Lecture 7: *Random Routing & Load Balancing*

Advanced Algorithms

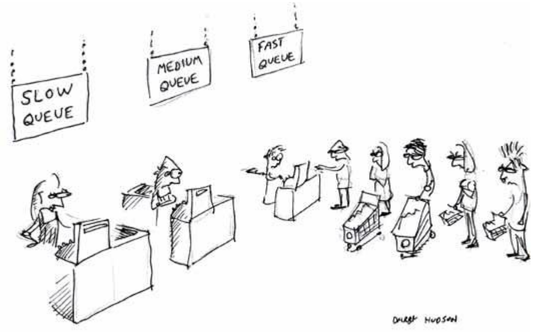
Sid Chi-Kin Chau

Australian National University

sid.chau@anu.edu.au

October 5, 2022

What is the best queuing strategy?



- Which queue will you pick? Shortest? Fastest? Or just random?

$\lll$How to *route* in a complex network, like the Internet?$\ggg$

Problem: Traffic Routing 🚦

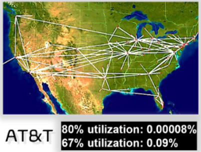
- Suppose you a minister of transportation. How can you reduce congestion?
  - Congestion is caused by traffic demand exceeding the capacity of transport resource
  - Solutions:
    - ★ To build more roads (to increase capacity)?
    - ★ To raise toll (to reduce demand)?
    - ★ Or to optimize the traffic routes and schedules (by better algorithm design)?
- Here is a radical idea – *random routing*
- Why does random routing reduce congestion in transport networks?

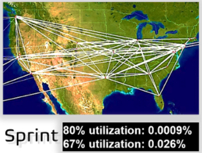
Random Routing

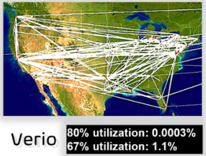
- A passenger wants to travel from a source to a destination
- Take a passenger from the source to a *random* location
- Then take the passenger from the *random* location to the destination

Random Routing in Technological Networks

- Technological networks are interconnections of systems and machines
  - High-performance computers require intense communications among computing nodes (CPUs, GPUs, storage units)
  - Telecommunication networks forward data packets among nodes
- The connections are often sparse (as to reduce connection costs)
  - Require multi-hop relaying from nodes to nodes
- The nodes and links have limited transmission capacity
  - Unprocessed data are buffered in queues
- Congestion is caused by the demand exceeding network capacity at capacitated relays and links
- Random routing can be implemented in technological networks to reduce congestion and improve performance

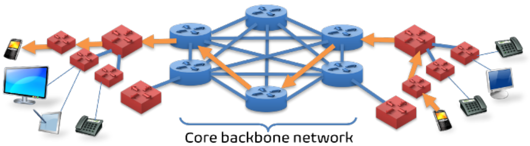






Valiant Load Balancing

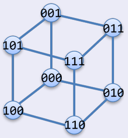
- Internet backbone networks are massively over-provisioned to provide reliable services
  - Hence, many links are vastly underutilized
- How can we minimize the resource provision with good reliability?
- Valiant load balancing:
  - The core backbone network is a full-meshed network
  - Instead of the 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 amortizes the utilization throughout the network



Parallel Routing in Hypercube

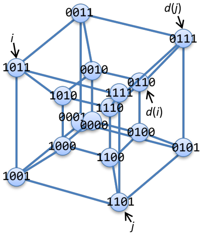
Definition (Hypercube)

- Hypercube is a common topology for computer 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



3D hypercube

- Assume each node i has a destination $d(i)$ (may not be a neighbor)
  - Hence, it requires multi-hop forwarding and buffering at some 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?
  - It is computationally hard to find the minimum schedule by a deterministic algorithm



4D hypercube

Bit-Fixing Routing in Hypercube

Definition (Bit-Fixing Routing)

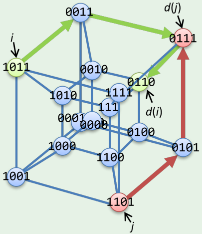
- Find a path $(i_1, i_2, \cdots, d(i_1))$, where
  - (i_t, i_{t+1}) differ in only one bit for all t
  - If (i_{t-1}, i_t) differ in the k-th leftmost bit and (i_t, i_{t+1}) differ in the l-th leftmost bit, then $k < l$

