Multiuser Allocation on n-cube

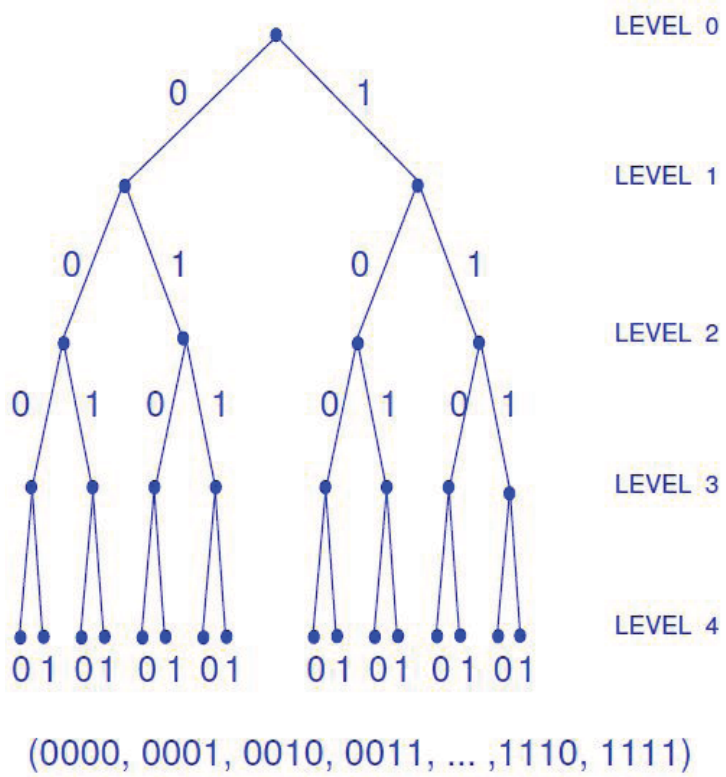
Users dynamically request subcubes which may, or may not be granted by the operating system depending on availability.

A good allocation strategy:

- *maximizes processor utilization, and*
- *minimizes system fragmentation.*

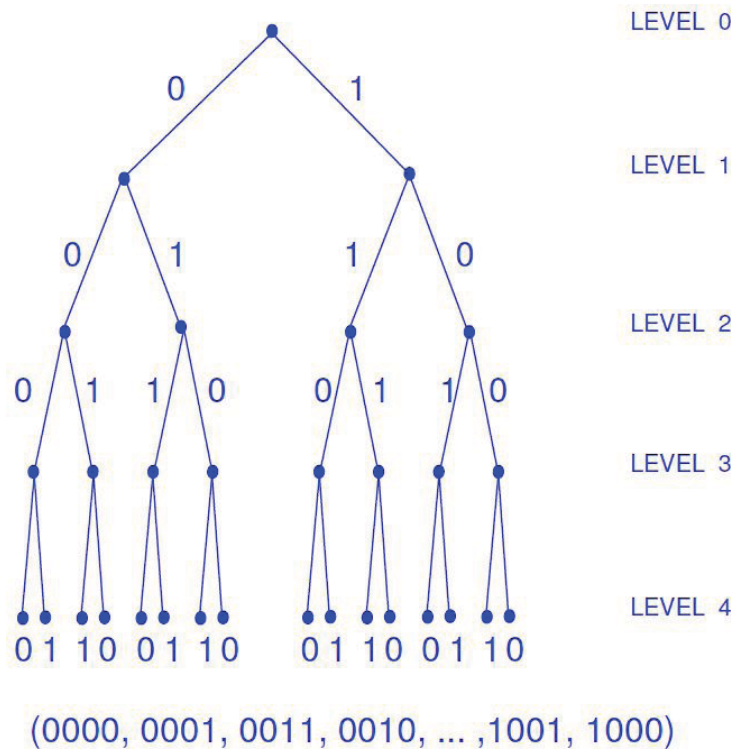
It is very difficult to detect the availability of a subcube of required size and merge released subcubes of small sizes into a larger subcube (exponential time delay).

