6.895 Theory of Parallel Systems Lecture 17

Butterfly Network Analysis and The Beneš Network

Lecturer: Charles Leiserson

Lecture Summary

1. *Network with N Nodes*
   This section proves part of the lower bound on expected routing time for an arbitrary N node network.
2. *The Beneš Network*
   This section motivates, introduces, and analyzes the Beneš connection network.
3. *Routing on Butterfly Networks*
   This section establishes that most routing problems take only $O(\lg N)$ time on butterfly networks.

1 Network with N Nodes

In this section, we prove a lemma that completes the proof of the ***Network with N Nodes*** theorem given in lecture 16.

Theorem 1 *For an arbitrary network with N nodes, the expected routing time to simultaneously send a single message from each processor to a random processor is*

$$E[\textit{Routing Time}] = \Omega(N/BW + \textit{diameter})$$

where BW is the minimum bisection width.

Proof

Lemma 2 *The expected routing time for the N random messages problem laid out in theorem 1 is $\Omega(N/BW)$.*

Proof Refer to the notes for lecture 16 for proof. □

Lemma 3 *Each node v in a N node network chooses a destination $dest(v)$ uniformly at random. Then, $E[\textit{Routing Time}] = \Omega(d)$, where d is the diameter.*

Proof Find node x such that $Pr\{\delta(x; dest(x)) \geq d/2\} \geq 1/2$ where $\delta(x, y)$ is the shortest distance from node x to node y. Then,

$$\begin{aligned}
E[\text{time to route } x \text{ to its destination } dest(x)] &= \sum_k k Pr\{\delta(x; dest(x)) = k\} \\
&\geq \sum_{k \geq d/2} k Pr\{\delta(x; dest(x)) = k\} \\
&\geq d/2 \sum_{k \geq d/2} Pr\{\delta(x; dest(x)) = k\} \\
&= d/2 Pr\{\delta(x, dest(x)) \geq d/2\} \\
&= d/2 \\
&= \Omega(d)
\end{aligned}$$

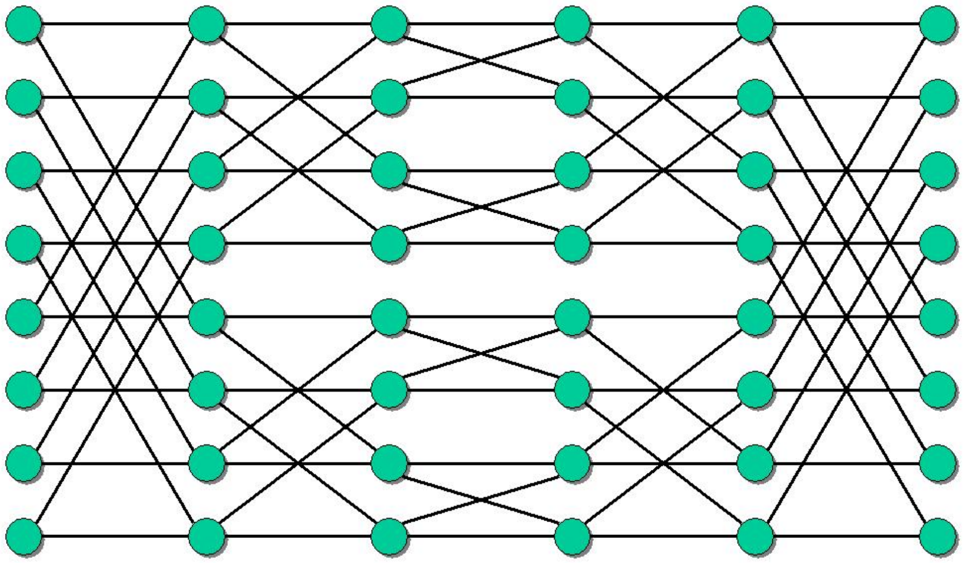

Figure 1: Example of a Beneš Network. Nodes aligned vertically are in the same level and nodes aligned horizontally are in the same row.

