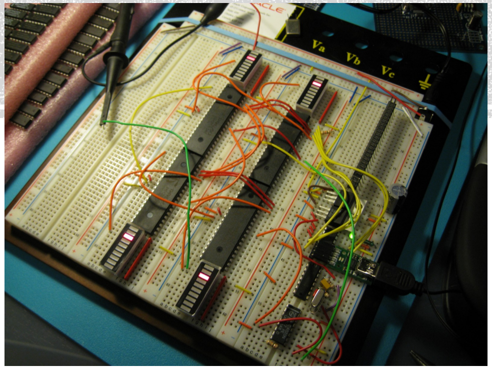
PACKET-ROUTING ALGORITHMS ON INTERCONNECTION TOPOLOGIES



PACKET-ROUTING ON INTERCONNECTION TOPOLOGIES (1)

- Up to now, in the routing problem we have considered the network as a graph unknown to the nodes and variable in time (faults, varying traffic, etc.)
- Nevertheless, when the network is an interconnection topology (and connects, for example, processors), it is known and fixed in time. Furthermore, efficiency is a primary issue.
- Solutions having stronger properties than the simple shortest path algs are required.

PACKET-ROUTING ON INTERCONNECTION TOPOLOGIES (2)

Many different types of routing models.

Here, we will focus on the store-and-forward model (also known as the packet-switching model):

- Each packet is maintained as an entity that is passed from node to node as it moves through the network
- A single packet can cross each edge during each step of the routing
- Depending on the algorithm, we may or may not allow packets to pile up in queues located at each node. When queues are allowed: effort to keep them short.

PACKET-ROUTING ON INTERCONNECTION TOPOLOGIES (3)

- Global controller to precompute routing paths not allowed
- Problem handled using only local control
- A routing problem is called one-to-one if at most one packet must be addressed to every node and each packet has a different destination.
- In contrast, one-to-many and many-to-one

BUTTERFLY NETWORK (1)

Def. Let $N=2^n$ (hence $n=log\ N$); the *n*-dimensional Butterfly is a layered graph with:

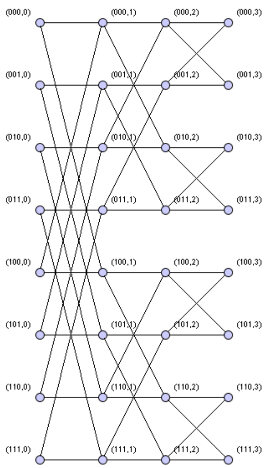
- $N(n+1)$ nodes ($n+1$ layers with 2^n nodes each) and
- $2Nn$ edges.

Nodes:

nodes correspond to pairs $(w,\ i)$, where:

- i is the *layer* of the node
- w is an n-bit *binary number* that denotes the *row* of the node.

…



BUTTERFLY NETWORK (2)

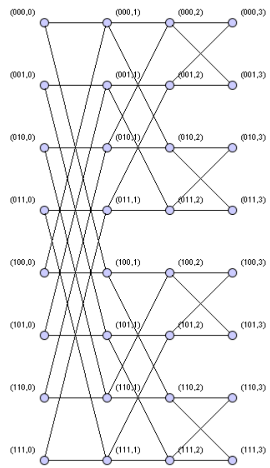
def. of n-dimensional butterfly (cntd)

…

Edges:

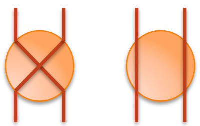
Two nodes (w, i) e (w', i') are linked by an edge iff $i'=i+1$ and either:

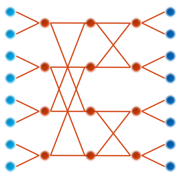
- $w=w'$ (*straight edge*) or
- w and w' differ in precisely the i-th bit (*cross edge*)



BUTTERFLY NETWORK (3)

- The nodes of the Butterfly are *crossbar switches,* i.e. switches with two input and two output values and can assume two states, *cross* and *bar*.
- Hence, the butterfly can be seen as a switching network connecting $2N$ ($N=2^n$) input units to $2N$ output units trough a $logN+1$ layered network, having N nodes each.
- Input and output devices are usually processors and are often omitted in the graphical representations for the sake of simplicity.

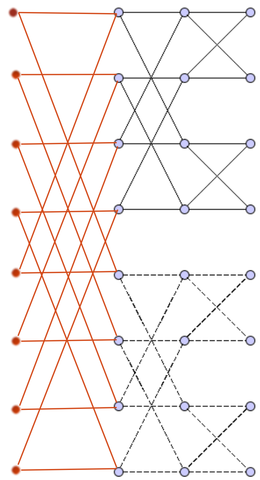




BUTTERFLY NETWORK (4)

The butterfly has a simple recursive structure:

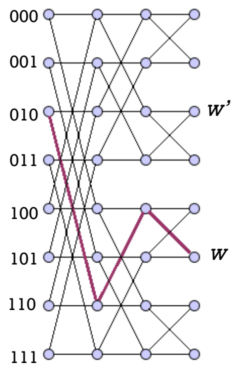
one n-dim. butterfly contains two (n-1)-dim. butterflies as subgraphs (just remove either the layer 0 nodes or the layer n nodes of the n-dim. butterfly to get two $(n\text{-}1)$-dimensional butterflies).



