Probability in Computing

LECTURE 5: MORE APPLICATIONS WITH PROBABILISTIC ANALYSIS, BINS AND BALLS

Agenda

- Review: Coupon Collector's problem and Packet Sampling
- Analysis of Quick-Sort
- Birthday Paradox and applications
- The Bins and Balls Model

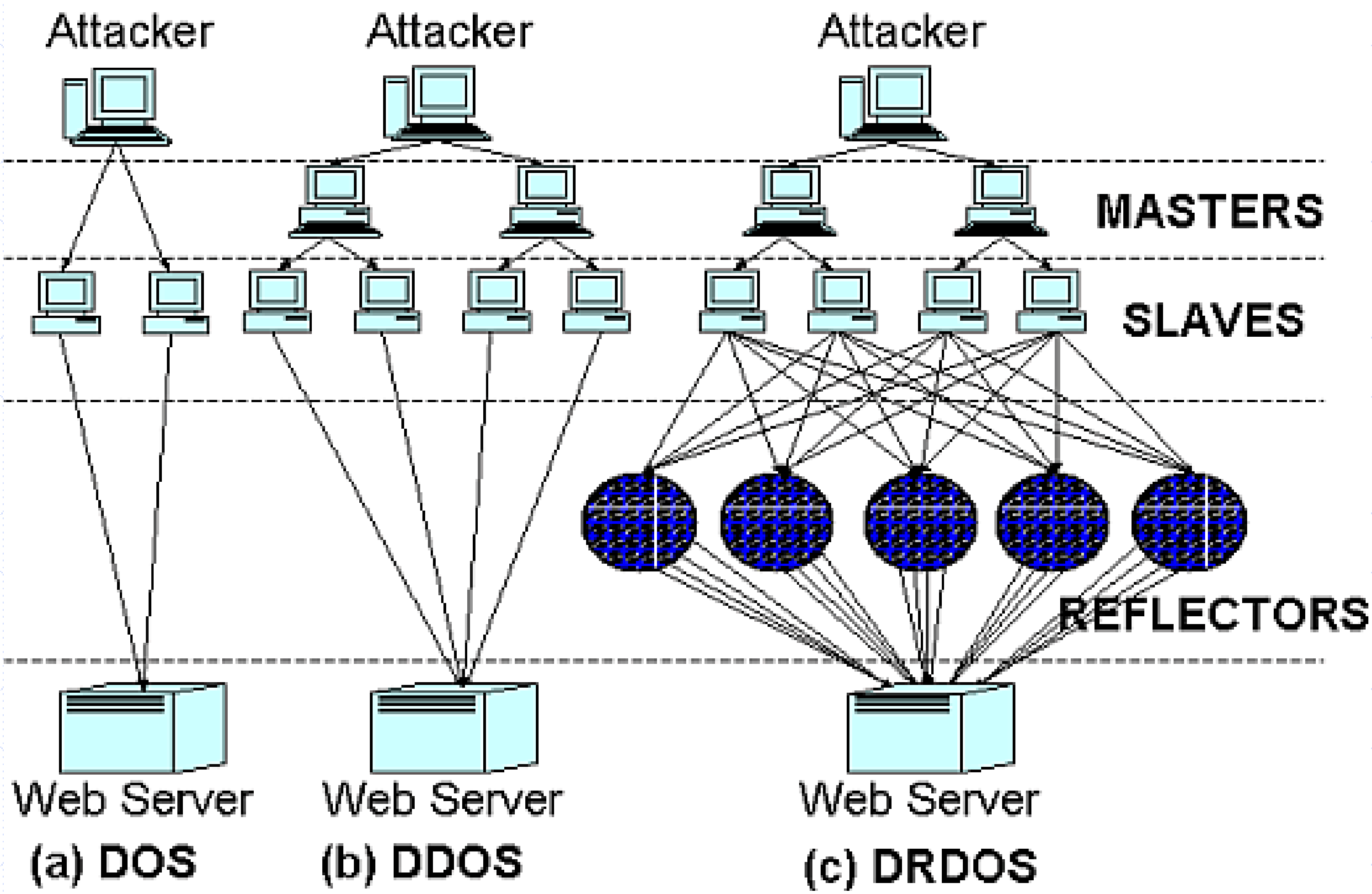
Coupon Collector Problem

- Problem: Suppose that each box of cereal contains one of n different coupons. Once you obtain one of every type of coupon, you can send in for a prize.
- Question: How many boxes of cereal must you buy before obtaining at least one of every type of coupon.
- Let X be the number of boxes bought until at least one of every type of coupon is obtained.
- $E[X] = nH(n) = n\ln n$