To find such an x, let $\delta(x, x') = d$, and define $S = \{v : \delta(x, v) < d/2\}$ and $S' = \{v : \delta(x', v) < d/2\}$. We know $S \cap S' = \varnothing$ since for all $z \in S \cap S'$, $\delta(x, x') \leq \delta(x, z) + \delta(z, x') < d/2 + d/2 < d$ gives a contradiction, implying no such z can exist. Therefore, at least one of $\overline{S}$ and $\overline{S'}$ has $\geq N/2$ nodes. Without loss of generality, assume that $\left|\overline{S}\right| \geq N/2$. Thus, $dest(x) \in \overline{S}$ with probability $\geq 1/2$, i.e., $Pr\{\delta(x, dest(x)) \geq d/2\} \geq 1/2$. $\square$

From lemma 2 and lemma 3, we know that $E[\text{Routing Time}] = \Omega(N/BW)$ and $E[\text{Routing Time}] = \Omega(diameter)$. Therefore, we can conclude that $E[\text{Routing Time}] = \Omega(N/BW + diameter)$. $\square$

2 The Beneš Network

2.1 Worst Case Problems for the Butterfly Network

We saw last lecture that the worst case congestion for the butterfly network is $\sqrt{n}$ for an n input network. This was motivated by considering the problem where inputs $x_1x_2x_3x_4 0000 \longrightarrow$ outputs $0000 x_1x_2x_3x_4$. There are many such bad inputs that cause $\sqrt{n}$ congestion for the butterfly network. For example, consider the problem of computing the matrix transpose where each node represents an element of a matrix and we wish to compute $(i, j) \longrightarrow (j, i)$.

Matrix transpose and other operations that elicit the worst case congestion in a butterfly network are common in programs written for supercomputers. It is therefore a real problem to build supercomputers with butterfly networks because typical applications suffer serious performance penalties.

3 The Beneš Network

This section introduces the Beneš network, a network that allows provably congestion-free communication from inputs to a permutation of the outputs. A Beneš network is constructed by overlapping the low-order

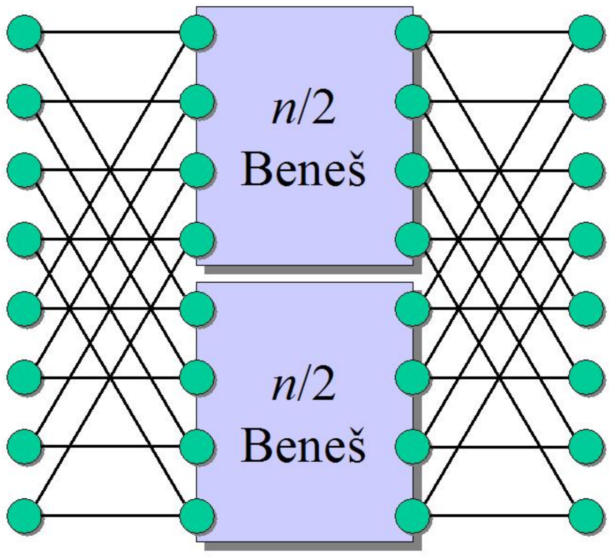

Figure 2: An example Beneš Network with the middle $d-2$ levels replaced with two half-sized Beneš networks.

cycles of two butterfly networks. An example is shown in figure 1.

Theorem 4 *Any n-permutation can be routed (off-line) on an n-input Beneš network with node-disjoint paths.*

Proof The proof uses the fact that Beneš networks are decomposable into smaller Beneš networks. Let $n = 2^d$. The proof is based on induction on d. For $d = 1$, the network is trivial and the theorem is obvious on examination.