Buddy Allocation on n-cube



- *Buddy strategy [ChSh87].* When an incoming request needs a k-subcube, the level $n-k$ of the tree is searched from left to right and the first free k-subcube is allocated. At most 2^{n-k} different subcubes are examined.
- *Buddy allocation is statically optimal.* It can accommodate any input request sequence iff the total requested processors is less than $N = 2^n$.

Gray Code Allocation on n-cube



- *Gray code (GC) strategy.* When an incoming request needs a k-subcube, the tree at level $n - k + 1$ is searched circularly, left to right for two adjacent free nodes.
- *GC strategy is statically optimal.* The number of k-subcubes recognizable by GC is 2^{n-k+1}, twice that of the buddy strategy.
- *BRGC strategy can not recognize all free subcubes (e.g.,* $\{0011,\ 0010,\ 1010,\ 1011\}$ *as a 2-subcube).* This is recognizable by $\{2, 1, 3, 4\}$ bit-permuted gray code.
- *GC strategy outperforms the buddy strategy.* It packs requests more densely (larger contiguous regions).

Dynamic Allocation on n-cube

- *Showing the existence of a free subcube of requested dimension.* $C(n,\ n/2)$ gray code sequences (out of $n!$ possible) are sufficient to recognize any subcube.
- *Finding the addresses of the free subcubes.*
- *Choosing a subcube from the set of candidates which results in less fragmentation.*

Dynamic Routing - Saturation Rate

Let m_g the packet generating rate and m the arrival rate at a communication link. We assume d-port full-duplex communication, and constant delay model.

LEMMA 1 *Under any routing, dynamic routing on a vertex symmetric graph with N nodes, degree d, and average distance D_a can be stable only if* $m_g \leq \frac{d\,(N-1)}{D_a\,N}$

PROOF: The no. packets generated per time unit is $N \cdot m_g$. Under any routing scheme, a generated packet has to cover a distance of at least $D_a\ N/(N-1)$ steps (no local packets). Hence, the avg demand is $m_g\ D_a\ N^2/(N-1)$ packet transmissions per time unit. However, only $d\ N$ transmissions per each time unit can be accomodated.

Proposed contention index $m_0 = d\ (N-1)/(D_a\ N)$.

Dynamic Routing - Latency

Routing using *randomly chosen shortest-paths*. Let P_t be the termination probability of a packet.

LEMMA 2 *Given a vertex symmetric graph with N nodes, degree d, and avg distance D_a, and dynamic packet routing using randomly chosen shortest paths,*

1. *The packet termination probability is*

$$P_t = \frac{N-1}{D_a\ N}$$

2. *The arrival rate is*

$$m = \frac{m_g}{d\ P_t}$$

3. *The average time delay is equal to*

$$T = 1 + \frac{\overline{b}}{m\ P_t}$$

where

$$\overline{b} = m + \frac{d - 2 + 2\ P_t - d\ P_t^2}{d-1} \cdot \frac{m^2}{2\ (1-m)}$$

Dynamic Routing - Latency

PROOF: Since on the average a packet makes D_a hops and no self-addressing is permitted, the packet termination probability is $P_t = \frac{N-1}{D_a\, N}$.

By the infinite buffer assumption no packet is lost. Thus, in steady state, at any node, the rate of packets accepted should match the generating rate (m_g). Hence, $m_g = d\, m\, P_t$.

Queuing at each node corresponds to (a) packet arrival rate m from any of d adjacent links, (b) termination probability P_t, and (c) service rate of one unit. This can be approximated by an M/D/1 queueing problem. The exact solution provided is obtained using Markov models.

VCT - Wormhole - Circuit Switching

Fast bit-serial algorithms are difficult to construct on the hypercube, since the head of the packet may be far from its tail, preventing up to $O(n)$ edges to be used.

PROBLEM 4 *Is wormhole routing to independent random destinations possible without deadlock for a long time, and with high probability?*

An n-dimensional hypercube is **circuit-switching rearrangeable** if for any permutation there exist 2^n edge-disjoint paths connecting all (source, destination) pairs.