BUTTERFLY NETWORK (5)

For each pair of rows w and w', there exists a unique path of length n (known as greedy path) from $(w,0)$ to (w', n);

this path passes through each layer exactly once, using a cross-edge from layer i to layer $i+1$ ($i=0,\ldots,n$) iff w and w' differ in the i-th bit and using a straight-edge otherwise.



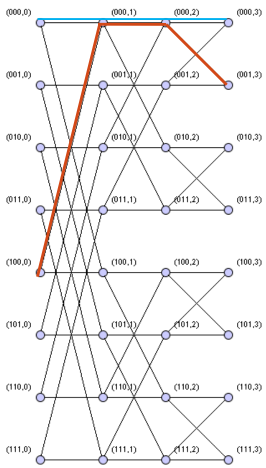
ROUTING ON THE BUTTERFLY (1)

Problem of routing N packets from layer 0 to layer n in an n-dimensional butterfly:

- Each node $(u,0)$ on layer 0 of the butterfly contains a packet that is destined for node $(\pi(u), n)$ on layer n, where $\pi:[1, N] \rightarrow [1,N]$ is a permutation.
- In the greedy routing algorithm, each packet is constrained to follow its greedy path.
- When there is only one packet to route, the greedy algorithm performs very well.
- Trouble can arise when many packets have to be routed in parallel…

ROUTING ON THE BUTTERFLY (2)

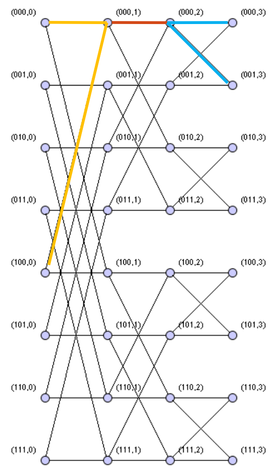
- Many greedy paths might pass through a single node or edge.
- Since only one of these packets can use the edge at a time, one of them must be delayed before crossing the edge.
- The butterfly is not able to route each permutation without delays, i.e. is a blocking network
- The congestion problem arising in this example is not overly serious. When N is larger, however, the problem can be much serious. In fact…



ROUTING ON THE BUTTERFLY (3)

- Assume for simplicity n odd (but similar results hold when n is even), and consider edge

  $e=((00\ldots0, (n-1)/2), (00\ldots0,(n+1)/2))$
- Node $(00\ldots0, (n-1)/2)$ is the root of a complete binary tree extending to the left having $2^{(n-1)/2}$ leaves
- Analogously to the right
- …



ROUTING ON THE BUTTERFLY (4)

- …
- The permutation can be such that <u>each greedy path from a leaf of the left tree arrives to a leaf of the right tree traversing e</u>
- There are $2^{(n-1)/2}=\sqrt{N/2}$ possible such paths, and thus $2^{(n-1)/2}=\sqrt{N/2}$ packets must traverse e. So at least one of them will be delayed by $\sqrt{N/2}-1$ steps.
- It takes at least $n=\log N$ steps to traverse the whole networks and to route a packet to its destination.
- In this case, the greedy algorithm can take $\sqrt{N/2}+\log N-1$ steps to route a permutation.
- In general…

ROUTING ON THE BUTTERFLY (5)

Th. *Given any routing problem on an n-dimensional butterfly for which at most one packet starts at each layer 0 node and at most one packet is destined for each layer n node, the greedy algorithm will route all the packets to their destinations in $O(\sqrt{N})$ steps.*

Proof. For simplicity, assume n odd (but the case n even is similar)

- Let e be any edge in layer i, $0<i\leq n$, and define n_i to be the number of greedy paths that traverse e
- $n_i \leq 2^{i-1}$ (left tree) and, similarly, $n_i \leq 2^{n-i}$ (right tree)
- Any packet crossing e can only be delayed by the other n_i-1 packets that want to cross the edge.
- …

ROUTING ON THE BUTTERFLY (6)

- …
- As this packet traverses layers 1, 2, …, n, the total delay encountered can be at most:

$$\sum_{i=1}^{n}(n_i-1) = \sum_{i=1}^{(n+1)/2}(n_i-1) + \sum_{i=(n+3)/2}^{n}(n_i-1) \leq \sum_{i=1}^{(n+1)/2}(2^{i-1}-1) + \sum_{i=(n+3)/2}^{n}(2^{n-i}-1) \leq$$

=(n+1)/2+1

$$= \sum_{j=0}^{(n+1)/2-1}(2^j-1)$$

$$= \sum_{j=0}^{(n-3)/2}(2^j-1)$$

$$\leq 2^{(n+1)/2} + 2^{(n-1)/2} - n = O(\sqrt{N}) - n = O(\sqrt{N}) \quad \blacksquare$$