As the inductive step, assume that the theorem holds for a $n = 2^d$ input Beneš network. Observe that the middle $d-2$ levels of the Beneš network form two Beneš networks, one on top and one on bottom, each with $n/2 = 2^{d-1}$ inputs, as shown in figure 2. By assumption, any input to one of the sub-Beneš networks can be routed to any output of the same subnetwork without congestion. Therefore, all we need to do is determine whether each input is to be routed through the top or the bottom subnetwork and show that the subnetwork outputs can be correctly routed to the final outputs without congestion.

The only constraint we must observe to avoid congestion is that connected pairs at level 1 and at level d do not overlap. That is, each butterfly at the first and last level must either route packets straight or crossed: butterflies cannot send both input packets to a single output node. Therefore, by observation, the two inputs comprising a butterfly at the first level must be routed into different subnetworks and the two inputs comprising a butterfly at the last level must come from different subnetworks. It is this final constraint that is important.

Define a butterfly pair as

$$bfly(i) = \begin{cases} i + n/2 & (i \leq n/2), \\ i - n/2 & (i > n/2). \end{cases}$$

The butterfly pair is the other node (either input or output) that defines a butterfly in the network, $bfly(i) = j$ and $bfly(j) = i$. Define $\pi(i)$ as the permutation that maps inputs to outputs: $\pi(i) = j$ denotes that input i should be routed to output j and $\pi^{-1}(j) = i$ denotes that output j should be routed from input i.

We now show how to route inputs into the upper and lower subnetworks in a way that satisfies our constraints. We start by routing the first input through the upper subnetwork and connecting it to the correct output, $\pi(1)$. Next, we fulfill the constraint at the last level by routing the first path's output butterfly pair, $\mathit{bfly}(\pi(1))$, through the lower network and back to its correct input, $\pi^{-1}(\mathit{bfly}(\pi(1)))$. We satisfy the new constraint at the input by routing the appropriate input, $\pi^{-1}(\mathit{bfly}(\pi(1)))$, through the upper subnetwork and to its correct output. We continue going back and forth through the upper subnetwork when connecting an input to an output and through the lower subnetwork when connecting an output to an input until we have defined a completed loop. Since the first path went through the upper subnetwork and the final return path went through the lower subnetwork, the input constraint on the original path is satisfied. If the first cycle of paths doesn't include all nodes, we pick an unrouted node and repeat this process until all inputs are routed. Following this algorithm, half of the inputs are routed through the top subnetwork and half of the inputs are routed through the bottom subnetwork without congestion.

We have satisfied the inductive hypothesis by showing that given a $n/2 = 2^{d-1}$ input congestion free Beneš network, we can construct a $n = 2^d$ input congestion free Beneš network. By induction, we have proved the theorem. □

As a side note, since we always have a choice of how to route the first input, we can remove a switch from each inductive level of the butterfly network without affecting its performance characteristics. Consequently, for a $n = 2^d$ input network, we can remove d switches from the network without loss of generality.

Corollary 5 *An n-input Beneš network can simulate any n-node, degree-d network in $O(d \lg n)$ time.*

Proof Let G be an n-node network with maximum degree d. Let H be an n-input Beneš network. We prove the corollary by showing how to simulate G on H. Identify the ith node of G with the ith input of H to simulate node computation. The tricky part is showing how to simulate communication on edges in G using H's network. We do this by identifying $d+1$ subsets of the edges in G such that for each subset, we can route packets from input i to output j for each edge (i, j) in the subset of G with at most $O(\lg n)$ congestion. Each subset is routed in a separate round. $d+1$ subsets with $O(\lg n)$ congestion each gives us a total of $O(d \lg n)$ time to simulate G.

We now construct $d+1$ disjoint and spanning subsets of edges in G. Construct a bipartite graph $\Gamma_G = (U, V, E)$ where $U = \{u_1, u_2, ...u_n\}$, $V = \{v_1, v_2, ...v_n\}$, and $E = \{(u_i, v_j) | (i, j) \text{ is an edge of } G\}$. Communication from i to j in G is represented by edge (u_i, v_j) in Γ_G. Since the maximum degree in G is d, the maximum degree in Γ_G is also d, making Γ_G a d-regular bipartite graph. Therefore, we can construct an edge-coloring of Γ_G using $d+1$ colors (see Leighton's *Introduction to Parallel Algorithms and Architectures* for proof). Each set of edges with a given color forms one of the $d+1$ subsets.

