Online-routing on the butterfly network: probabilistic analysis

Andrey Gubichev

19.09.2008

Contents

1 Introduction: definitions

In this talk we will examine the average-case behavior of the greedy algorithm in butterfly network. Let us first introduce some useful notions and give simple examples.

Definition 1 (Butterfly)**.** *The r-dimensional butterfly consists of $(r+1)2^r$ nodes and $r2^{r+1}$ edges. A node is a pair $\langle w, i\rangle$, i is the level of the node, w is the row number (r-bit). An edge links two nodes $\langle w, i\rangle$ and $\langle w', i'\rangle$ if and only if $i' = i + 1$ and either $w = w'$, or w and w' differs in the ith bit.*

Figure 1 shows an example of 3-dimensional butterfly.

The packet routing problem is the problem of routing N packets from level 0 to level $\log N$ in a $\log N$-dimensional butterfly. Each packet $\langle u, 0\rangle$ has its own destination $\langle \pi(u), \log N\rangle$ where $\pi : [1, N] \to [1, N]$ is a permutation.

The most commonly used permutations are the bit-reversal permutation:

$$\pi(u_1 \cdots u_{\log N}) = u_{\log N} \cdots u_1,$$

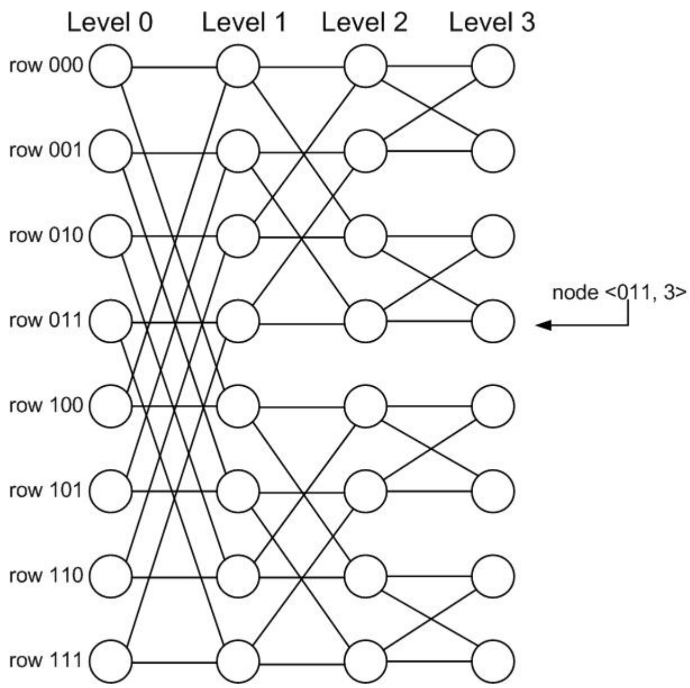

Figure 1: Three-dimensional butterfly.

and the transpose permutation:

$$\pi(u_1 \cdots u_{\frac{\log N}{2}} u_{\frac{\log N}{2}+1} \cdots u_{\log N}) = u_{\frac{\log N}{2}+1} \cdots u_{\log N} u_1 \cdots u_{\frac{\log N}{2}}$$

We will insist that our routing algorithms be *on-line*: there is no global controller that can precompute routing paths, each node decides what to do with a packet that pass through it based on its local controller and information from packet.

Definition 2. *The greedy path from* $\langle u, 0\rangle$ *to* $\langle v, \log N\rangle$ *is the unique path of length* $\log N$ *from the first node to the second node.*

The *greedy* algorithm is the algorithm that constrains each packet to follow its greedy path.

The congestion problem is that many packets might pass through a single node or edge, but only one packet can use the particular edge or node at a time.

Theorem 1. *The greedy algorithm will route* N *packets to their destinations in a* $\log N$*-butterfly in* $O(\sqrt{(}N))$ *steps.*

In fact, the bit-reversal permutation and the transposal permutation are the worst-case permutations for greedy routing.

In the following section we will find out that in average case the greedy algorithm behaves much better.

2 Average case behavior of the greedy algorithm

We will divide our analysis into two parts. First of all, we will bound the congestion. If we obtain the bound C for congestion, we will automatically have a bound for the running time: $(C-1)\log N$. In the second part we will get a tighter bound for the running time.

In this section we consider the routing problem for which each packet has a random destination (destinations are selected independently and uniformly from among the N possible outputs). Here we also allow more than one packet to start at each input (denote by p the number of packets at each input).