LEMMA 3 *The n-dimensional hypercube is circuit-switching rearrangeable only if $n \leq 3$ [Szy'89], [Lub'90].*

Meshes

Sorting depends on the ordering scheme.

The lth largest (or smallest) value is placed at the address (x, y), where:

a) in **row major**, $l = xn + y + 1$,

b) in **snake-like row major** either $l = xn + y + 1$ if x is even, or $l = xn + n - y$ if x is odd, and

c) in **blockwise snake-like row major**, the shuffle row major permutation (π_7) is applied to the row major ordering scheme.

Column major ordering schemes defined similarly.

Ordering Schemes for Meshes

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

0	4	8	12
1	5	9	13
2	6	10	14
3	7	11	15

0	1	2	3
7	6	5	4
8	9	10	11
15	14	13	12

0	1	4	5
2	3	6	7
8	9	12	13
10	11	14	15

Indexings of a 4×4 mesh. From left to right: row-major, column-major, snake-like row-major, shuffled row-major indexing (also called blockwise snake-like).

Converting from snake-like row major to row major, and vice-versa takes n steps on the *MIMD* $n \times n$ mesh.

Converting between (any) ordering schemes takes $O(n)$ steps on the *SIMD* or *MIMD* $n \times n$ mesh.

Torus Embeddings onto Mesh

A torus can be embedded onto a mesh of equal size with dilation 2.

0	9	1	8	2	7	3	6	4	5

Dilation 2 embedding of one-dimensional torus of length 10 onto one-dimensional mesh. Embedding of 2-dimensional torus onto a mesh with dilation 2 is realized analogously, by 'folding' in both directions.

- *Any optimal torus algorithm can be simulated on a mesh with delay factor 2 (except for SIMD model).*
- *The opposite is not true.* There are problems that cannot be solved twice as fast on a torus. An example is sorting on one-dimensional arrays.

Mesh Routing/Sorting - *any/RM/SL/SRM* order

SIMD Model	**Lower Bound**	**Upper Bound**
Sorting	$4n-4$ *(any)*	$7n$ *(SL)*
Permutation Routing	$4n-4$	$4n+\mathcal{O}(n/q)$
Off-line Permutation		$3n-3$, q=2
BPC		$4n-4$, q=$o(1)$
MIMD Model	**Lower Bound**	**Upper Bound**
Sorting*, 1 packet/node	$\Omega(3n)$ *(RM,SL)*	$3n+O(\sqrt[4]{n^3})$ *(SL)*
Sorting*, 1 packet/node	$\Omega(2.27n)$ *(any)*	
Sorting	$\Omega(2n)$ *(RM,SL,SRM)*	$2n+o(n)$ *(SL)*
Prob. Sorting		$2n+o(1)$ *(SRM)*
Permutation Routing	$2n-2$	$2n-2$, $q=32$
Permutation Routing		$2n+o(1)$, $q=2$
Prob. Permutation	$2n-2$	$2n-2$, q=$o(1)$
Off-line Permutation		$3n-3$, q=1
Off-line Permutation		$2.5n$, q=2
Off-line Permutation		$2.25n$, q=4
Omega/Inv.Omega		$2n-2$, q=1
VLSI Model	**Lower Bound**	**Upper Bound**
Sorting ($m \times n$ mesh)	$5(m+n)$, *(any)*	$16(m+n)$ *(RM,SL)*
VCT Mode	**Lower Bound**	**Upper Bound**
Permutation	$\Omega(bn/2)$, b=flit size	$bn/2+o(bn)$