- Bit-fixing routing find the shortest path between source and destination
- There exists a set of sources/destinations requiring at least $\frac{2^{n/2}}{2}$ steps in bit-fixing routing
  - Consider n is even, for every source $i = (l_i \ r_i)$, and set its destination $d(i) = (r_i \ l_i)$ (i.e., $d(i)$ is a transpose permutation of i)
    - For source $i =$ (?...?1 0...00) and its destination $d(i) =$ (0...00 ?...?1) (i.e., l_i is odd and r_i is zero), it must traverse (0...01 0...00) by bit-fixing routing
  - There are $\frac{2^{n/2}}{2}$ nodes with address (?...?1 0...00)
    - Only one source can traverse (0...01 0...00) at one step
    - At least $\frac{2^{n/2}}{2}$ steps are needed for relaying from these nodes

Bit-Fixing Routing in Hypercube: Example

- There exists a set of sources/destinations requiring at least $\frac{2^{n/2}}{2}$ steps in bit-fixing routing
  - Consider n is even, for every source $i = (l_i \ r_i)$, we assign the destination to be $d(i) = (r_i \ l_i)$(i.e., $d(i)$ is a transpose permutation of i)
    - ★ For source $i = $ (?...?1 0...00) and its destination $d(i) = $ (0...00 ?...?1) (i.e., l_i is odd and r_i is zero), it must traverse (0...01 0...00) by bit-fixing routing

Example (Worst-case Bit-Fixing Routing)



Bit-fixing routing

i	→	$d(i)$
0000	→	0000
0001	→	0100
0010	→	1000
0011	→	1100
0100	→	0001
0101	→	0101
0111	→	1101
1110	→	1011
1111	→	1111

A worst-case configuration

Random Bit-Fixing Routing in Hypercube

Random Bit-Fixing Routing

- Pick a random relay node $r(i)$ in the hypercube independently at each time
- First, use bit-fixing routing from i to $r(i)$
- Then, use bit-fixing routing from $r(i)$ to $d(i)$

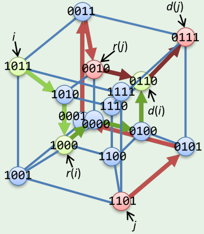
- For deterministic bit-fixing routing, the worst case is at least $\frac{2^{n/2}}{2}$ steps (exponentially)
- But for random bit-fixing routing, it requires $\mathrm{O}(n)$ steps with high probability
  - Using more than $\mathrm{O}(n)$ steps has a vanishing probability converging to 0, as $n \to 0$
- Obviously, longer paths are needed for random bit-fixing routing. But why is this better?
- The intuition is that random routing can amortize the worst case configuration from deterministic routing
  - The probability that a randomly generated configuration is the worst case is very low, and is diminishing for large n

Random Bit-Fixing Routing in Hypercube: Example

Random Bit-Fixing Routing

- Pick a random relay node $r(i)$ in the hypercube independently at each time
- First, use bit-fixing routing from i to $r(i)$
- Then, use bit-fixing routing from $r(i)$ to $d(i)$

Example (Random Bit-Fixing Routing)



Random bit-fixing routing

i	→	$r(i)$	→	$d(i)$
0000	→	0000	→	0000
0001	→	0001	→	0100
0010	→	1000	→	1000
0011	→	0101	→	1100
0100	→	0001	→	0001
0101	→	1110	→	0101
0111	→	1101	→	1101
1110	→	0000	→	1011
1111	→	1110	→	1111

A two-stage configuration

Principles of Random Bit-Fixing Routing

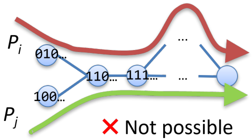
Theorem

The number of steps in random bit-fixing routing is $O(n)$ with high probability

Proof:

- 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 $\mathbb{1}(P_i \cap P_j)$ be the indicator function for testing if P_i and P_j intersect (once)
- The delay for source i is bounded by: $\text{delay}_i \leq \sum_{j=1}^{2^n} \mathbb{1}(P_i \cap P_j)$

