Lecture 6

1 Overview

Last time, we showed you that a variant of the butterfly network, the Benes network, could route any permutation, as long as one computed the routing "off-line." Today, we will discuss routing on-line in a butterfly network. We will also discuss the effect of scheduling.

2 On-line solutions: Valiant/Brebner routing.

Benes routing in some sense says that the hypercube/butterfly are great networks. On the other hand, the algorithm is not very "parallel", i.e., for each pattern of messages one needs to figure out how to route them.

Here we consider a really parallel algorithm for routing permutations in the Benes network. Route to a random intermediate destination in the first butterfly. Then route greedily to the final destination. This method was proposed by Valiant and Brebner for the hypercube.

By the way, one can view this equivalently, as flipping a coin at each level in the butterfly and deciding whether to route up or down.

Before analysing this method, let's get some basics. What is the highest load on any intermediate destinations? This is just the question of what the maximum load on any of n bins when one throws n balls into them. With high probability, we know it to be $O(\log n/\log\log n)$.

(BTW, Valiant has done several truly amazing things in complexity, learning, algorithms, and parallel computing. This is simple but elegant example of his contributions.)

Theorem 1
The maximum number of packets that use any edge is $O(\log n)$ with high probability.

What is the expected number of messages that use any edge? Consider the ith level in the first butterfly. The number of inputs that could use it are 2^i.

That is the load can be expressed as the sum of 2^i $0-1$ random variables. The number of outputs it can get to is 2^{n-i} (within a factor of 2). The probability of any of the inputs choosing this output is thus $2^{n-i}/2^n = 2^{-i}$ Thus, the expected number of messages is 1.

We finish by using a Chernoff bound. Here, we have random indicator variables $X_1, \ldots, X_k$, where $k = 2^i$ for a level i edge.

Here the expectation of $X = \sum_i X_i$ is 1 and we wish to show that max load is at most $O(\log n)$, thus we will choose $\delta = 2\log N$. We then use the form that states that

$$P[X > (1+\delta)\mu] \leq 2^{-\mu\delta}.$$

Plugging in, we get that the max load is at more $(2 \log n + 1)$ with probability than $1/N^2$. This allows us to conclude that any of the $N \log N$ edges is congested with probability at most $\log N/N$ using the union bound. Thus, the theorem follows.

Actually, we can show with high probability that max load is $O(\log n/ \log\log n)$. The probability that the load on a level i edge exceeds congestion k is at most

$$\binom{K}{k} 1/K^k \leq \left(\frac{e}{k}\right)^k.$$

This is less than $1/N^2$, when one chooses, for example, $k \geq 4 \log N/ \log\log N$.

3 Scheduling.

In the previous section, we showed how to select routes that are good in the sense of congestion. We next discuss whether that is sufficient, whether we can actually schedule packet movements to solve the problem within the congestion bounds. We refer to this as the scheduling problem.

In this section, we will examine the details of the routing a bit more closely. Notice, though that at each switch, two packets could come in, and both could be destined for the output, packets will build up at the switch.

Thus, we may need to store packets at internal switches. For now, we wish to avoid backups, so will allow ourselves to do so.

In particular, let's assume that the packets are routed FIFO, and each switch has sufficient queues so that any packet that is destined for it, can be transmitted.

Clearly, we should be able to route any permutation in $O(\log^2 N/ \log\log N)$ time on a Benes with high probability. You can't possibly wait more than $O(\log N/ \log\log N)$ setps at any edge.

But perhaps one can do a bit better. First, we define a concept called path congestion.

Definition 1 *The path congestion of a path, P in a routing is the sum over the edges in the path of the other packets that use the edge. The path congestion of the routing is the maximum over the paths of their path congestion.*

Now, if the path congestion is P_c, each packet only "sees" P_c packets and thus should only be delayed P_c times. That is, in each step either the packet moves or someone else moves. Thus, the routing time is at most $D + P_c$, where D is the total length of a packet's path, and P_c is the total movement on the packet's path.

What is the path congestion for permutation routing? Well, here we do two steps. We first bound the number of packets that touch the path at all by $O(\log N)$? Then, we need to say that they don't use the path for too long. That is, for example, each packet that touches the path doesn't stay on the path for say $\Omega(\log N)$ steps.

The, first step is easy. That is, for each packet, X_p we have a $0-1$ variable that indicates whether it hits the path. For packets in the same subbutterfly at level i, the probability is around $1/2^i$. There are around 2^i such packets. Thus, one gets the expectation of the sum of these random variables, which is the path congestion, is around $O(\log N)$. We can once again use a Chernoff bound to show that with probability $1 - 1/n^2$ this random variable has a value below $O(\log N)$. This time we use the full power of the Chernoff bound; i.e., the probabilities of hitting the path are different and thus the random variables that we are adding are not independently distributed.

The second step is to show the number of hits is $O(\log N)$. We can assume that we start with at most $K = O(\log N)$ packets. For each packet, the probability that it hits the path again is $1/2$. The probability it hits it yet again is $1/2$, and so on. This is an exponentially distributed variable. And the path congestion just the sum of exponentially distributed variables with mean 1. There is surely some bound that bounds the probability that this sum is large. For the sake of completeness, we will do it differently and break it down as follows.

Each of the K random variables can be viewed as a sequence of coin tosses. The random variable is l if the coin tosses come up with heads l times in a row.

Thus, the sequence $HHTHHHTHHHT$ means that packet 0 stayed on the path two times after the first, packet 1 hit three times, packet 3 three times. The length of the sequence is the total length is the path congestion.

For the number of "heads" in the sequence to be k (we know the number of tails is K), an upperbound on the probability is the sum over choices of $\{l_1, \ldots, l_K\}$ where the $\sum_i l_i = k$ of

$$\pi_{v \in \{l_1, l_2, \ldots, l_K\}} (1/2)^v = \left(\frac{1}{k}\right)^k$$

The number of terms choices of $\{l_1, \ldots, l_K\}$ is $\binom{k+K}{K} \leq ((k+K)e/K)^K$. Let's assume that $k = cK$ for some value of c. An upper bound on the probability is

$$((c+1)e)^K (1/2)^{cK}.$$

Question 1: Show that for $K = O(\log N)$, that c can be a constant and the probability will be less than $1/N^2$

This is also $O(\log N)$ with high probability.

Remark 1 As someone asked in some class. This is a bit strange in that the second depends on the first. A bit more formally, we first bound the probability of event A, which is the event that more than say $4 \log N$ packets see the path at all. Then, we bound the event, B, that the total congestion on the path is too high given that not too many packets hit the path. (I.e. the packets don't overstay their welcome.) That is, we bound $Pr[B|\overline{A}]$. Thus, using $Pr[B] \leq Pr[A] + Pr[B|\overline{A}]$, we have a bound on $Pr[B]$.

Thus, the total number of hits is at most $O(\log N)$, and thus greedy routing on the butterfly routes in time $O(\log N)$.