximation to } q!] \]

\[ \ln p \approx -q + (q + 1/2) \ln q + \ln(2\pi)/2 = \Theta(q \log q) \quad \text{or} \quad q = \Theta(\log p / \log \log p) \]

Hence, node degree and diameter are sublogarithmic

Star graph is asymptotically optimal to within a constant factor with regard to Moore’s diameter lower bound

Routing on star graphs is simple and reasonably efficient; however, virtually all other algorithms are more complex than the corresponding algorithms on hypercubes

<table>
<thead>
<tr>
<th>Network diameter</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>Star nodes</td>
<td>24</td>
<td>--</td>
<td>120</td>
<td>720</td>
<td>--</td>
<td>5040</td>
</tr>
<tr>
<td>Hypercube nodes</td>
<td>16</td>
<td>32</td>
<td>64</td>
<td>128</td>
<td>256</td>
<td>512</td>
</tr>
</tbody>
</table>
The Star-Connected Cycles Network

Replace degree-$(q - 1)$ nodes with $(q - 1)$-cycles

This leads to a scalable version of the star graph whose node degree of 3 does not grow with size

The diameter of SCC is about the same as that of a comparably sized CCC network

However, routing and other algorithms for SCC are more complex

Fig. 16.4 The four-dimensional star-connected cycles network.
Pancake Networks

Similar to star networks in terms of node degree and diameter

Dimension-\(i\) neighbor obtained by “flipping” the first \(i\) symbols; hence, the name “pancake”

We need two flips per symbol in the worst case; \(D \leq 2q - 3\)

Source node

<table>
<thead>
<tr>
<th>Dimension-2 link to</th>
<th>Dimension-6 link to</th>
<th>Last 2 symbols now adjusted</th>
<th>Last 4 symbols now adjusted</th>
<th>Dimension-2 link (Dest’n)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 5 4 3 6 2</td>
<td>5 1 4 3 6 2</td>
<td>2 6 3 4 1 5</td>
<td>4 3 6 2 1 5</td>
<td>3 4 6 2 1 5</td>
</tr>
</tbody>
</table>
Cayley Networks

Group:
A semigroup with an identity element and inverses for all elements.

**Example 1:** Integers with addition or multiplication operator form a group.

**Example 2:** Permutations, with the composition operator, form a group.

Star and pancake networks are instances of Cayley graphs

Cayley graph:
Node labels are from a group $G$, and a subset $S$ of $G$ defines the connectivity via the group operator $\otimes$

Node $x$ is connected to node $y$ iff $x \otimes \gamma = y$ for some $\gamma \in S$

Elements of $S$ are “generators” of $G$ if every element of $G$ can be expressed as a finite product of their powers
Star as a Cayley Network

Four-dimensional star:

Group $G$ of the permutations of $\{1, 2, 3, 4\}$

The generators are the following permutations:

$(1 \ 4) \ (2) \ (3)$

$(1 \ 3) \ (2) \ (4)$

$(1 \ 4) \ (2) \ (3)$

The identity element is:

$(1) \ (2) \ (3) \ (4)$

Fig. 16.3  The four-dimensional star graph.
16.3 Ring-Based Networks

Rings are simple, but have low performance and lack robustness.

Hence, a variety of multilevel and augmented ring networks have been proposed.

Fig. 16.5 A 64-node ring-of-rings architecture composed of eight 8-node local rings and one second-level ring.
Chordal Ring Networks

Routing algorithm: Greedy routing

Given one chord type $s$, the optimal length for $s$ is approximately $p^{1/2}$

Fig. 16.6
Unidirectional ring, two chordal rings, and node connectivity in general.
Chordal Rings Compared to Torus Networks

The ILLIAC IV interconnection scheme, often described as \(8 \times 8\) mesh or torus, was really a 64-node chordal ring with skip distance 8.

Fig. 16.7 Chordal rings redrawn to show their similarity to torus networks.

Perfect Difference Networks

A class of chordal rings, studied at UCSB (two-part paper in *IEEE TPDS*, August 2005) have a diameter of \(D = 2\).

Perfect difference \(\{0, 1, 3\}\): All numbers in the range 1-6 mod 7 can be formed as the difference of two numbers in the set.
Periodically Regular Chordal Rings

Modified greedy routing: first route to the head of a group; then use pure greedy routing
Some Properties of PRC Rings

Fig. 16.9 VLSI layout for a 64-node periodically regular chordal ring.

Remove some skip links for cost-performance tradeoff; similar in nature to CCC network with longer cycles.

Fig. 16.10 A PRC ring redrawn as a butterfly- or ADM-like network.