Principles of Random Bit-Fixing Routing

Proof (Cont.):

- Hence, the expected delay:

$$
\begin{aligned}
\mathbb{E}[\mathsf{delay}_i] &\le \mathbb{E}\Big[\sum_{j=1:j\neq i}^{2^n} \mathbb{1}(P_i \cap P_j)\Big] \\
&= \sum_{j=1:j\neq i}^{2^n} \mathbb{E}[\mathbb{1}(P_i \cap P_j)] \\
&\le \sum_{e\in P_i} \sum_{j=1:j\neq i}^{2^n} \mathbb{P}(e \in P_j)
\end{aligned}
$$

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

- Note that there are $n2^{n-1}$ links in a hypercube and 2^n paths from 2^n nodes, where each path has at most n links

Principles of Random Bit-Fixing Routing

Proof (Cont.):

- Thus, the expected number of paths that include a particular link e is 2:

$$\sum_{j=1}^{2^n} \mathbb{P}(e \in P_j) \leq 2^n \frac{n}{n2^{n-1}} = 2$$

- Note that P_j contains at most n links. Therefore,

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

- Our aim is to show that $\mathbb{P}\big(\sum_{j=1:j\neq i}^{2^n} \mathbb{1}(P_i \cap P_j) \geq \mathrm{O}(n)\big) \leq \frac{1}{n}$
- Hence, $\mathbb{P}\big(\text{delay}_i \geq \mathrm{O}(n)\big) \leq \frac{1}{n}$ (i.e., it takes $\mathrm{O}(n)$ steps with high probability)

Principles of Random Bit-Fixing Routing

Proof (Cont.):

- Note that P_i and P_j are independent random variables
  - Because $r(i)$ and $r(j)$ are picked independently
  - So $\mathbb{1}(P_i \cap P_j)$ and $\mathbb{1}(P_i, P_k)$ are independent random variables for $i \neq j \neq k \neq i$
  - Fix a path P_i, let Bernoulli random variable $X_j \triangleq \mathbb{1}(P_i \cap P_j)$, where

$$\mathbb{P}(X_j = 1) = \mathbb{E}[X_j] \leq \frac{n}{n2^{n-1}}$$

- Hence, $\mathbb{P}\big(\mathsf{delay}_i \geq x\big) \leq \mathbb{P}\big(\sum_{j=1}^{2^n} X_j \geq x\big)$, which is the tail probability of a sum of independent Bernoulli random variables
- Chernoff bound for a sum of independent Bernoulli random variables shows that $\sum_{j=1}^{2^n} X_j$ is concentrated on $\sum_{j=1}^{2^n} \mathbb{E}[X_j]$, and its tail probability is exponentially decreasing:

$$\mathbb{P}\big(\mathsf{delay}_i \geq \mathsf{O}(n)\big) \leq \mathbb{P}\Big(\sum_{j=1}^{2^n} X_j \geq \mathsf{O}(n)\Big) \leq e^{-\mathsf{O}(n)}$$

Random Routing: Summary

- Random routing takes a detour to a random intermediate node from the source before reaching the destination
- Random routing can amortize the worst-case traffic by 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

⋘ A closer look at Balls and Bins model ⋙

Problem: Load Balancing ⚖️

- Round-Robin Load Balancing
  - Select each queue in rotation – the first job is assigned to queue A, the next job to queue B, and so on. Once a job is assigned to the last queue, the process repeats from queue A
  - When some queue has stalled, jobs are still assigned to that queue. The backlog will get longer and longer
- Shortest Queue Load Balancing
  - Watch each queue, and each time a job arrives, assign that job to the shortest queue
  - It seeks to balance the lengths of the queues, and avoids adding more jobs to a queue that has stalled
- Least Time Load Balancing
  - Each queue has a counter, indicating how many jobs have been processed in, for example, the last 10 minutes?
  - Then assign jobs to a queue based on its length and how quickly it is being processed. That's an effective way to distribute load

Power of Two-Choices Load Balancing

- Load balancing requires a *complete view* of all the queues and response time
  - Can we achieve load balancing without a complete view of all the queues and response time?
- Solution: **Power of two-choices** load balancing
  - Instead of making the absolute best choice by a complete view, with power of two-choices we pick two queues at random and choose the better option of the two, avoiding the worse choice
- Power of two-choices is an efficient solution without a complete view of all the queues
  - We don't have to compare all queues to choose the best option each time; instead, we only need to compare two
  - Counter-intuitively, it works better in practice than the best-choice algorithms, as it avoids the undesired herd behavior by the simple approach of avoiding the worst queue and distributing jobs with a degree of randomness
- Why does power of two-choices work?