VLSI Sorting Model

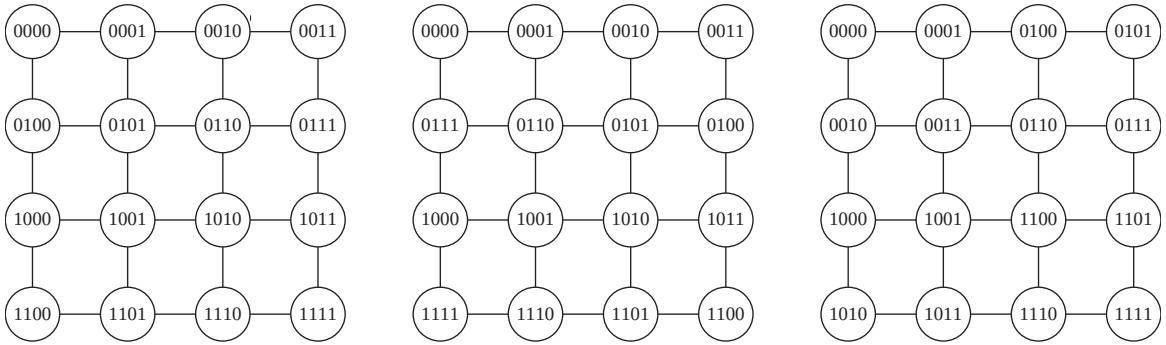
VLSI sorting model consists of a repeated sequence of [row sort + permutation routing + column sort]. The permutation is usually a shift, shuffle, or transpose.

Results for h relation, sorting, permutation sequences.

PROBLEM 7 *a) Closing gaps between lower and upper bounds, b) providing practical algorithms ($T \leq cn$, with small c), and c) finding optimal ordering(s) for sorting problems.*

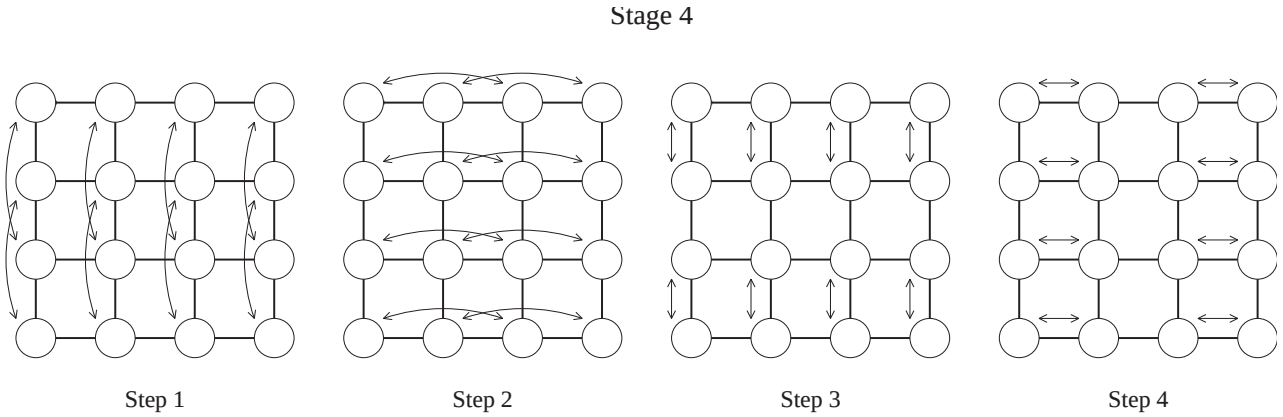
Bitonic Sorting onto Mesh

- *Most frequent compare & swaps must correspond to mesh neighbors.*
- *Embedding bitonic sorting network onto a* 4×4 *mesh using: (a) row-major, (b) row-major snakelike, and (c) shuffled row-major ordering.*



Bitonic Sorting onto Mesh - SRM

- *Concentrate on* **shuffled row-major** *ordering.* Links corresponding to the i^{th} least-significant bit are mapped onto mesh processors $2^{\lfloor (i-1)/2 \rfloor}$ steps away.
- *Example showing compare & swaps for the last stage of bitonic sorting on a* 4×4 *mesh.*



Bitonic Sorting onto Mesh - SRM

- *For $p = N = n^2$, the sorting time on $n \times n$ mesh is:*

$$T(N) = \frac{(1 + logN)logN}{2} + \sum_{j=1}^{logN} \sum_{i=1}^{j} 2^{\lfloor (i-1)/2 \rfloor}$$

This reduces to: $T(N) = O(0.5log^2N + 7\sqrt{N}) = \Theta(\sqrt{N})$ which matches the lower bound, but it is not cost-optimal.

