ADMINISTRIVIA

Units: this is a 2-unit class

One-sentence proposals due Wednesday

Lecture on Friday!!! (but none next Wednesday)

Multiprocessor Topologies

Meshes and Tori

Butterflies

Fat Trees

Multibutterflies

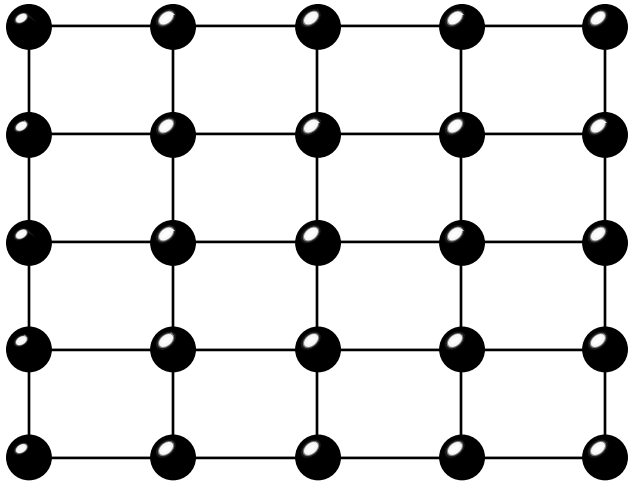
The Multiplexed Metabutterfly

Topology Evaluation

Metrics:

- Long vs. short wires
- Complexity (easy to build, manage?)
- Latency: how many hops?
  - Switches: > 50 ns
  - Wire: 15 cm/ns $\Rightarrow$ one switch $\cong 7.5$ m of wire
- Bisection bandwidth
- Fault tolerance (depends on routing algorithms)
- Load balancing
- The impact/requirement of locality
- Easy to partition into multiple networks?

2D Mesh



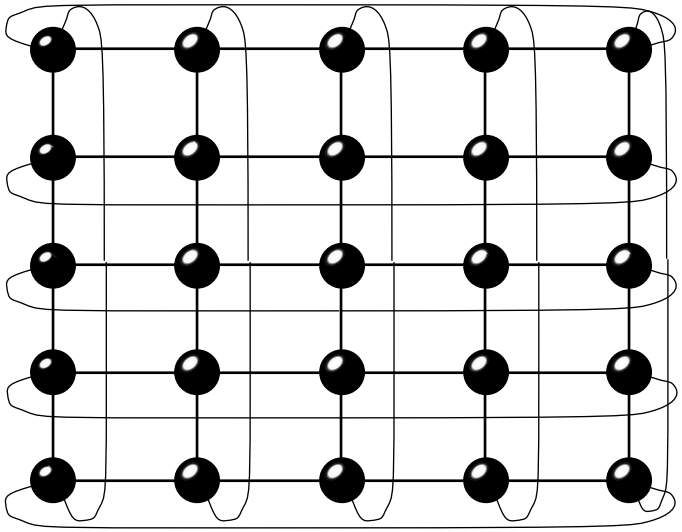
Analysis:

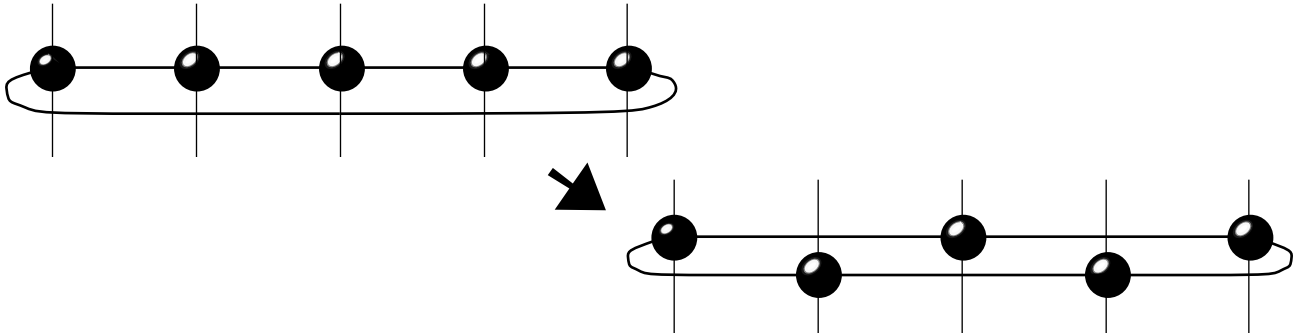
- Short wires
- Easy to build
- $2\sqrt{n}$ hops
- $\sqrt{n}$ bisection bandwidth
- Difficult to do fault tolerance
- Poor load balancing
- Requires locality for good performance
- Easy to partition

2D Torus

Analysis:

- Short wires (think donut)
- Still easy to build
- About half as many hops
- Twice the bisection bandwidth
- Slightly better fault tolerance
- Even load balancing
- Requires locality
- Hard to partition!