Review: Balls-and-Bins Model

Definition (Balls-and-Bins Model)

Throw m balls into n bins, where each ball is uniformly distributed among the bins at random

- Key questions
  - Idling: How many non-empty bins are there?
  - Loading: What is the maximum number of balls in all the bins?
- Let random variable X_i be the random number of balls in the i-th bin
- The probability that the i-th bin has r balls follows a binomial distribution

$$\mathbb{P}(X_i = r) = \binom{m}{r}(\frac{1}{n})^r(1-\frac{1}{n})^{m-r} = \frac{1}{r!}\frac{m(m-1)\dots(m-r+1)}{n^r}(1-\frac{1}{n})^{m-r}$$

- We can approximate by Poisson distribution: $\mathbb{P}(X_i = r) \approx \frac{e^{\frac{-m}{n}}(\frac{m}{n})^r}{r!}$
- The probability of a non-empty bin is

$$\mathbb{P}(X_i \neq 0) \approx 1 - \mathbb{P}(\text{Pois}(\lambda) = 0) = 1 - e^{-\lambda}$$

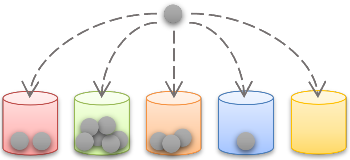
Review: Balls-and-Bins Model

Theorem

The maximum load in n bins with n balls is less than $\frac{3\ln n}{\ln\ln n}$ with high probability

$$\mathbb{P}\left(\max_{i=1,\ldots,n} X_i \geq \frac{3\ln n}{\ln\ln n}\right) \leq \frac{1}{n}$$

- Randomly putting balls into bins will result in load unbalance – the maximum load is $\frac{3\ln n}{\ln\ln n}$



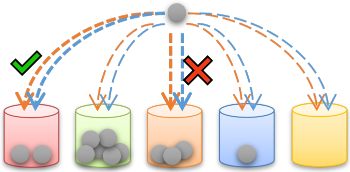
Power of Two-Choices Model

Definition (Power of Two-Choices Model)

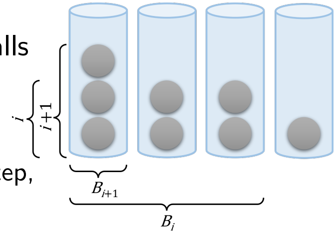
Throw m balls into n bins in the following manner:

- Each ball chooses two bins (both) uniformly at random
- The ball is put into the bin that has lesser number of balls (at that time)
- If both bins have identical number of balls, then put the ball in either of the bins

- Consider n bins with n balls
- What is to the maximum load of power of two-choices?
- It is exponentially better than one random choice – the maximum load is at most $\log\log n + O(1)$
- If $d \geq 2$ random choices, then not much improvement – the maximum load is at most $\frac{\log\log n}{\log d} + O(1)$



Power of Two-Choices Model: Intuition



- Let B_i be the number of bins with $\geq i$ balls after throwing n balls
- Goal: Try to bound B_{i+1} given B_i, and recursively given B_2
- Evidently, $B_2 \leq \frac{n}{2}$; otherwise there would be more than n balls
  - Given that $B_2 \leq \frac{n}{2}$ after throwing n balls, at any intermediate step, there will certainly be at most $\frac{n}{2}$ bins with at least 2 balls
- To bound B_3, consider the 3rd (or 4th, etc) ball in a bin with at least two balls
  - By power of two-choices, it must be the case that both of its choices had at least 2 balls
  - Hence, the probability that a ball becomes the 3rd ball in a bin is at most $(\frac{B_2}{n})^2$
  - Since $B_2 \leq \frac{n}{2}$, each ball will become the 3rd ball in a bin with probability at most $(\frac{1}{2})^2 = \frac{1}{4}$
    - ★ Hence, the expected number of bins with at least 3 balls: $\mathbb{E}[B_3] = \frac{n}{4}$
  - For simplicity, let's approximate that the events of different balls becoming a "3rd ball in a bin" to be independent
    - ★ For independent events, by Chernoff bound, we expect $B_3 \leq \frac{n}{4} + \mathrm{o}(n)$ with high probability