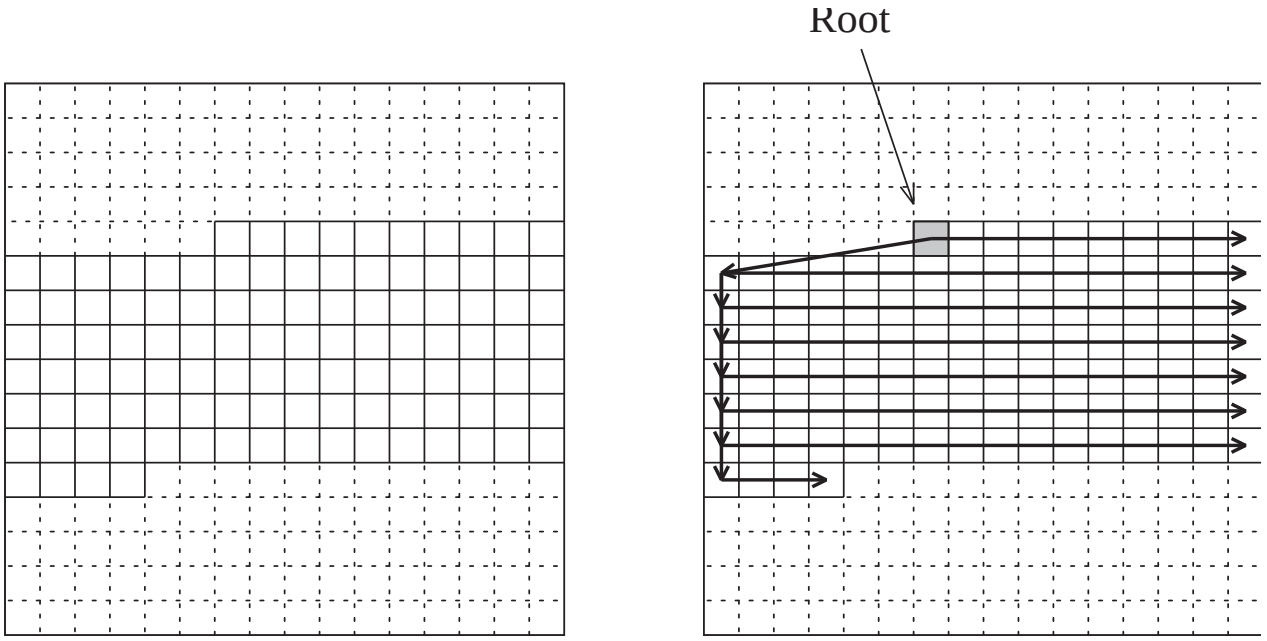
- *For $p > N = n^2$, the sorting time on $n \times n$ mesh is:*

$$T(N) = (p/N) \cdot log(p/N) + (p/N) \cdot log^2N + p/\sqrt{N}$$

The algorithm is cost-optimal if $N = \Theta(log^2p)$. The isoefficiency function is $\Theta(2^{\sqrt{N}} \cdot \sqrt{N})$.

Quicksort on Mesh

Binary tree (b) embedded onto a submesh (a) used for pivot broadcast, gathering, and scattering.



Quicksort on Mesh - RM

a. Concentrate on **row-major** *ordering.*

b. Pivot element (shaded) selected at random.

c. Broadcast pivot downwards.

d. Determine # of smaller & larger elements.

e. Gather # of smaller & larger elements upwards.

