The Power of Random Bits
Randomized Algorithms: Applications & Principles

Part II: Random Routing and Load Balancing

Sid Chi-Kin Chau
sidckchau@gmail.com
Problem: Traffic Routing

- Suppose you are in charge of transportation. What do you do to reduce congestion?
  - Congestion is caused by traffic demand exceeding the capacity of transport resource
  - To build more roads (to increase capacity)?
  - To raise toll (to reduce demand)?
  - Or to optimize the traffic routes and schedules (from algorithmic design)?

- Here is a radical idea – “random routing”:
  1. A passenger wants to travel from a source to a destination
  2. Take a passenger from the source to a “random” location
  3. Then take the passenger from the “random” location to the destination

- Does this reduce congestion in transport networks?
- But this works in computer networks and telecommunication networks.
Random Routing in Tech Nets

- Technological networks are interconnections of many nodes of systems and machines
  - High-performance supercomputers require intense communications among computing nodes (CPUs, GPUs, storage units)
- Telecommunications need to forward numerous calls and data packets across places
  - The connections are often sparse (as to reduce connection costs)
    - Require multihop relaying from nodes to nodes
  - The nodes and links have limited I/O capacity
  - Unprocessed data are buffered in queues
- Congestion is caused by traffic demand exceeding network capacity at relays and links
  - Random routing is implemented in these networks to reduce congestion and improve performance
Many Internet backbone networks are massively over-provisioned to provide reliable services.

Hence, the links are vastly underutilized.

How can we minimize the resource provision with satisfactory reliability?

Valiant load balancing:
- The core backbone network is a full-meshed network.
- Instead of the direct route between the source and destination, the route has to traverse a random intermediate router (i.e., random routing).
- This balances the traffic among all routers in the core backbone network and averages out the utilization.
Parallel Routing in Hypercube

- Hypercube is an interconnection topology for supercomputers and peer-to-peer networks.
- There are $N = 2^n$ nodes, each labelled by an $n$-bit coordinate.
- There is a link between every pair of nodes with 1 bit difference in their coordinates.
- Each link can transmit one packet at one time, and excessive packets will be buffered at nodes.
- Assume that each node $i$ has a destination $d(i)$, which may not necessarily be a neighbour (hence requiring multihop forwarding and buffering at relays).
- What is the minimum schedule of parallel routing (i.e., a sequence of sets of activated links) to forward the traffic from all the sources to destinations?
- Any simple algorithms? Computationally hard to find the minimum schedule by deterministic algorithms.
A simple routing algorithm is oblivious to other flows -- find the shortest path between source and destination.

Bit-fixing routing is to find a path $\langle i_1, i_2, \ldots, d(i_1) \rangle$, where:

- $(i_t, i_{t+1})$ differ in only one bit for all $t$
- if $(i_{t-1}, i_t)$ differ in the $k$-th leftmost bit and $(i_t, i_{t+1})$ differ in the $l$-th leftmost bit, then $k < l$

There exists a configuration of sources and destinations that requires at least $2^{n/2}/2$ steps by bit-fixing routing:

- Consider $n$ is even, for every source $i = (1_i \, r_i)$, we assign the destination to be $d(i) = (r_i \, 1_i)$ (i.e., $d(i)$ is a transpose permutation of $i$).
- Then for source $i = (\ldots?1 \, 0\ldots00)$ and its destination $d(i) = (0\ldots00 \, ?\ldots?1)$ (i.e., $1_i$ is odd and $r_i$ is zero), it must traverse $(0\ldots01 \, 0\ldots00)$ by bit-fixing routing.
- There are $2^{n/2}/2$ nodes with address $(?\ldots?1 \, 0\ldots00)$.
- Only one source can traverse $(0\ldots01 \, 0\ldots00)$ at one step.
- At least $2^{n/2}/2$ steps needed for relaying from these nodes.
Random Routing in Hypercube

- For deterministic bit-fix routing, the worst case requires at least $2^{n/2}/2$ steps (exponential in $n$).
- But for random bit-fix routing, it requires $O(n)$ steps with high probability (i.e., using more than $O(n)$ steps has a vanishing probability converging to 0, as $n \to 0$).
- Random bit-fix routing has two stages:
  1. Pick a random node $r(i)$ in the hypercube independently, and use bit-fixing routing from $i$ to $r(i)$.
  2. Use bit-fixing routing from $r(i)$ to $d(i)$.
- Obviously, longer paths are needed for random bit-fix routing. *Then why is this better?*
- Intuition is that random routing can *average out* the worst case configuration from deterministic routing.
- The probability that a randomly generated configuration is the worst case is very low, and is vanishing for large $n$.
- This intuition is behind many randomized algorithms.
It suffices to show that it requires $O(n)$ steps with high probability for the first stage of random bit-fixing routing.