Power of Two-Choices Model: Intuition

- Let's ignore the $o(n)$ term, and continue the argument
- Assume at most $B_3 \le \frac{n}{4}$ bins have 3 or more balls. To bound B_4, consider the 4th (or 5th, etc) ball in a bin with at least three balls
  - By power of two-choices, it must be the case that both of its choices had at least 3 balls
  - Hence, the probability that a ball becomes the 4th ball in a bin is at most $(\frac{B_3}{n})^2 \le (\frac{1}{4})^2$, yielding that $\mathbb{E}[B_4] \approx \frac{n}{2^4}$, and we expect $B_4 \le \frac{n}{2^4} + o(n)$ with high probability
- In general, ignoring the $o(n)$ term, if $\frac{B_i}{n} \le c$, then we expect $\frac{B_{i+1}}{n} \le \left(\frac{B_i}{n}\right)^2 \le c^2$
  - Suppose $B_i \ll \sqrt{n}$, then for each ball, the probability it is the $(i+1)$-st ball in a bin will be at most $\left(\frac{1}{\sqrt{n}}\right)^2 < \frac{1}{n}$, and by a union bound, $B_{i+1} < 1$ with reasonable probability
- Therefore, with $\frac{B_{i+1}}{n} \le \left(\frac{B_i}{n}\right)^2$, we have

$$B_2 \le \frac{n}{2}, \; B_3 \le \frac{n}{2^2}, \; B_4 \le \frac{n}{2^4}, \; \ldots, \; B_i \le \frac{n}{2^{2^{i-2}}}$$

This implies $B_i < 1$ when $2^{2^{i-2}} > n$, which is precisely when $i > 2 + \log \log n$

Power of Two-Choices Model

Theorem

The maximum load in n bins with n balls with power of two-choices is less than $\log\log n + O(1)$ with high probability

$$\mathbb{P}\Big(\max_{i=1,\dots,n} X_i \geq \log\log n + O(1)\Big) \;\leq\; o\Big(\frac{1}{n}\Big)$$

Proof:

- Let B_i be the number of bins with at least i balls after throwing n balls, and $B_{i,j}$ be the number of bins with at least i balls after throwing j balls. Note that

$$0 = B_{i,0} \leq B_{i,1} \leq B_{i,2} \leq \dots \leq B_{i,n} = B_i$$

- Let $\text{Height}(i)$ be the height of ball i in its bin, e.g., if a bin has 3 balls: 2-nd, 5-th, 7-th balls, then $\text{Height}(2) = 1, \text{Height}(5) = 2, \text{Height}(7) = 3$

Power of Two-Choices Model

Proof (Cont.):

- Let us bound B_i, the number of bins with at least i balls, for $i \geq 6$ using induction
- Let $\beta_6 = \frac{n}{2e} > \frac{n}{6}$ and $\beta_{i+1} = e \cdot \frac{\beta_i^2}{n}$. Note that $\beta_{6+i} = \frac{n}{2^{2^i} e}$
- Let E_i be the event that $B_i \leq \beta_i$
- Since the number of bins with at least 6 balls is at most $\frac{n}{6}$, $\mathbb{P}(E_6) = 1$ (i.e. $B_6 \leq \beta_6$)
- Let $Y_j(i+1)$ be the indicator random variable that is 1 if Height$(j) \geq i+1$ and $B_{i,j-1} \leq \beta_i$ after $j-1$ balls are thrown
- Let Z_j be the bin of the j-th ball
- Suppose we have already thrown $j-1$ balls. In order for the j-th ball to have height at least $i+1$, both of its selected bins must have at least i balls. Hence,

$$\mathbb{P}\Big(Y_j(i+1) = 1 \ \mid \ Z_1, ..., Z_{j-1}\Big) \leq \left(\frac{B_{i,j-1}}{n}\right)^2 \leq \left(\frac{\beta_i}{n}\right)^2$$

- Let us define conditional probability and mention a few useful results ...