2.1 Bounds on congestion

Theorem 2. *For all but at most a $1/N^{3/2}$ fraction of the possible routing problems with p packets per input in a $\log N$-dimensional butterfly at most C packets pass through each node during a greedy routing where*

$$C = \begin{cases} 2ep, \text{ if } p \geq \dfrac{\log N}{2} \\ 2e\log N/\log\left(\dfrac{\log N}{p}\right), \text{ if } p \leq \dfrac{\log N}{2} \end{cases}$$

Proof. Our main aim here is to bound the probability $P_r(v)$ that r or more packet paths pass through some node v for each $r > 0$ and for each node from $\log N$-dimensional butterfly.

Let v be the node on ith level of the butterfly. There are 2^i inputs that can reach v and $2^{\log N - i} = N2^{-i}$ choices of destinations that can cause the packet to pass through v. Since we choose destinations randomly among N destinations, the probability for each of $p2^i$ packets to pass through v is $N2^{-i}/N = 2^{-i}$. A simple illustration is given on figure 2.

If r or more packets pass through v, then there exists a subset of r packets and all of them must pass through v:

$$P_r(v) \leq \binom{p2^i}{r}(2^{-i})^r \leq \left(\frac{p2^ie}{r}\right)^r 2^{-ir} = \left(\frac{pe}{r}\right)^r$$

The upper bound does not depend on v, i. Hence, the probability that r or more packets pass through all nodes in the butterfly is at most $N\log N(pe/r)^r$.

This bound decreases if r increases. If $p \geq \frac{\log N}{2}$, let $r = 2ep$ and we get

$$N\log N\left(\frac{pe}{r}\right)^r \leq N\log N\left(\frac{1}{2}\right)^{e\log N} = N^{1-e}\log N \leq 1/N^{3/2}$$

In case that $p \leq \frac{\log N}{2}$ let $r = \frac{2e\log N}{\log\left(\frac{\log N}{p}\right)}$ and $x = \frac{\log N}{p} \geq 2$:

$$N\log N\left(\frac{pe}{r}\right)^r = N\log N\left(\frac{\log x}{2x}\right)^{\frac{2e\log N}{\log x}} = N\log N N^{-\frac{2e\log(2x/\log x)}{\log x}}$$

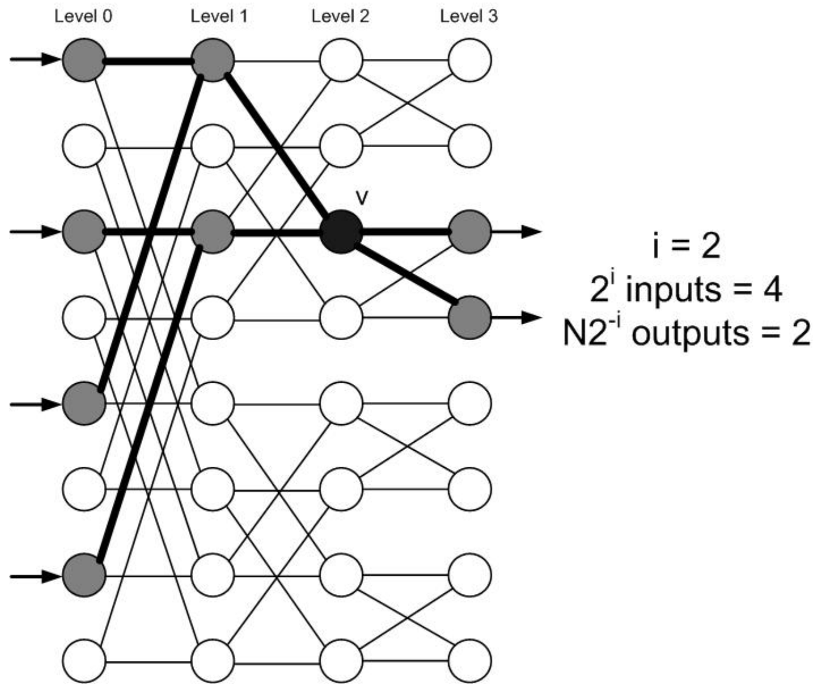

Figure 2: Choices of inputs and destinations that cause the packet to pass through v

The minimum for $\frac{\log(2x/\log x)}{\log x}$ where $x \geq 2$ occurs if $\log x = 2e$:

$$N \log N N^{-\frac{2e \log(2x/\log x)}{\log x}} \leq N \log N N^{-2e+\log x} \leq 1/N^2$$

These facts complete the proof. □

In fact, the fraction of "bad" routing problems can be made arbitrary small.