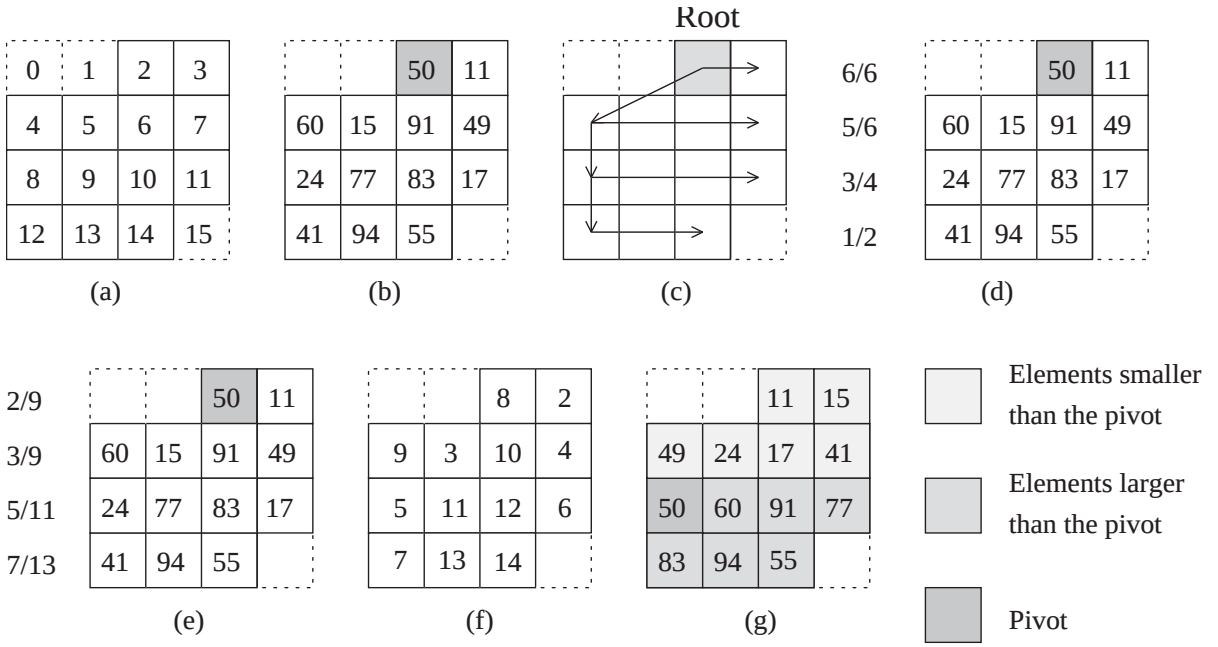
f. Scatter routing information downwards.

g. Permutation routing.

For $p = N = n^2$*, the sorting time on* $n \times n$ *mesh is:*
$T(N) = O(\sqrt{N}\ logN)$. Cost-optimal for large p.

Example: Quicksort on Mesh - RM

Partitioning 13 elements on 4×4 *mesh.*



Randomized Routing on Mesh

Assume packet Ψ heads from node $(0,0)$ to node (X,Y).

For W, a random node in $\{0,\ldots,n-1\}$,

- Phase 1: **Transmit Ψ to** $(W,0)$
- Phase 2: **Transmit Ψ to** (W,Y)
- Phase 3: **Transmit Ψ to** (X,Y)

LEMMA 5 *The above algorithm runs on the MIMD $n \times n$ mesh in $(3n + can^{\delta})$ steps with high probability $(\geq 1 - N^{-a})$, where $\delta > 1/2$ and c, a are fixed constants [Va'82].*

Deterministic Routing on Mesh

$2n - 2$ steps and constant queues [LeMaTo'89, RaTs'92]:

1. Divide the mesh into row slices of size εn.
2. Each processor chooses W randomly within its slice.
3. Overlap the three phases of randomized routing. Superior packets are distanced at most $2\varepsilon n$ from any corner.
4. Employ queue priorities:
   a) Phase I packets have priority over Phase III packets,
   b) superior Phase II packets have priority over remaining packets, otherwise priority in Phase II is according to farthest origin (within that Phase),
   c) farthest destination first is used in Phase III.

Dynamic Routing on Mesh

THEOREM 24 *If the packet arrival rate in the MIMD $n \times n$ mesh is less than the network capacity $\lambda < 4/n$, then in a window of T steps, any packet is delayed at most $(2n + o(\log T + \log n))$ steps with queues of size $O(1 + \log T/\log n)$ with high probability [Le'90].*

Similarly, for routing N packets to random destinations,

THEOREM 25 *$N = n^2$ packets initially located at distinct nodes of the MIMD $n \times n$ mesh can be routed to independently chosen random destinations with the basic greedy routing algorithm in $(2n + o(\log n))$ steps with constant queues with high probability.*

Wormhole Routing on Mesh

For wormhole routing congestion analysis is hard; packets which have partially crossed an edge can block other packets for an arbitrary time. Good performance for delayed wormhole routing: random initial delay before greedy routing.