Conditional Probability

Definition (Conditional Probability)

For two events E_1 and E_2, the conditional probability of E_1 conditional on E_2 is defined as

$$\mathbb{P}(E_1 \mid E_2) \triangleq \frac{\mathbb{P}(E_1 \cap E_2)}{\mathbb{P}(E_2)}$$

If E_1 and E_2 are independent, then $\mathbb{P}(E_1 \mid E_2) = \mathbb{P}(E_1)$ and $\mathbb{P}(E_2 \mid E_1) = \mathbb{P}(E_2)$

Example (Conditional Probability)

- Consider an experiment of drawing two cards from a deck. The sample space is

$$\Omega = \{\mathtt{A}\diamondsuit\,\mathtt{A}\clubsuit, \mathtt{A}\diamondsuit\,\mathtt{A}\heartsuit, \mathtt{A}\diamondsuit\,\mathtt{A}\spadesuit, \mathtt{A}\clubsuit\,\mathtt{A}\heartsuit, \mathtt{A}\clubsuit\,\mathtt{A}\spadesuit, \mathtt{A}\heartsuit\,\mathtt{A}\spadesuit, \mathtt{A}\diamondsuit\,\mathtt{2}\diamondsuit, \mathtt{A}\diamondsuit\,\mathtt{2}\clubsuit, \ldots, \mathtt{K}\heartsuit\,\mathtt{K}\spadesuit\}$$

- Let $E_1^{\mathtt{A}}$ be the event that the first card is $\mathtt{A}$ and E_{pair} be that the two cards are a pair
- $\mathbb{P}(E_{\text{pair}} \mid E_1^{\mathtt{A}}) = \frac{6}{2\cdot 13\cdot 4\cdot(13\cdot 4-1)} \cdot 13 = \frac{3}{4\cdot(13\cdot 4-1)}$

Power of Two-Choices Model

Lemma

Let $Y_1, ..., Y_n$ be a set of binary random variables, and let $Z_1, ..., Z_n$ be a set of random variables such that Y_j depends on the outcomes of $Z_1, ..., Z_j$. Let S_n be a binomial random variable $\mathsf{BIN}(n, p)$*. If $\mathbb{P}(Y_j = 1 \mid Z_1, ..., Z_{j-1}) \leq p$ for all j, then for any c*

$$\mathbb{P}\Big(\sum_{j=1}^{n} Y_j \geq c\Big) \leq \mathbb{P}\Big(S_n \geq c\Big)$$

- If we consider Y_j one at a time, then each Y_j is less likely to take on the value 1 than an independent Bernoulli trial with success probability p, regardless of the outcome of Z_j
- The proof then follows by a mathematical induction

Theorem (Chernoff Bound)

Let S_n be $\mathsf{BIN}(n, p)$*. The tail probability is bounded by $\mathbb{P}(S_n \geq enp) \leq e^{-np}$*

Proof (Cont.):

- For $\frac{{\beta_i}^2}{n} \geq 2\ln n$:
  - Let S_n be a binomial random variable BIN$\left(n, \left(\frac{\beta_i}{n}\right)^2\right)$
  - By the above lemma, we have

$$\begin{aligned}\mathbb{P}(\neg E_{i+1}|E_i) &= \mathbb{P}\Big(\sum_{j=1}^{n} Y_j(i+1) \geq \beta_{i+1} \mid E_i\Big) = \frac{\mathbb{P}\big(\sum_{j=1}^{n} Y_j(i+1) \geq \beta_{i+1}\big)}{\mathbb{P}(E_i)} \\ &\leq \frac{\mathbb{P}\big(S_n \geq \beta_{i+1}\big)}{\mathbb{P}(E_i)} \leq \frac{1}{n^2\mathbb{P}(E_i)}\end{aligned}$$