Corollary 1. *$\forall \alpha$ the congestion for all but $1/N^\alpha$ problems is at most $O(\alpha p) + o(\alpha \log N)$*

Proof. In order to prove it we can modify the previous proof.

If $p \geq \frac{\log N}{2}$ let $r = 2ep\alpha = O(p\alpha)$:

$$N \log N \left(\frac{pe}{r}\right)^r \leq N^{-\alpha}$$

If $p \leq \frac{\log N}{2}$ let $r = \frac{2e \log N}{\log\left(\frac{\log N}{p}\right)} = o(\alpha \log N)$:

$$N \log N \left(\frac{pe}{r}\right)^r \leq N^{-\alpha}$$

□

Now we see that routing problems with bit-reversal permutation and transpose permutation are incredibly rare: for 99% of all routing problems at most $C + O(1)$ packets pass through any node during the routing.

Let us consider two special cases of the theorem. If $p = 1$, we have a single N-packet routing problem. With high probability, the maximum number of packets that pass through any node is $O(\log N/\log\log N)$ with high probability.

The second case is when $p = \Theta(\log N)$, and we have $\Theta(N \log N)$ packets on an $N \log N$-node butterfly. At most $O(\log N)$ packets pass through any node with high probability.

2.2 Bounds on running time

If two or more packets are waiting to exit a node, we need to specify a protocol for deciding which packet will exit the node first. We will use a random-rank protocol in such cases:

- assign a random priority key $r(P) \in [1, K]$ to each packet P
- define a total order on packets: $t(P)$ is the rank of P
- define $w(P) = (r(P), t(P))$. If $P/ = P'$, we say that $w(P < w(P')$ if and only if $r(P) < r(P')$, or $r(P) = r(P')$ and $t(P) < t(P')$.
- if there is a collision, we choose the packet with minimal w

Consider the routing problem with congestion C. Let P_0 be the last packet to reach its destination v_0 at time T. It was delayed at v_1, l_0 is the number of steps in path $v_1 \to v_0$. P_0 was delayed during the step $T - l_0$.

Let P_1 be the packet responsible for delaying P_0. Next record the path of P_1 from the time it was last delayed before step $T - l_0$ until the step $T - l_0$. Let l_1 be the number of edges in this path and v_2 the node where P_1 was last delayed at step $T - l_0 - l_1 - 1$.

We proceed to record the sequence of delays and remove repeated nodes. We get *delay path* $P = v_0 \to v_1 \to \ldots \to v_s$ - a simple path of length $\log N$.

An example of the delay path is given on figure 3. Each packet on the figure 3 consists of the destination (binary number),the name and the random rank.

It is obvious that $T - l_0 - l_1 - \cdots - l_s - (s - 1) = 1$ and $l_0 + \cdots + l_s = \log N$. Hence, $T = s + \log N$.

An active delay sequence consists of

- a delay path $\mathbf{P}$
- integers $l_0 \geq 1, l_1 \geq 0, \ldots, l_{s-1} \geq 0$, $l_0 + \ldots + l_{s-1} = \log N$
- nodes v_0,v_1,. . .,v_s: v_i is the node of $\mathbf{P}$ on level $\log N - l_0 - \ldots - l_{s-1}$
- different packets $P_0, P_1, \ldots, P_s$: the greedy path for P_i contains v_i
- keys $k_0, k_1, \ldots, k_s$ for the packets: $k_s \leq k_{s-1} \leq \ldots \leq k_0$, $k_i \in [0, K]$ and $r(P_i) = k_i$ for $0 \leq i \leq s$.