PROBLEM 8 *Is wormhole routing to independently chosen, random destinations possible without deadlock for a long time and with high probability?*

Mesh-Like Networks

Many routing/sorting results for mesh-like networks;
[GrHsKr'94];

- ***D*-dimensional Mesh**
- ***D*-dimensional Torus**
- ***D*-dimensional Diagonal Mesh**
- **Circular/Rhomboid Mesh**
- **Coated Mesh**

LEMMA 9 *On a reconfigurable $n \times n$ mesh, n numbers can be sorted in a constant number of steps [WaChLi'90, BePeRaSc'91].*

Using column sort less than 40 steps are needed!

Hypercubic and Related Networks

A hypercube algorithm is **normal** if at any step, all active communication links associate with a single dimension, with consecutive dimensions used in consecutive steps.

THEOREM 12 *Hypercubic networks (butterfly, cube-connected, shuffle-exchange) simulate any normal hypercube algorithm with constant slowdown [Lei'92].*

Many routing/sorting results exist [GraHsuKra'94];

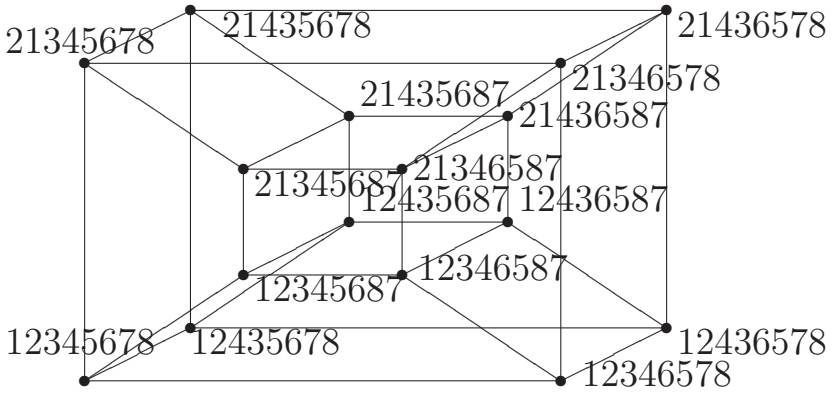
- *Butterfly Network*
- *Cube-Connected Cycles*
- *Shuffle-Exchange*
- *De Bruijn Graphs*
- *Generalized Hypercubes/Hypermeshes*

Cayley Graphs and Others

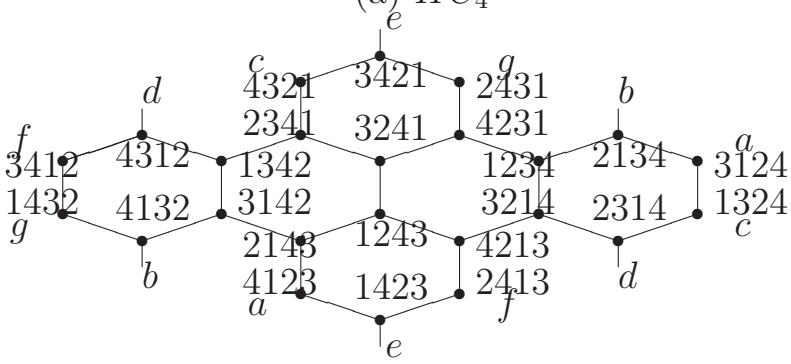
Some routing/sorting results exist for interesting graphs.