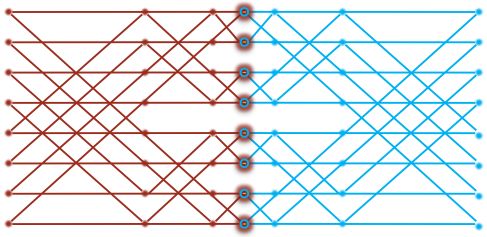
recalling that $\sum_{j=0}^{k} 2^j = 2^{k+1}-1$

ROUTING ON THE BUTTERFLY (7)

- Despite the fact that the greedy routing algorithm performs poorly in the worst case, <u>the greedy algorithm is very useful in practice</u>.
- For many useful classes of permutations, the greedy algorithm runs in n steps, which is optimal and, for most permutations, the greedy algorithm runs in $n + o(n)$ steps.
- As a consequence, the greedy algorithm is widely used in practice.

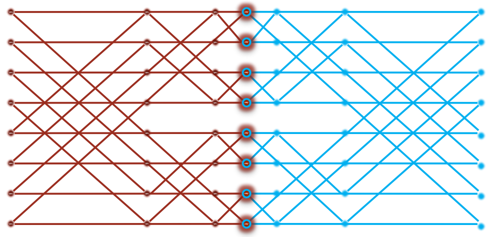
BENEŠ NETWORK (1)

- A possibility to avoid a routing with delays is providing a non blocking topology.
- Beneš network has this property
- It consists of two back-to-back butterflies



BENEŠ NETWORK (2)

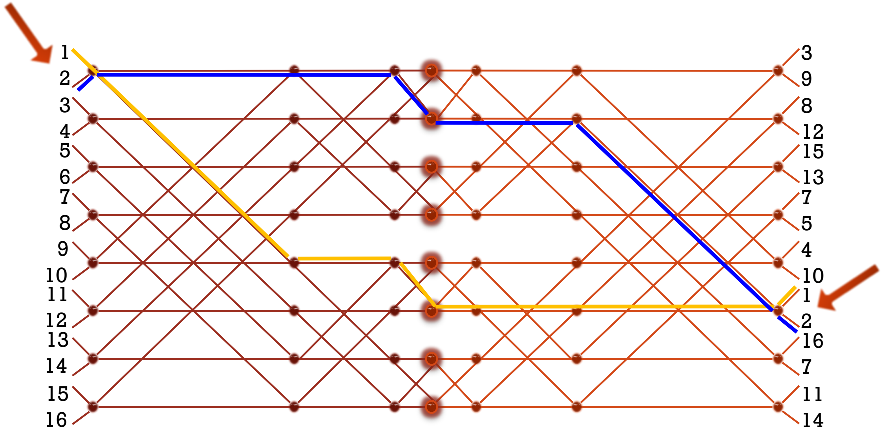
- The n-dimensional Beneš network has $2n+1$ layers, each with 2^n nodes.
- The first and last $n+1$ layers in the network form an n-dimensional Butterfly (the middle layer is shared).
- Not surprisingly, the Beneš network is very similar to the Butterfly, in terms of both its computational power and its network structure.



BENEŠ NETWORK (3)

- The reason for defining the Beneš network is that it is an excellent example of a rearrangeable network.
- **Def.** A network with N inputs and N outputs is said to be rearrangeable if for any one-to-one mapping π of the inputs to the outputs (i.e. for any permutation), we can construct edge-disjoint paths in the network linking the i-th input to the $\pi(i)$-*th* output for $1 \leq i \leq N$.
- In the case of the n-dimensional Beneš network, we can have *two* inputs for each layer 0 node and *two* outputs for every layer $2n$ node, and still connect every permutation of inputs to outputs with edge-disjoint paths.
- Hence, in this case, # of inputs$=2^{n+1}$

BENEŠ NETWORK (4)



BENEŠ NETWORK (5)

It seems extraordinary that we can find edge-disjoint paths for <u>any</u> permutation. Nevertheless, the result is true, and it is even fairly easy to prove, as we show in the following:

Th. *Given any one-to-one mapping π of 2^{n+1} inputs to 2^{n+1} outputs on an r-dimensional Beneš network, there is a set of edge-disjoint paths from the inputs to the outputs connecting input i to output $\pi(i)$ for $1 \leq i \leq 2^{n+1}$.*

Proof. …

BENEŠ NETWORK (6)

PROOF OF THE REARRANGEABILITY OF THE BENEŠ NETWORK (CNTD)

Proof. By induction on *n.*

- <u>Basis</u>: if $n=1$, the Beneš network consists of a single node (i.e. a single 2x2 switch) and the result is obvious.
- <u>Induction</u>: assume that the result is true for an $(n-1)$-dimensional Beneš network
- Key observation: the middle $2n-1$ layers of an n-dimensional Beneš network comprise two $(n-1)$-dimensional Beneš networks
- …

BENEŠ NETWORK (7)

PROOF OF THE REARRANGEABILITY OF THE BENEŠ NETWORK (CNTD)