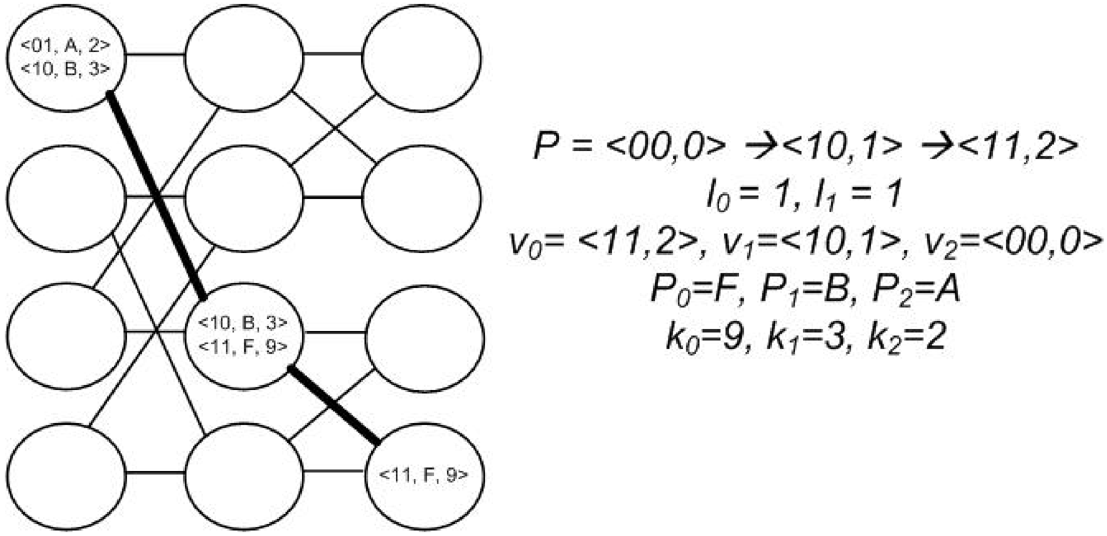

Figure 3: Delay path

There exist lots of possible delay sequences.

The probability that $r(P_i) = k_i$ for $0 \le i \le s$ is $K^{-(s+1)}$. There are N^2 possible delay paths (they are uniquely defined by endpoints). There are $\binom{s+\log N-2}{s-1}$ choices for $l_0, \ldots, l_s$: there is one-to-one correspondence between choices for l_i and $(s + \log N - 2)$-bit binary string t with $s-1$ zeros, where l_i is the number of "1" between $(i+1)$st and $(i+2)$nd zeros in the string $01t0$

Since we fix P and l_i's, the nodes v_i are determined. Then there are at most C choices for any P. Hence, there are at most C^{s+1} choices for all packets. We also have $\binom{s+k}{s+1}$ ways to choose $k_0, \ldots, k_s$ such that $k_s \le k_{s-1} \le \cdots \le k_0$ and $k_i \in [1, K]$: there is one-to-one correspondence between choices for k_i and $(s + K)$-bit binary string u with $s+1$ zeros, where k_i is the number of "1" to the left of the $(s+1-i)$th zero in the string $1u$.

Put it all together: the probability that there is an active delay sequence with $s+1$ packets is at most

$$N^2 \binom{s+\log N-2}{s-1} C^{s+1} \binom{s+K}{s+1} K^{-(s+1)}$$

We can show that if $K = s+1 = 8eC$, and $C \ge \frac{\log N}{2}$, this probability is at most

$$N^3 \left(\frac{4eC}{K}\right)^K \le N^{3-4e} = o(N^{-7}),$$

and if $K = s+1 = 8e \log N / \log\left(\frac{\log N}{C}\right)$, and $C \le \frac{\log N}{2}$, this probability is at most

$$N^3 \left(\frac{4eC}{K}\right)^K \le o(N^{-12})$$

Our result is that with high probability there is no active delay path with $s+1$ packets, where

$$s+1=\begin{cases} 8eC, \text{ if } C \geq \dfrac{\log N}{2} \\ 8e\log N/\log\left(\dfrac{\log N}{C}\right), \text{ if } C \leq \dfrac{\log N}{2} \end{cases}$$

Since $T=\log N+s$, we get

$$T=\begin{cases} O(C)+\log N, \text{ if } C \geq \dfrac{\log N}{2} \\ O(\log N/\log\left(\dfrac{\log N}{C}\right)), \text{ if } C \leq \dfrac{\log N}{2} \end{cases}$$

This fact completes the analysis.

3 Conclusion

"Typical" routing problem in practice are not at all the same as "typical" routing problems in a mathematical sense: while the latter are likely to have a reasonable running time, the former have very bad estimation of running time.

4 Bibliography

1 F. T. Leighton. Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes. Morgan Kaufmann Publ., 1992

2 Friedhelm Meyer auf der Heide. Kommunikation in Parallelen Rechenmodellen.