    - ★ The last inequality is by Chernoff bound: $\mathbb{P}\big(S_n \geq \beta_{i+1} = e \cdot \frac{\beta_i^2}{n}\big) \leq e^{-\frac{{\beta_i}^2}{n}} \leq \frac{1}{n^2}$
  - Note that

$$\mathbb{P}(\neg E_{i+1}) = \mathbb{P}(\neg E_{i+1}|E_i)\cdot\mathbb{P}(E_i) + \mathbb{P}(\neg E_{i+1}|\neg E_i)\cdot\mathbb{P}(\neg E_i) \leq \mathbb{P}(\neg E_{i+1}|E_i)\cdot\mathbb{P}(E_i) + \mathbb{P}(\neg E_i)$$

  - Hence, E_{i+1} occurs with high probability, $\mathbb{P}(\neg E_{i+1}) \leq \frac{1}{n^2} + \mathbb{P}(\neg E_i) \leq \mathbb{P}(\neg E_6) + \frac{i-6}{n^2} \to 0$

Power of Two-Choices Model

Proof (Cont.):

- Note that β_i is decreasing. Let $i^* = \log\log n + 6$, such that $\beta_{i^*} = \frac{1}{2^{2^{i^*-6}}e} < 1$. We conclude E_{i^*} (i.e. $B_{i^*} \le \beta_{i^*} < 1$) occurs with high probability

* For $\beta_{i^*}^2 \le 2n\ln n$ (i.e., $\left(\frac{\beta_{i^*}}{n}\right)^2 \le \frac{2\ln n}{n}$):
  - We first show by Chernoff bound,

$$\mathbb{P}\Big(\sum_{j=1}^{n} Y_j(i^*+1) \ge 6\log n \mid E_{i^*}\Big) \le \frac{\mathbb{P}\big(\mathsf{BIN}(n, \left(\frac{\beta_{i^*}}{n}\right)^2) \ge 6\log n\big)}{\mathbb{P}(E_{i^*})} \le \frac{1}{n^2\mathbb{P}(E_{i^*})}$$

    Thus, there are at most $6\log n$ balls at height i^*+1 with high probability
  - Then we show by Markov's inequality,

$$\mathbb{P}\Big(\sum_{j=1}^{n} Y_j(i^*+2) \ge 1 \mid \sum_{j=1}^{n} Y_j(i^*+1) \ge 6\log n\Big) \le \frac{\mathbb{P}\big(\mathsf{BIN}(n, (\frac{6\log n}{n})^2) \ge 1\big)}{\mathbb{P}(E_{i^*})} \le \frac{36\log^2 n}{n\mathbb{P}(E_{i^*})}$$

    Therefore, there is no ball at height i^*+2 with high probability

Power of Two-Choices: Scalable Load Balancing

- Power of two-choices is an effective strategy for scalable load balancing with multiple load balancers
- Classic load-balancing methods such as least connections work very well when you operate a single active load balancer which maintains a complete view of the state of the load-balanced nodes
- The power of two-choices approach is not as effective on a single load balancer, but it deftly avoids the bad-case "herd behavior" that can occur when you scale out to a number of independent load balancers
- This scenario is not just observed when you scale out in high-performance environments; it's also observed in environments where multiple proxies each load balance traffic to the same set of service instances

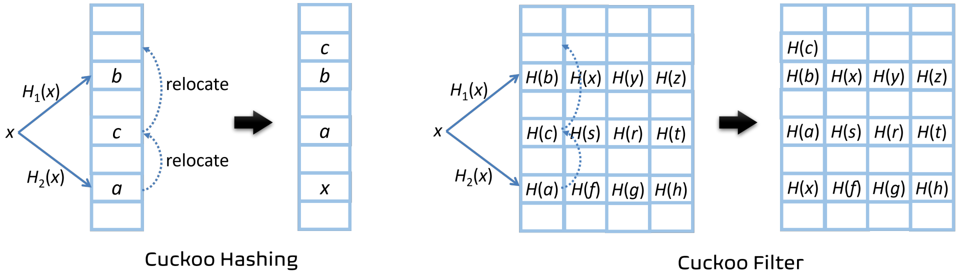
Power of Two Choices: 2-Way Chaining Hashtable

Definition (2-Way Chaining)

2-Way chaining technique in hashtable uses two random hash functions:

- The two hash functions define two possible options in the table for each item
- The item is inserted to the location that is the least full of the two options at the time of insertion
- Items in each entry of the table are stored in a linked list

- In the worst case, a lookup in a chained hashtable takes $\mathsf{O}(n)$.
  - Happens when all items are added into the same bin (unlikely in practice)
- In the expected worst-case, a lookup takes $\mathsf{O}(\log n / \log\log n)$ by balls-and-bins model
- In a 2-way chaining hashtable, a lookup takes $2\log\log n + \mathsf{O}(1)$ in the expected worst-case by the power of two-choices



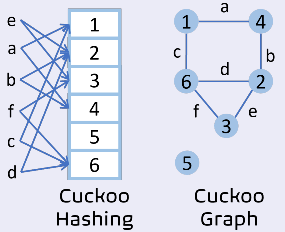
- Cuckoo Hashing: 2-way chaining hashtable. Each item maps to 2 locations. Relocate the occupied items to alternate locations if all hashed locations are occupied
- Cuckoo Filter: Uses a n-way chaining hashtable based on cuckoo hashing to store the fingerprints of items. Every bin of the hashtable can store up to multiple fingerprints

Cuckoo Hashing

Definition (Cuckoo Graph)

Cuckoo graph is a derived from a cuckoo hashtable as follows:

- Each table slot is a node
- Each item is an edge
- Edges link the slots where each item can be
- Each insertion introduces a new edge into the graph



- An insertion in a cuckoo hashtable traces a path through the cuckoo graph
- An insertion succeeds, if and only if the connected component containing the inserted value contains at most one cycle
- If x is inserted into a cuckoo hashtable, the insertion fails if the connected component containing x has two or more cycles

Cuckoo Hashing

Theorem (Cycle in Cuckoo Graph)

Consider a cuckoo graph with n nodes and $m = (1-\epsilon)\frac{n}{2}$ edges
For any constant $\epsilon > 0$, with high probability all the connected components in the graph are either single vertices, trees, or having a single cycle

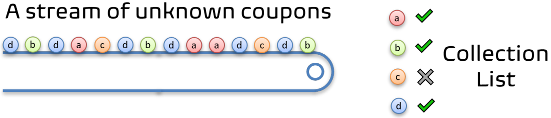
- All connected components with more than one node in the cuckoo graph are either trees or having a single cycle
- What is the expected size of a connected component in the cuckoo graph?

Theorem (Connected Component in Cuckoo Graph)

Consider a cuckoo graph with n nodes and $m = (1-\epsilon)\frac{n}{2}$ edges for any constant $\epsilon > 0$

- *With high probability the largest connected component has size $O(\log n)$*
- *The expected size of a connected component is $O(1)$*

Coupon Collector's Problem



Definition (Coupon Collector's Problem)

- There are n different coupons
- Goal: Collect all n coupons from a sequence of independent draws
  - Each time a random coupon is drawn; each coupon appears with a uniform probability $\frac{1}{n}$
  - Sometime, a coupon drawn may have appeared before
- Let X be the number of draws required to collect all n coupons: $X = \sum_{i=1}^{n} X_i$, where X_i is number of draws to collect the i-th different coupon that has not been collected before

Coupon Collector's Problem

Theorem

- *Let X be the number of draws required to collect all n types of coupons. Then, for any constant c,*

$$\lim_{n\to\infty} \mathbb{P}(X > n\ln n + cn) = 1 - e^{-e^{-c}}$$

Basic Ideas:

- Based on balls-and-bins model: balls $=$ draws, bins $=$ types of coupons
- Use Poisson approximation to model the number of balls throwing into bins, such that each bin has at least one ball, or equivalently no bin is empty
- If the expected total number of balls is $m = n\ln n + cn$, then the expected number of balls per bins is $\ln n + c$
- Find the probability of an empty bin by Poisson approximation

References

Reference Materials

- Probability and Computing (Mitzenmacher, Upfal), 2nd ed, Cambridge University Press
  - Chapter 4.6: Packet routing in sparse networks
  - Chapter 17: Balanced allocations and cuckoo hashing

Recommended Materials

- *Designing a Predictable Internet Backbone with Valiant Load-Balancing* (Rui, McKeown), IWQoS 2005