Dimension 1 \( s_4 = 16 \)
Dimension 1 \( s_3 = 8 \)
Dimension 2 \( s_2 = 4 \)
Dimension 1 \( s_1 = \text{nil} \)

No skip in this dimension
16.4 Composite or Hybrid Networks

Motivation: Combine the connectivity schemes from two (or more) “pure” networks in order to:

- Achieve some advantages from each structure
- Derive network sizes that are otherwise unavailable
- Realize any number of performance/cost benefits

A very large set of combinations have been tried
New combinations are still being discovered
Composition by Cartesian Product Operation

Properties of product graph \( G = G' \times G'' \):

- Nodes labeled \((x', x'')\), \(x' \in V', x'' \in V''\)
- \(p = p'p''\)
- \(d = d' + d''\)
- \(D = D' + D''\)
- \(\Delta = \Delta' + \Delta''\)

Routing: \(G'\)-first
\((x', x'') \rightarrow (y', x'') \rightarrow (y', y'')\)

Broadcasting

Semigroup & parallel prefix computations

Fig. 13.4 Examples of product graphs.
Other Properties and Examples of Product Graphs

If $G'$ and $G''$ are Hamiltonian, then the $p' \times p''$ torus is a subgraph of $G$

For results on connectivity and fault diameter, see [Day00], [AlAy02]

Fig. 16.11 Mesh of trees compared with mesh-connected trees.
16.5 Hierarchical (Multilevel) Networks

We have already seen several examples of hierarchical networks: multilevel buses (Fig. 4.9); CCC; PRC rings

Can be defined from the bottom up or from the top down

Take first-level ring networks and interconnect them as a hypercube

Take a top-level hypercube and replace its nodes with given networks

Fig. 16.13 Hierarchical or multilevel bus network.
Example: Mesh of Meshes Networks

The same idea can be used to form ring of rings, hypercube of hypercubes, complete graph of complete graphs, and more generally, $X$ of $X$s networks.

When network topologies at the two levels are different, we have $X$ of $Y$s networks.

Generalizable to three levels ($X$ of $Y$s of $Z$s networks), four levels, or more.

Fig. 16.12 The mesh of meshes network exhibits greater modularity than a mesh.
Example: Swapped Networks

Build a $p^2$-node network using $p$-node building blocks (nuclei or clusters) by connecting node $i$ in cluster $j$ to node $j$ in cluster $i$.

Also known in the literature as OTIS (optical transpose interconnect system) network.

We can square the network size by adding one link per node.

Two-level swapped network with $2 \times 2$ mesh as its nucleus.
Swapped Networks Are Maximally Fault-Tolerant

For any connected, degree-\(d\) basis network \(G\), \(\text{Swap}(G) = \text{OTIS}(G)\) has the maximal connectivity of \(d\) and can thus tolerate up to \(d - 1\) faults.

One case of several cases in the proof, corresponding to source and destination nodes being in different clusters.

Source: Chen, Xiao, Parhami, IEEE TPDS, March 2009
Example: Biswapped Networks

Build a $2p^2$-node network using $p$-node building blocks (nuclei or clusters) by connecting node $i$ in cluster $j$ of part 0 to node $j$ in cluster $i$ of part 1.
16.6 Multistage Interconnection Networks

Numerous indirect or multistage interconnection networks (MINs) have been proposed for, or used in, parallel computers.

They differ in topological, performance, robustness, and realizability attributes.

We have already seen the butterfly, hierarchical bus, beneš, and ADM networks.

Fig. 4.8 (modified)
The sea of indirect interconnection networks.
Self-Routing Permutation Networks

Do there exist self-routing permutation networks? (The butterfly network is self-routing, but it is not a permutation network)

Permutation routing through a MIN is the same problem as sorting

Fig. 16.14 Example of sorting on a binary radix sort network.
Partial List of Important MINs

**Augmented data manipulator (ADM):** aka unfolded PM2I (Fig. 15.12)
**Banyan:** Any MIN with a unique path between any input and any output (e.g. butterfly)
**Baseline:** Butterfly network with nodes labeled differently
**Beneš:** Back-to-back butterfly networks, sharing one column (Figs. 15.9-10)
**Bidelta:** A MIN that is a delta network in either direction
**Butterfly:** aka unfolded hypercube (Figs. 6.9, 15.4-5)
**Data manipulator:** Same as ADM, but with switches in a column restricted to same state
**Delta:** Any MIN for which the outputs of each switch have distinct labels (say 0 and 1 for $2 \times 2$ switches) and path label, composed of concatenating switch output labels leading from an input to an output depends only on the output
**Flip:** Reverse of the omega network (inputs × outputs)
**Indirect cube:** Same as butterfly or omega
**Omega:** Multi-stage shuffle-exchange network; isomorphic to butterfly (Fig. 15.19)
**Permutation:** Any MIN that can realize all permutations
**Rearrangeable:** Same as permutation network
**Reverse baseline:** Baseline network, with the roles of inputs and outputs interchanged
16.6* Natural and Human-Made Networks

Since multistage networks will move to Part II’ (shared memory), the new version of this subsection will discuss the classes of small-world and scale-free networks.

Example of a collaboration network (e.g., co-authors of papers) showing clusters and inter-cluster connectivity.

Image credit: Univ. Leiden
Professional Connections Networks

Image credit: LinkedIn