Array routing

Adnan Aziz

The University of Texas

References:

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

Packet routing problem

Given a network (modeled as a graph), assume M packets each headed to node p_i, find "best" way of routing packets to destinations

- use as few "steps" as possible, under constraint that only one packet traverses a link in each step
- desirable to have local control at each node making forwarding decisions
- often "1-to-1" routing: unique destinations

Greedy routing

Route packets along shortest path (not always unique)

- linear array: greedy algorithm moves packets left/right as needed
  - assume: start with at most 1 packet at each node, 1-to-1 routing
  - observe: never any contention for a link
  - each packet reaches its destination in at most d steps
  - example $[3, 6, 1, 5, -, 2]$

Resolving contention

General networks can have contention

- strategies for resolving contention, even if routing is constrained to be greedy: farthest-first, blocking
  - assume infinite queues for now

On 2-D $\sqrt{N} \times \sqrt{N}$ array: basic greedy algorithm is farthest-first greedy, with shortest path chosen to traverse rows then columns.

Theorem: basic greedy algorithm will always route in at most $2\sqrt{N} - 2$ steps

Fix i, define the "priority packets" to be those which are headed to one of the i rightmost nodes

- a priority packet can never be delayed by a nonpriority packet, so we'll ignore these

Consider the rightmost priority packet (break ties arbitrarily)

- it never gets blocked, so gets to one of the i rightmost nodes in $N - i$ steps

Next rightmost priority packet gets to one of the i rightmost nodes in $N - i + 1$ steps

- in general, the i-th is delayed for at most $i - 1$ steps, takes $N - i$ edges, hence $N - 1$ steps

Clearly, this time is optimum

- queue length can grow large: $\Theta(\sqrt{N})$

Average case analysis

How does the basic greedy algorithm for 2-D arrays perform on the average?

Various notions of "average"

- static — each node has one packet for a random location (possibly more than one to the same node)
- dynamic — packets generated at random times over arbitrarily long time intervals (issues: delay, queue size, stability, throughput)

Analysis: reach correct column in at most $\sqrt{N} - 1$ steps (no contention to worry about)

- may have queuing at nodes when traversing columns, can't use linear array argument

Lemma: In an N node array, with an arbitrary number of packets per node, but with only one packet destined to any node. If contention is resolved by giving priority to packets that need to go farthest, packets will be routed in at most $N - 1$ steps.

Proof: left and right moving packets don't interfere $\Rightarrow$ focus on right moving packets

For each i, we'll show that any packet destined to one of the i-rightmost packets reaches one of the i rightmost nodes in $N - 1$ steps. (Note this suffices to prove the claim!)

For static case, probability that any packet is delayed Δ steps is at most $e^{-\Delta/6}$

- probability that it reaches in $d + O(\log N)$ steps is at most $1 - O(1/N)$
- never more than 4 packets waiting in a queue for some edge with probability almost 1

Spirit behind analysis:

- no contention on rows
- queue size increases on a column only when two or more packets arrive simultaneously
- say a packet "turns" on a node if it comes into node on row edge, leaves on column edge
  - size of queue is at most number of packets turning at node
- since at most $\sqrt{N}$ packets turn in each row, and destinations are random, probability that at least r packets turn in some node is $C_r^{\sqrt{N}} \cdot (1/\sqrt{N})^r \leq (e/r)^r$
- actually much better: don't necessarily turn at same time

Dynamic case

New packet shows up with probability λ at each step at each node, has a random destination

- Must have $\lambda < 4/\sqrt{N}$, otherwise unstable, since only $\sqrt{N}$ can cross from left to right

Theorem: If arrival rate is at most 99% of capacity, then probability of delay for Δ steps is at most $e^{-c\Delta}$, where c is independent of N, and time of operation

Valiant's paradigm

Send packet to first to a random intermediate destination, then to final destination

- use basic greedy routing

Fact: for **any** 1-to-1 routing problem, this approach terminates in $O(\sqrt{N})$ steps, using queue lengths of $O(\log N)$ with high probability

Deterministic algorithms with small queues

Deterministic solutions are preferred

- can achieve in $6\sqrt{N} + o(\sqrt{N})$ steps using queue size of 1
  - sort by col destination, then apply basic greedy algorithm — sorting removes conflicts
- need 300% speedup: can achieve $2\sqrt{N} + 4\sqrt{N}/q + o(\sqrt{N}/q)$ complexity using $2q - 1$ sized queues

Online algorithms

Can perform routing in $3\sqrt{N} - 3$ steps with queues of size 1

- Phase 1: permute packets in each column so that at most one packet in each row is headed for each column
- Phase 2: route within rows to columns
- Phase 3: route within columns to rows

Hard part: permuting within columns