Butterfly routing

Adnan Aziz

The University of Texas

References:

- Parallel Algorithms and Architectures. T. Leighton. Morgan Kaufmann, 1992.

Butterfly formalized

r-dimensional butterfly — $(r+1) \cdot 2^r$ nodes, $r \cdot 2^{r+1}$ edges

- use $\langle w, i \rangle$ to represent a node at level i, and row w
- edge $(\langle w, i \rangle, \langle w', i' \rangle)$ iff $i = i' + 1$ and w and w' are equal or differ only in i'-th bit
  - straightedges, and crossedges

Note there exists a unique path from any level 0 vertex to any level r vertex

- path traverses each level exactly once, takes a cross edge iff source and destination differ in the $i+1$ bit

Analogous to Boolean n-cube

- merge all nodes in a row, remove extra edge

r-dimensional butterfly contains 2 $r-1$-dimensional butterflys (remove level 0 or level r nodes to see this)

Wrapped butterfly

Take butterfly, and merge $\langle w, 0 \rangle$ with $\langle w, r \rangle$ — left with r levels

- alternately, on $(r) \cdot 2^r$ nodes, $(r-1) \cdot 2^{r+1}$ edges
- edge $(\langle w, i \rangle, \langle w', i' \rangle)$ iff $i = i' + 1 \bmod r$ and either $w = w'$ or w and w' differ in i-th bit

Theorem: Given N node wrapped butterfly with $\leq$ 1 packet/node, and permutation $\pi : \{1, \ldots, N\} \mapsto \{1, \ldots, N\}$, there is a way of routing packets through the butterfly in $\leq 3 \cdot \lg N$ steps.

Proof idea — view as $2^r \times r$ array, use offline routing algorithm for arrays

- use straight edges for Phases 1 and 3

Observe: all packets starting at $\langle u_1, u_2, \ldots, u_{\lg N/2}, 0 \ldots, 0 \rangle$ pass through $\langle 0, 0, \ldots, 0 \rangle$

- given $\Omega\sqrt{(N/2)}$ lower bound on greedy algorithm
- same phenomenon with "transpose" routing — $\langle u_1, u_2, \ldots, u_{\lg N/2}, \ldots, u_{\lg N} \rangle$ going to $\langle u_{\lg N/2+1}, \ldots, \ldots, u_{\lg N}, u_1, u_2, \ldots, u_{\lg N/2} \rangle$

Theorem: Greedy algorithm completes in $O(\sqrt{N})$ steps

Proof idea — for any edge at level i, we have $\leq 2^{i-1}$ greedily chosen paths which could enter and at most $2^{\lg N - i}$ greedily chosen paths which could exit

- bound on number of greedy path passing through an edge is $O(\sqrt{N})$

For small N (¡ 100) not too bad, since $\sqrt{N}$ is not that far from the lower bound of $\lg N$

- for large N, complexity is an issue, especially since many natural problems exhibit worst case

- traverse butterfly to and from end ⇒ effectively a Benes network, can perform Phase 2

No more than 3 pkts ever at any node — one which has reached its destination, one traveling in each direction

Online routing

Previous algorithm needed global knowledge of packets and their destinations

- not very practical (also, cannot parallize it)

Would prefer algorithms where local routing decisions are made without precomputation and without global knowledge

- will restrict attention to 1-to-1 static end-to-end routing

Consider performing "bit reversal" permutation routing with greedy algorithm

- packet at input $\langle u_1, u_2, \ldots, u_{\lg N} \rangle$ headed to output $\langle u_{\lg N}, u_{\lg N-1}, \ldots, u_1 \rangle$

Average case behavior of greedy algorithm

Results based on "congestion analysis"

For most routing problems, with p packets/input, at most C packets pass through each node

- $C = O(p)$ if $p \geq \log N/2$
- $C = O(\log N/\log(\log N/2))$ if $p \leq \log N/2$

Succinctly, $c \leq O(p) + o(\log N)$

Note that the queue size is never more than the congestion

Analysis is easier when we use the following congestion resolution protocol:

- give a random rank to each packet (a number in $[1 \ldots \Theta(p + \log N)]$)
- lower rank wins (break ties using destination address)

Theorem: Given any routing problem with congestion C on a $\log N$ dimensional butterfly, the greedy algorithm will complete in T steps with probability $\geq 1 - N^{-7}$ when the random rank protocol is used

- $T = O(C)$, if $C \geq \log N/2$, and
- $T = \log N + O(\log N/\log(\log N/C))$ if $C \leq \log N/2$

Again, can use Valiant's paradigm

Fact: queues can grow to be $\Theta(\sqrt{N})$); if queues are restricted to be $O(1)$ sized, then greedy routing can take $\Theta(N)$ steps

Fact: if each of the $N \lg N$ nodes holds a packet, then the greedy algorithm completes in $\Theta(\sqrt{(N \lg N)})$ steps in worst case

Fact: if path depends only on origin & destination (and not other packet's source/destination, congestion) need $\Omega((\sqrt{N})/d)$ steps for 1-to-1 routing

Greedy algorithm is good for routing $M \leq N$ packets to first M slots

- referred to as "packing"

Greedy routing good for "spreading"

- reverse of packing

Greedy routing good for "monotone routing"

- pack and spread

Removing queues

Use "information dispersal"

- break each packets into a collection of subpackets & route each subpacket in a greedy fashion
  - more balance
  - encode with some redundancy $\Rightarrow$ fault tolerance
  - need more bits for addressing, routing; more packets

Use coding theory to make a B bit packet P into $2B/\log N$ packets, each $B/\log N$ bits in size

- recover P from $\log N/2$ subpackets

Ranade's algorithm

Previously, no hard cap on queue size – unlikely to be larger than $O(p) + o(\log N)$

What if queue size fixed say to Q?

- use "back pressure"

Ranade's algorithm

- random rank contention resolution
- require packets pass through each node in sorted order
  - need "ghost packets" to inform neighbor of current transfer

Again, given any routing problem with congestion C on a $\log N$ dimensional butterfly with queues of max size $q \geq 1$,

greedy algorithm completes in T steps with probability $\geq 1 - N^{-7}$ when the random rank protocol is used, where

- $T = O(C)$, if $C \geq \log N/2$, and
- $T = \log N + O(\log N/\log(\log N/C))$ if $C \leq \log N/2$

Use in conjunction with Valiant's paradigm