K-ary *N*-cubes

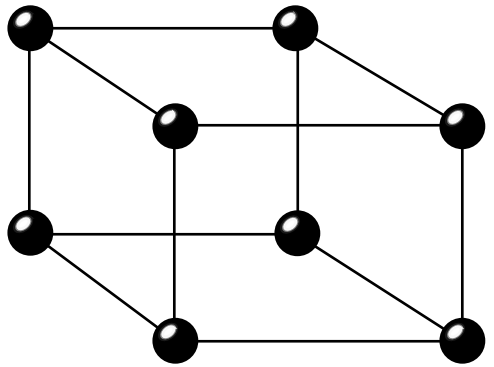
Generalization of 2D mesh/torus:

- n dimensions
- k nodes along each dimension
- k^n total nodes
- $2n$ connections per node (except for mesh edges)

Mesh/Torus: k-ary 2-cubes $\Rightarrow$ square

Hypercubes: 2-ary n-cubes

- $O(n)$ bisection bandwidth
- $Log\ n$ hops
- Long wires for $n \geq 4$



How Many Dimensions?

$n = 2$ or 3:

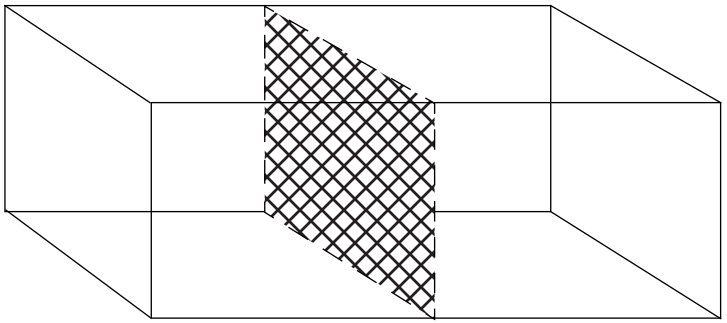
- Short wires, easy to build
- Many hops, low bisection bandwidth
- Requires traffic locality

$n \geq 4$:

- Requires long wires (?)
- Harder to build, more wires, longer average length
- Fewer hops, better bisection bandwidth
- Can handle non-local traffic

The 3D Limitation

For n nodes, bisection area is $O(n^{2/3})$:



For large n, bisection bandwidth is limited to $O(n^{2/3})$:

- Bill Dally, IEEE TPDS, [Dal90a]
- For fixed bisection bandwidth, low-dimensional k-ary n-cubes are better (otherwise higher is better)
- I.e., a few short fat wires are better than many long thin wires
- What about many long fat wires?

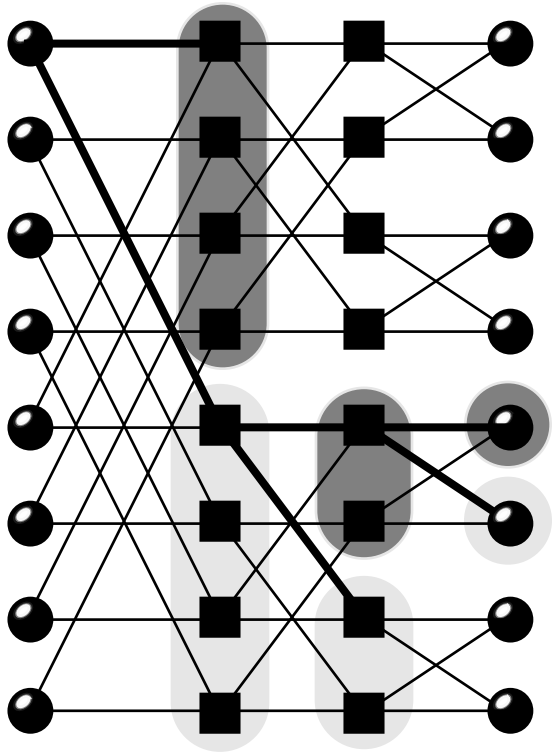
BUTTERFLIES

Indirect networks:

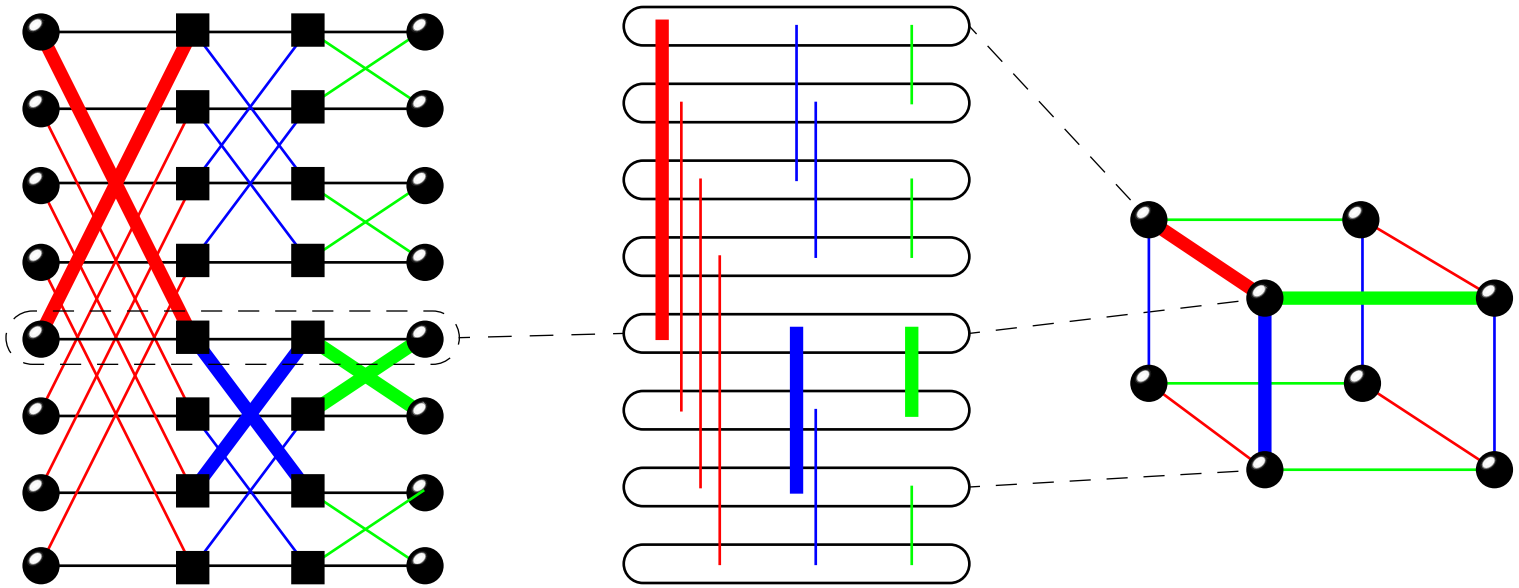
- Travel through routers
- Routers are much simpler than nodes

Butterflies:

- Split into *up* & *down* at each stage
- $Log\ n$ hops
- Bisection bandwidth: $O(n)$
- Some long wires
- No fault tolerance
- No locality
- Easy to partition



Relation to Hypercubes



Wiring is isomorphic!

