"Ideal" parallel computer (Slides 2-3)
Problem: # wires = \(\Theta(N^2)\) bad
degree = \(\Theta(N)\) bad
diameter = \(\Theta(1)\) good

Implement as low-degree network (Slides 4-8)
\(N \times N\) mesh of trees:
# switches = \(\Theta(N^2)\) bad
degree = \(\Theta(1)\) good
diameter = \(\Theta(lg N)\) good

Direct network: every node is a processor
Indirect network: processors + switches
(inputs/outputs)

Routing on \(N \times N\) MOT
\(N\) messages at row roots
Route to column roots
- Assume perm, since otherwise hotspot could make any network look bad.
Time = \(\Theta(lg N)\) — but lots of hardware.

Hypercube (Slides 9-10)
Routing: flip any bit that's wrong by routing on that dimension.

\[
\begin{array}{cccccccc}
1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\
1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\
\end{array}
\]

Bitwise XOR of current msg location and dest.

\(\begin{array}{cccccccc}
1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{array}\)
But, msgs may collide.
Also, degree = \(lg N\).
Cube-connected cycles (Slide 11)

- $N = n \log n$ nodes
- degree $= \Theta(1)$
- diameter $= \Theta(\log N)$

Butterfly (FFT) network (Slides 12-13)

- $n$ inputs, $n$ outputs $\quad \langle \text{direct vs. indirect} \rangle$
- $N = n \log n$ nodes
- $\Theta(1)$ degree
- Diameter $= \Theta(\log N)$ $\langle \text{little tricky if not } 2 \text{ or } 0 \rangle$

Isomorphic to CCC, but authors didn't realize.

Decomposing a butterfly (Slides 14-24)

- Remove major cycles $\Rightarrow 2 \cdot n/2$-input butterflies
- Remove minor cycles $\Rightarrow$ .....

Routing on butterfly (Slide 25)

- Just like hypercube, but uses a specific order of dimensions
- $\{ \text{dest} = 0 \Rightarrow \text{go up} \}$ or $\{ \text{xor} = 0 \Rightarrow \text{straight} \}$
- $\{ \text{dest} = 1 \Rightarrow \text{go down} \}$ or $\{ \text{xor} = 1 \Rightarrow \text{cross} \}$

Tree embeddings in butterfly (Slides 26-27)

- CBT rooted at each input
- CBT " " " output
Packet routing on butterfly

source \( y_{d-1} y_{d-2} \ldots x_0 \) \( \rightarrow \) \( y_{d-1} y_{d-2} \ldots y_0 \)

Route major to minor:

\( x_{d-1} x_{d-2} \ldots x_0 \)
\( y_{d-1} x_{d-2} \ldots x_0 \)
\( y_{d-1} y_{d-2} \ldots x_0 \)
\( y_{d-1} y_{d-2} \ldots y_0 \)

\( d \leq \lg n \) steps, but might have congestion!

\( n \) packets on \( n \)-input butterfly

What is worst-case perm?
- \( \sqrt{n} \) packets at sources \( x_1 x_2 x_3 x_4 \), go to dests \( 0000 x_1 x_2 x_3 x_4 \)
All go through line \( 00000000 \) halfway through network \( \Rightarrow \) congestion = \( \sqrt{n} \).

Beneš network (Slides 28–29)

Thm. Any \( n \)-perm can be routed (off-line) on an \( n \)-input Beneš with node-disjoint paths.

PF. Induction on \( n \).
Base \((N=2)\): \( \circ \circ \frac{\times}{\circ} \frac{\times}{\circ} \)

Inductive case (Slides 31–39) \( \Box \)

Corollary: A \( d \)-in, \( d \)-out Beneš network can simulate any \( n \)-node, degree-\( d \) network in \( O(\log (dn)) \) time. \( \Box \)

Bounded-degree \( \Rightarrow O(\log n) \) time

Beneš network is \( O(\log n) \)-universal for offline simulation of bounded-degree networks.
<< Analogy to universal Turing machines >>