Let S be the kth of the $d+1$ subsets. For each edge (i, j) in S, we simulate communication on (i, j) by sending a packet from input i to output j on H during round k. Since the edges incident to i and the edges incident to j are all colored differently, we know that no two packets originate from or are delivered to the same node during the same round. Therefore, S can be routed with our $O(\lg n)$ bound. One round for each of the $d+1$ subsets with $O(\lg n)$ routing time each completes the proof. □

4 Routing on Butterfly Networks

Even though the butterfly network has a routing time of $\Omega(\sqrt{n})$ on certain permutations (including many "interesting" permutations), most routing problems take only $O(\lg n)$ time. The following theorem covers arbitrary N-packet routing problems on butterfly networks, not just those that route inputs to outputs.

Theorem 6 *Consider the N^N N-packet routing problems on an N-node (n-input, where $n = \Theta(N/\lg N)$) butterfly network. At least $N^N(1 - 1/N^{\Omega(1)})$ of these problems can be routed in $O(\lg N)$ time.*

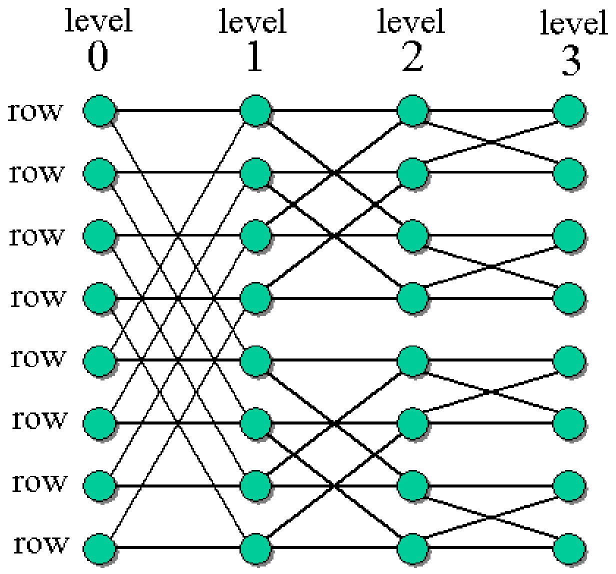

Figure 3: Depiction of a butterfly network. Nodes aligned vertically are in the same level and nodes aligned horizontally are in the same row.

Proof We will see the proof of a weaker result than is needed to establish the theorem. The proof establishes a congestion bound that leads to an $O(\lg^2 n)$ time result.

WLOG, we route packets in three phases (figure 3 shows the rows and levels in a butterfly network):

1. Route packets straight across rows in the network to the corresponding outputs (the rightmost node in each row).
2. Use greedy input to output routing, to get each packet to the correct row in the network.
3. Route packets straight across to the final destination.

We analyze the congestion at the output nodes at each row of the network in Phase 1. All packets from a given row end up at the output node of that row at the end of the phase. Each row contains $\lg n$ nodes, and $\lg n = O(\lg N)$, so the congestion in Phase 1 is $O(\lg n)$.

For Phase 2, we consider a node x at the kth level of the network (where the inputs are the 0th level and the outputs are the $(\lg n)$th level). By symmetry, the x is equivalent to any other node at the kth level. The number of packets that can reach x during Phase 2 is $2^k \lg n$ since there is a path to each node at the kth level of a butterfly network from 2^k input nodes and each input node has at most $\lg n$ packets at the beginning of Phase 2. Thus,

$$\text{worst case congestion at a node } x \text{ at the } k\text{th level} = 2^k \lg n.$$