$\Rightarrow$ low-dimensional tori are superior to butterflies

Except butterfly *always* takes $\log n$ hops

Butterflies vs. 3D Torus

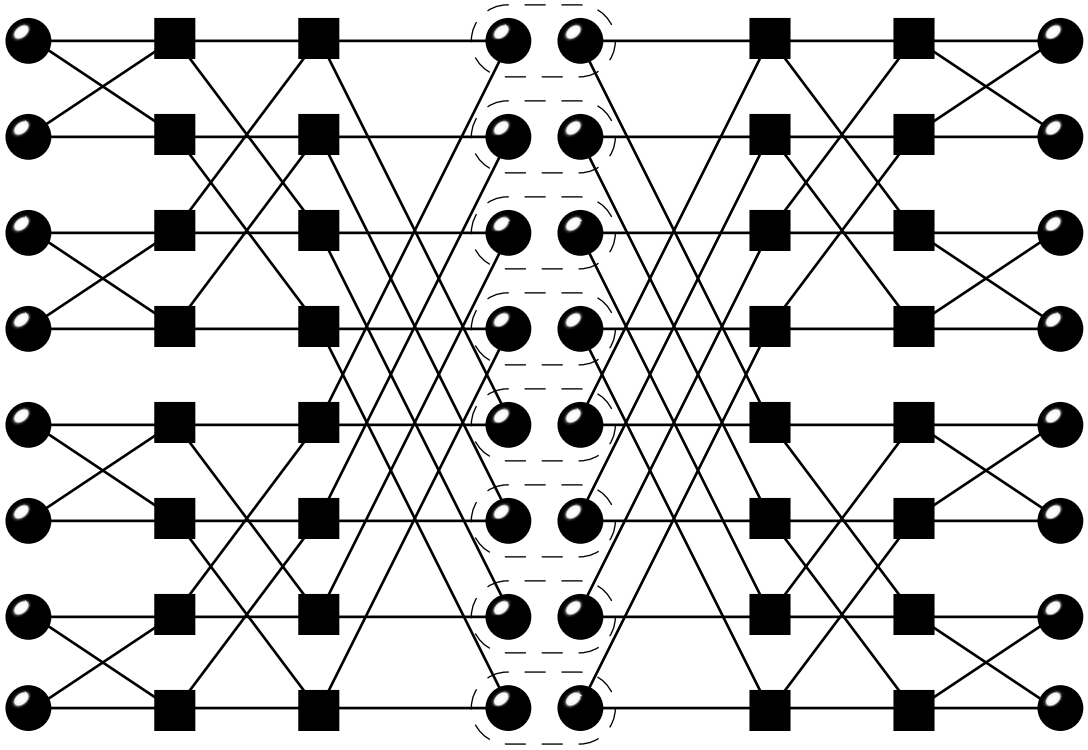
Butterfly has:

- More parts: $O(n \log n)$ routers
- More edges: $2n \log n$ unidirectional vs. $3n$ bidirectional
- Worse fault tolerance
- Lower latency unless significant locality
- Much better bisection bandwidth for fixed cable size
- *Fixed* bisection bandwidth implies worse performance
- Both have significant hot-spot problems
- Simple partitioning into sub-machines

Strictly more expensive, but potentially better performance

BENES NETWORK (ben'-ess)

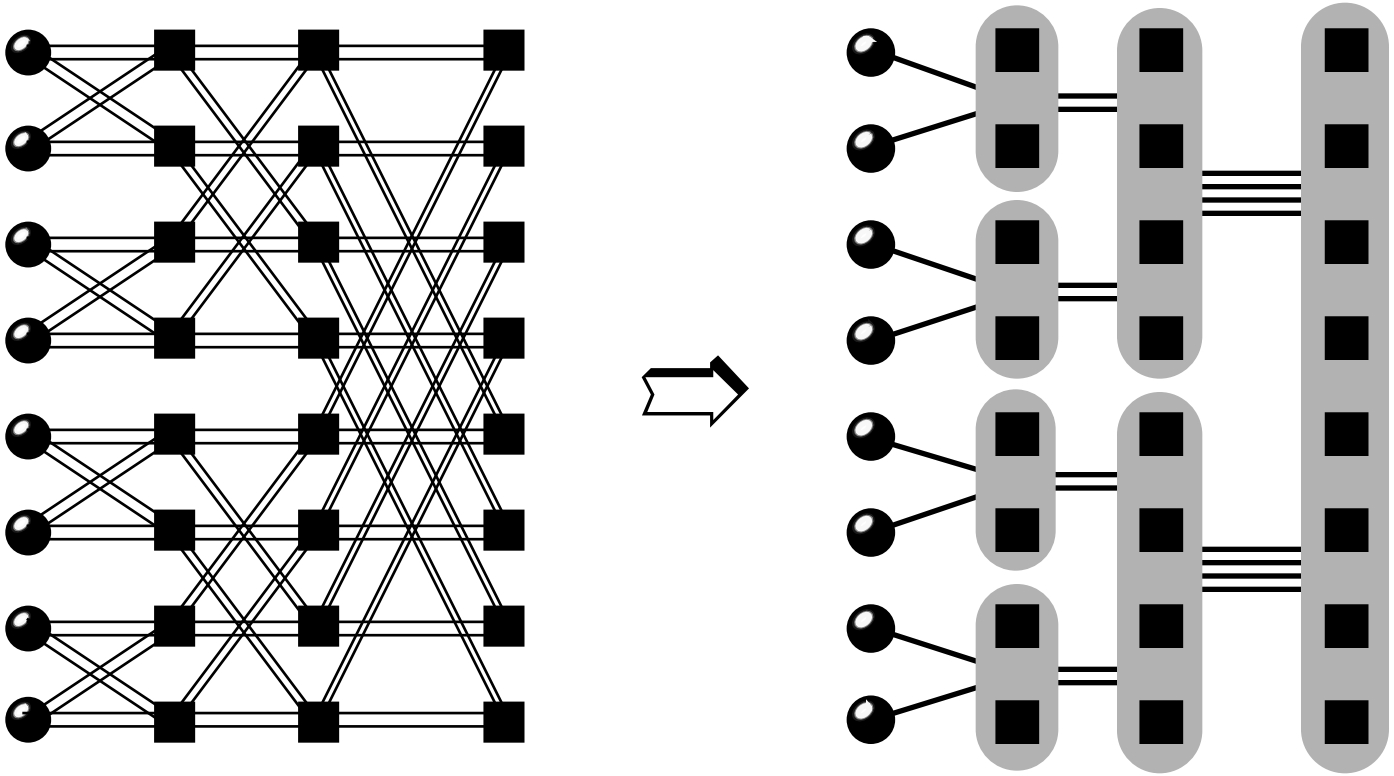
Back-to-back butterflies, first one reversed:



First half randomizes, second half routes

Fat Trees

Folded Benes network:



Except packets only go to least common ancestor

Fat Trees

Analysis:

- Long wires, but bidirectional
- Relatively easy to build, wires routed together
- Hops: $2\log n$ worst case, much better for local traffic
- $O(n)$ bisection bandwidth, *easily adjusted*
- Good fault tolerance
- Excellent load balancing
- Easy to partition into sub-machines

Random routing on the way up:

- Ensures load balancing
- Provides fault tolerance (what about routing down?)

Strictly superior to butterfly; 3D torus?

MULTIBUTTERFLIES

Randomly wired splitter network

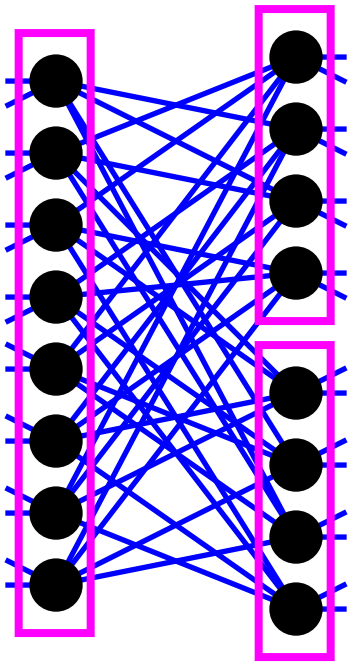
Multiple wires in each direction:

- Number is the *multiplicity*

Key property: *expansion*

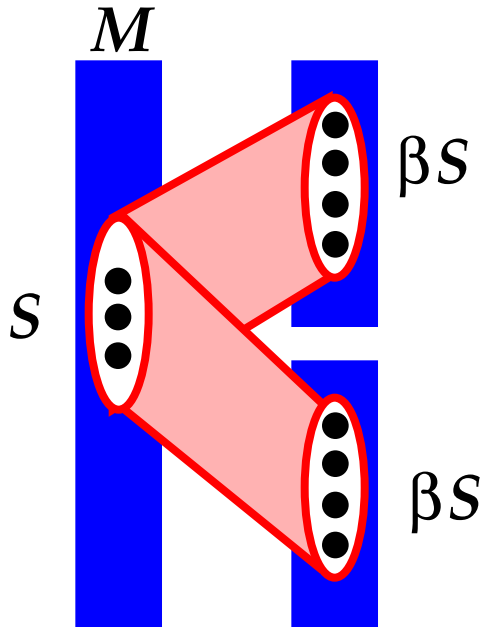
Tremendous performance

Tremendous fault tolerance



Radix: $r = 2$
Multiplicity: $m = 2$
Degree = $rm = 4$

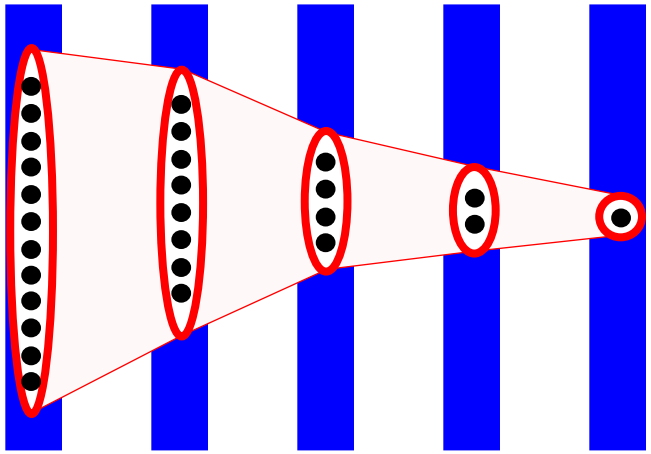
EXPANSION



A splitter has *expansion* if every set of $S < \alpha M$ inputs reaches at least βS outputs in each of the r directions, where $\beta > 1$ and $\alpha < \frac{1}{\beta r}$

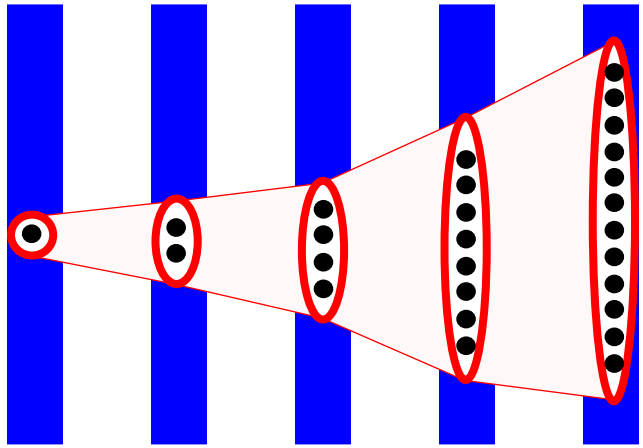
Why Expansion Helps

Butterfly



One fault blocks 2^h nodes h stages back.

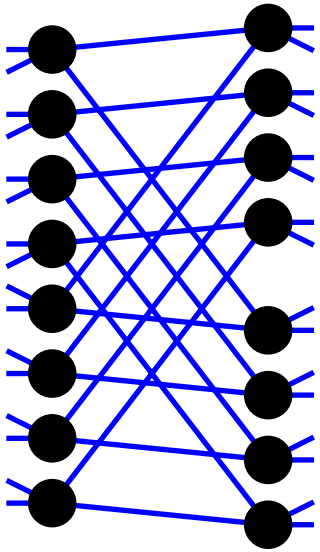
Multibutterfly



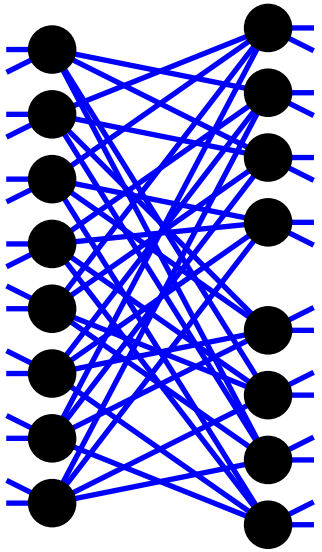
Expansion requires β^h faults to block *one* node h stages back.

For 1024 inputs, multibutterfly with 100 faults outperforms fault-free dilated butterfly [LM92]

Butterfly vs. Multibutterfly



Butterfly



Multibutterfly

	Butterfly	Multibutterfly
Cost	Low	Double
Complexity	Simple	Impractical
Performance	Heavy congestion	Excellent
Fault tolerance	None	Excellent