For each source $i$, let $P_i$ be the random path to a random node.

We observe a property of bit-fixing routing:

- If $P_i$ and $P_j$ intersect, then there is only one subpath of intersection.
- $P_i$ and $P_j$ cannot intersect at multiple disjoint subpaths, as there is a unique path between any pair of nodes.

Let $1(P_i, P_j)$ be the indicator function for testing if $P_i$ and $P_j$ intersect (once).

The delay for source $i$ is bounded by: $\text{delay}_i \leq \sum_{j=1}^{2n} 1(P_i, P_j)$.

Hence, the expected delay:

$$\mathbb{E}[\text{delay}_i] \leq \mathbb{E}[\sum_{j=1: j \neq i}^{2n} 1(P_i, P_j)] = \sum_{j=1: j \neq i}^{2n} \mathbb{E}[1(P_i, P_j)] \leq \sum_{e \in P_i} \sum_{j=1: j \neq i}^{2n} \mathbb{P}\{e \in P_j\}$$

where $e \in P_j$ denotes that $e$ is a link in the path $P_j$.

Continue in the next slide.
Principle of Random Routing

- Note that there are \( n2^{n-1} \) links in a hypercube and \( 2^n \) paths by bit-fixing, where each path has at most \( n \) links
- Thus, the expected number of paths including a particular link \( e \) is 2:
  \[ \sum_{j=1}^{2^n} \mathbb{P}\{e \in P_j\} \leq 2. \]
  Note that \( P_j \) contains at most \( n \) links
- Therefore, \( \mathbb{E}[\text{delay}_i] \leq \sum_{j=1:j \neq i}^{2^n} \mathbb{E}\left[1(P_i, P_j)\right] \leq 2n \)
- Our aim is to show that \( \mathbb{P}\{\sum_{j=1:j \neq i}^{2^n} 1(P_i, P_j) \geq cn\} \leq \frac{1}{2^n} \) for some \( c \)
- Hence, \( \mathbb{P}\{\text{delay}_i \geq cn\} \leq \frac{1}{2^n} \) (i.e., it takes \( O(n) \) steps with high probability)
- We note that \( P_i \) and \( P_j \) are independent random variables (because \( r(i) \) and \( r(j) \) are picked independently)
  - So \( 1(P_i, P_j) \) and \( 1(P_i, P_k) \) are independent random variables for \( i \neq j \neq k \neq i \)
  - Let \( X_j \equiv 1(P_i, P_j) \) be a Bernoulli random variable:
    \[ \mathbb{P}\{X_j = 1\} = \mathbb{E}[X_j] \leq \frac{n}{n2^{n-1}} \]
  - Obtaining the distribution of sum of independent Bernoulli random variables, \( \mathbb{P}\{\sum_{j=1}^{N} X_j \geq x\} \), requires Chernoff Bound
Summary

• Random routing takes a detour to a random intermediate node before reaching the destination
• Random routing can average out the worst case traffic patterns to deterministic routing algorithms
• Random routing has been implemented in telecommunication networks (Valiant load balancing) and in supercomputer architecture (parallel routing in hypercube)
• A key tool to prove the effectiveness of random routing is based on the Chernoff bound which estimates the exponential tail distribution of a sum of independent Bernoulli random variables
  • Hence, the probability that routing random deviates from the expected value is exponentially small in the size of network
  • Chapter 4.5: Packet routing in sparse networks
  • Chapter 4.2: Chernoff bound

• Additional reference
  • Rui Zhang-Shen and Nick McKeown, “Designing a Predictable Internet Backbone with Valiant Load-Balancing”, Proceeding Workshop of QUALITY OF SERVICE (IWQoS) 2005