- **Pancake Graph**
- **Alternating Group Graph**
- **Bubblesort Graph**
- **Kautz Graph**
- **Fat Trees**

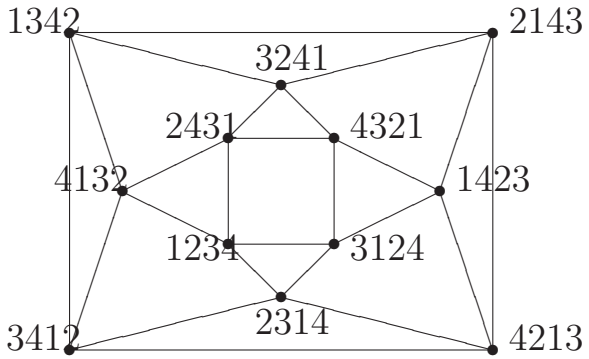
Examples of Cayley graphs HC_4, ST_4 and AG_4



(a) HC_4



(b) ST_4



(c) AG_4

Graph	Size	Degree	Diameter	Symmetry
HC_n	2^n	n	n	yes
ST_n	$n!$	$n-1$	$\lfloor \frac{3(n-1)}{2} \rfloor$	yes
AG_n	$\frac{n!}{2}$	$2(n-2)$	$\lfloor \frac{3(n-1)}{2} \rfloor$	yes

Topological and symmetry properties

Graph	N	m_0	Graph	N	m_0	Graph	N	m_0
HC_{10}	1024	2.00	ST_6	720	1.04	AG_6	360	2.22
HC_{11}	2048	2.00	ST_7	5040	1.02	AG_7	2520	2.13
HC_{12}	4096	2.00	ST_8	40320	1.01	AG_8	20160	2.08

Maximum generating rate m_0 for stable dynamic routing

HC_{11}			ST_6			AG_6		
m_g	T	Simulation	m_g	T	Simulation	m_g	T	Simulation
0.2	6.78	6.77	0.2	6.22	6.21	0.2	4.78	4.79
0.4	7.12	7.11	0.4	6.91	6.94	0.4	4.97	4.99
0.6	7.56	7.54	0.6	8.22	8.29	0.6	5.21	5.25
0.8	8.15	8.09	0.8	11.68	11.13	0.8	5.51	5.63
1.0	8.98	8.85	1.0	46.79	44.81	1.0	5.92	6.16
HC_{12}			ST_7			AG_7		
m_g	T	Simulation	m_g	T	Simulation	m_g	T	Simulation
0.2	7.30	7.29	0.2	7.45	7.50	0.2	5.88	5.88
0.4	7.68	7.66	0.4	8.40	8.43	0.4	6.14	6.16
0.6	8.17	8.12	0.6	10.24	10.39	0.6	6.47	6.53
0.8	8.82	8.74	0.8	15.41	14.90	0.8	6.90	7.06
1.0	9.73	9.56	1.0	121.87	119.24	1.0	7.48	7.87

Average packet latency for dynamic routing

Star Graph Routing/Sorting

This graph has $N = n!$ nodes, degree $n - 1$, and diameter $\lfloor 3(n-1)/2 \rfloor$.

MIMD Model	**Lower Bound**	**Up. Bound**
Sorting	$\Omega(n \log n)$	$O(n^3 \log n)$
Perm. Routing (greedy)	$\Omega(\sqrt{N}/n)$	$O(\sqrt{N})$, q=$O(\sqrt{N})$
Perm. Routing (adaptive)		$O(n^3)$
Isotonic Routing		$O(n^2)$
Prob. Permutation		cn
Off-line Permutation		$4n - 7$, q=1
Pipelined MIMD	**Lower Bound**	**Upper Bound**
Isotonic (n-packets)		$O(n^2)$

PROBLEM 9 *a) Closing gaps between lower, upper bounds, b) providing practical algorithms for small graphs, c) results for virtual cut-through, wormhole, or circuit-switching mode.*

Star Graph Routing/Sorting

CLAIM 2 *a) Probabilistic permutation routing can be performed in $O(n)$ steps using constant queues, b)* BPC*, Omega, and Inverse Omega permutations can be routed in near-optimal time.*

Sorting is based on shearsort [MeSo'90], or bitonic sort [RaWe'93].

The algorithm for packings first uses prefix computation, for determining the correct destination of each packet. Then, it performs greedy routing.

Adaptive routing repeats n identical phases, consisting of greedy routing and packing, essentially re-distributing one packet per each star node.