METABUTTERFLY

Goal: a small number cables per board

- Fixed degree for any size network

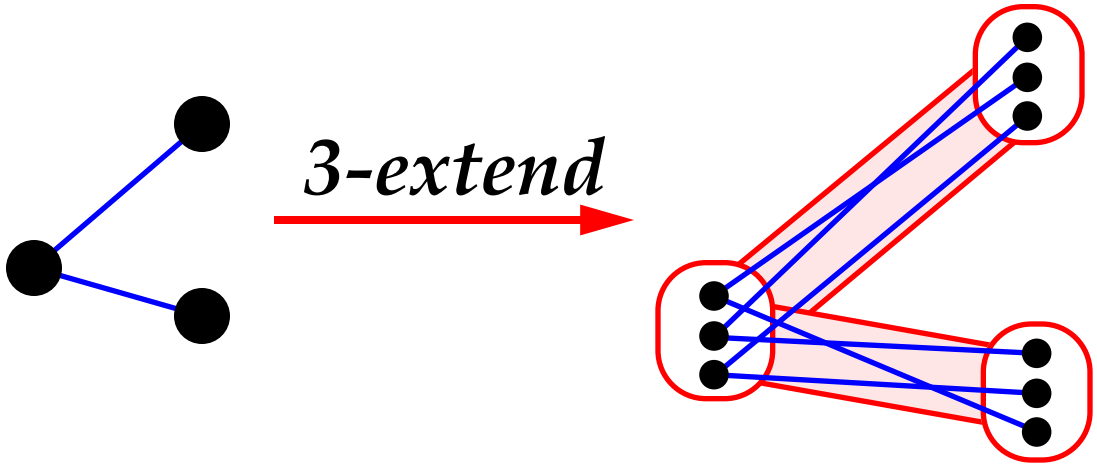
Key idea: ***coerce the wires into cables***

How to get expansion?

- Random cabling
- Random wiring within a cable

As good as fully random wiring? ***Yes***

RANDOM K-EXTENSIONS



Replace:

- each node with “metanode” of k nodes
- each edge with “channel” of k edges

Channels are wired with a random permutation

Metabutterfly is just a k-extended multibutterfly

k-extension preserves degree and expansion [BCL94]

METABUTTERFLIES VS. MULTIBUTTERFLIES

Same hardware – twice as many wires as a butterfly

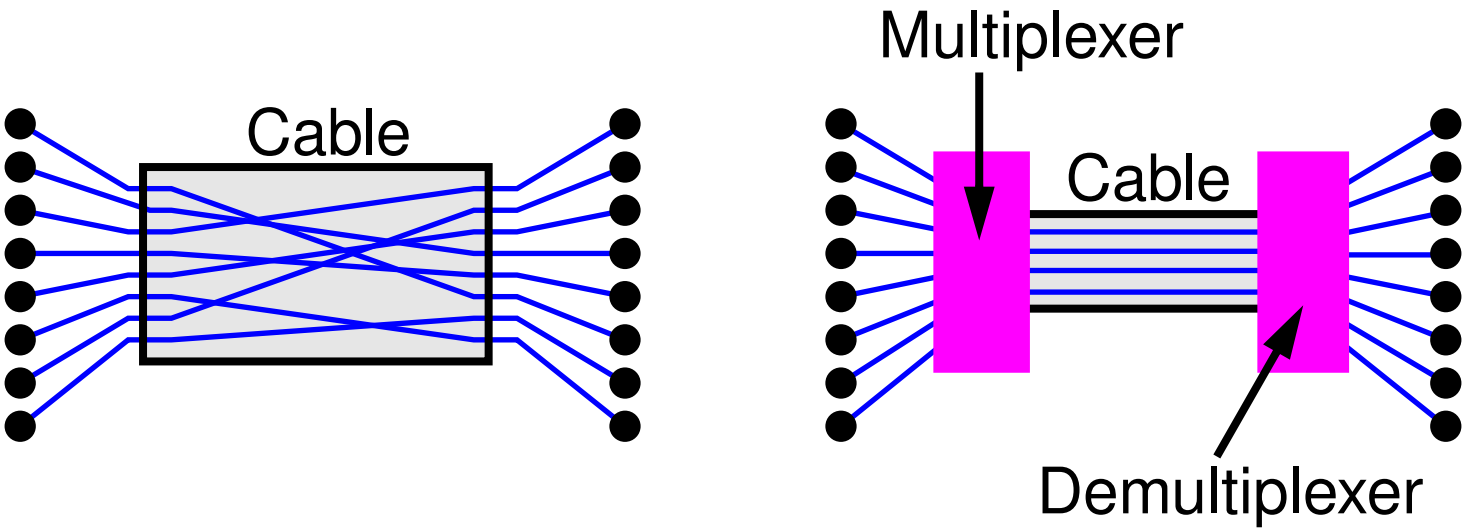
Same performance [CBLK94]

Metabutterflies are buildable:

- Small number of cables
- Cables follow the pattern of a *small* multibutterfly
- Wires within cables are randomly connected

Metabutterflies allow multiplexing

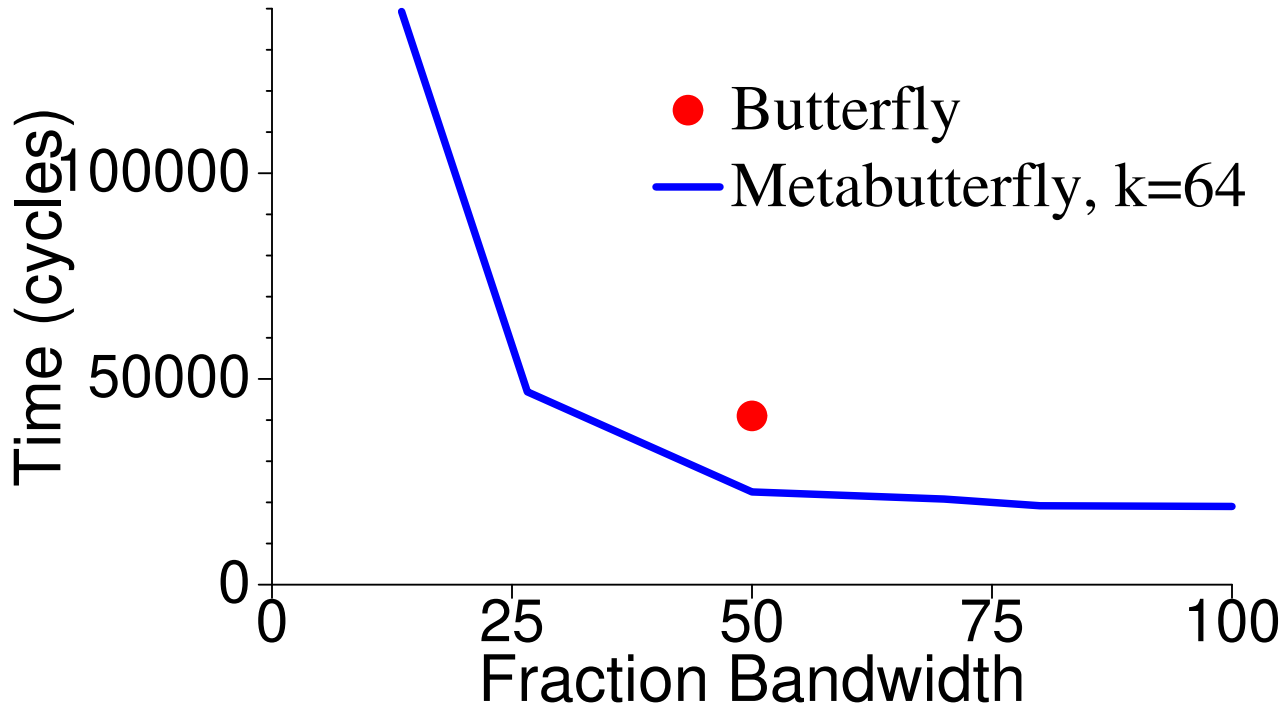
MULTIPLEXING



Cables enable multiplexing:

- Not possible with multibutterfly
- *Same logical edges*, but fewer wires
- Randomness inside multiplexer $\Rightarrow$ standard cables
- Multiplexing/demultiplexing inside routers

Multiplexing Results



Can remove half the wires!

- Twice as fast as the butterfly, despite similar hardware
- Even better with faults or congestion, often > 5x

METABUTTERFLY VS. BUTTERFLY

Multiplexed Metabutterfly:

- 2-5 times better performance
- Great fault tolerance
- Even load balancing
- Similar hardware:
  - Same number of routers
  - Twice as many cables, each half as thick
  - No multiplexing at end stages (which are on one board)
- Easy to build
- Easy to increase bandwidth (just add wires as needed)

SUMMARY

3D mesh/torus

- Torus has lower latency, better bisection bandwidth
- Mesh can be partitioned
- Short wires only, requires locality
- Poor bisection bandwidth (so over-engineer?)

Fat trees

- Precise control over bandwidth
- Exploits locality and load balances
- Best for partitioning

Multiplexed metabutterflies

- Strictly better than the rest of the butterflies
- Best fault/congestion tolerance
- No locality but great load balancing