The probability that a given packet passes through node x is at most 2^{-k}. This result follows from the binary tree property of butterfly networks and the fact that the destination of each packet is chosen independently and uniformly at random from all the nodes in the network. (See figure 4 for a depiction of the binaray tree property). Thus,

$$\Pr[\text{a given packet passes through node } x] \leq 2^{-k}.$$

Consider any set of r specific packets. Since the destinations of the packets are independent of one another, the probability that all the packets pass through x is given by

$$\Pr[\text{all } r \text{ packets pass through } x] \leq (2^{-k})^r = 2^{-kr}.$$

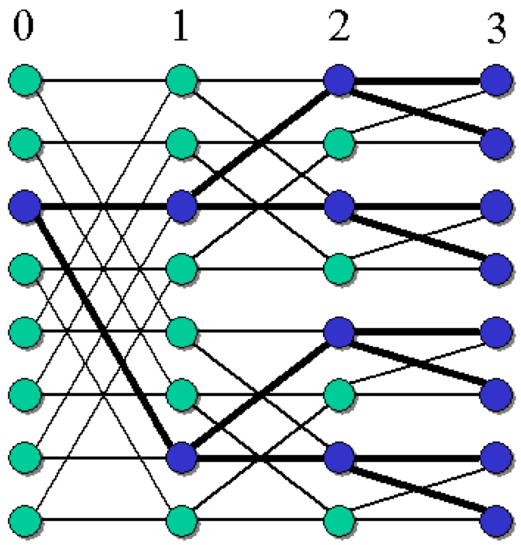

Figure 4: Binary tree property of a butterfly network. Each choice on a path from an input node to an output node is the head of a binary tree.

The probability that at least r packets pass through x is bounded by the number of ways to choose r packets from the total number of packets multiplied by the probability that all r packets pass through x:

$$\Pr[\geq r \text{ packets pass through } x] \leq \binom{2^k \lg n}{r} 2^{-kr}.$$

Note: This argument overcounts. If $r + \Delta$ packets pass through x, this event is counted $\binom{r+\Delta}{r}$ times within the $\binom{2^k \lg n}{r}$ ways.

Using the formula $\binom{a}{b} \leq \left(\frac{ea}{b}\right)^b$, we have

$$\begin{aligned}\Pr[\geq r \text{ packets pass through } x] &\leq \left(\frac{e2^k \lg n}{r}\right)^r 2^{-kr} \\ &= \left(\frac{e \lg n}{r}\right)^r.\end{aligned}$$

If we choose $r = 2e \lg N$, then we obtain

$$\begin{aligned}\Pr[\geq r \text{ packets pass through } x] &\leq \left(\frac{1}{2}\right)^{2e \lg N} \\ &\leq N^{-2e} \\ &\leq \frac{1}{N^{5.4}}.\end{aligned}$$

By Boole's inequality, the probability that *any* node has more than $r = 2e \lg N$ packets is $\leq N$ times the probability that a given node has more than r packets:

$$\begin{aligned}\Pr[\geq r \text{ packets pass through some node}] &\leq N\left(\frac{1}{N^{5.4}}\right) \\ &= N^{-4.4}.\end{aligned}$$

Therefore, $\geq N^N(1 - 1/N^{4.4})$ routing problems see $\leq 2e \lg N$ congestion. Since there are $O(\lg N)$ levels and each node at a level has at most $O(\lg N)$ congestion, the total congestion for Phase 2 is $O(\lg^2 N)$. At the end of Phase 2, there are $O(\lg N)$ packets at each output node with high probability. Therefore, at each node in Phase 3 there is $O(\lg N)$ congestion with high probability. Thus, the overall time bound is $O(\lg^2 N)$ for at least $N^N(1 - 1/N^{\Omega(1)})$ routing problems. $\square$

Corollary 7

$$\begin{aligned} E[\textit{routing time}] &= O(\lg N)(1 - 1/N^{4.4}) + O(N)(1/N^{4.4}) \\ &= O(\lg N).\square \end{aligned}$$