Application: Packet Sampling

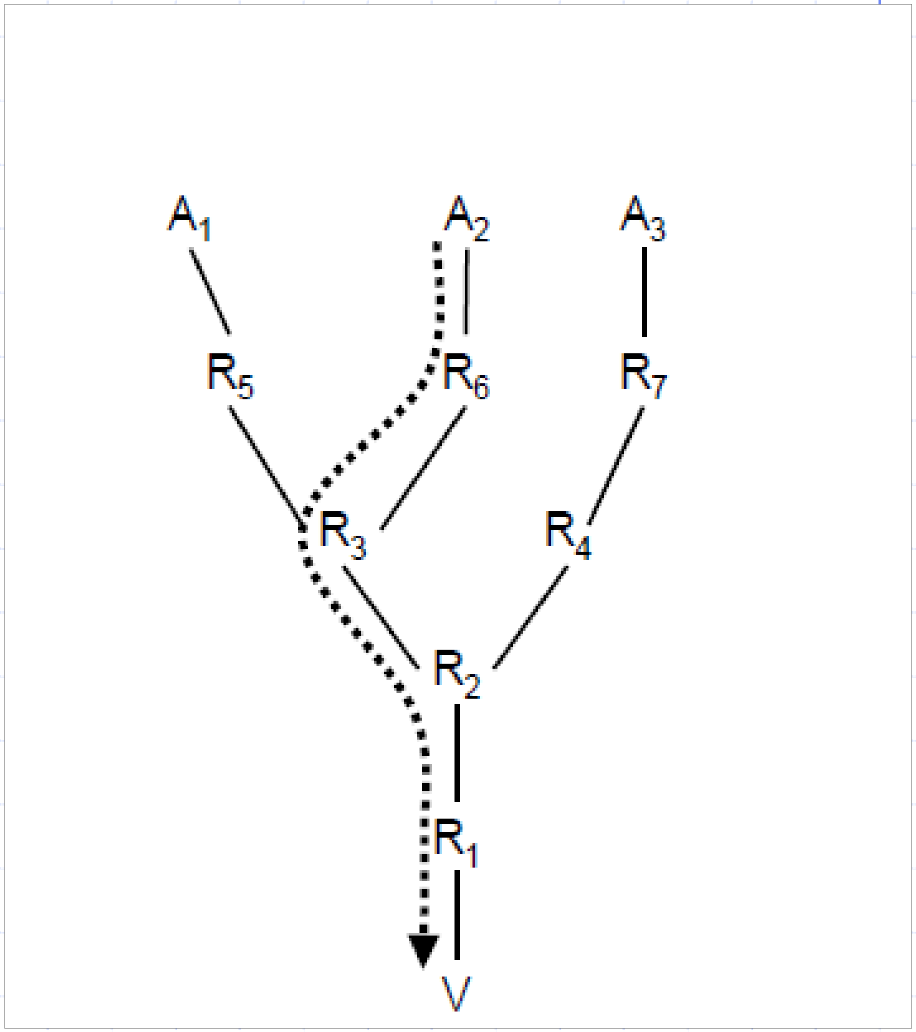
- Sampling packets on a router with probability p
  - The number of packets transmitted after the last sampled packet until and including the next sampled packet is geometrically distributed.
- From the point of destination host, determining all the routers on the path is like a coupon collector's problem.
- If there's n routers, then the expected number of packets arrived before destination host knows all of the routers on the path = nln(n).

DoS attack

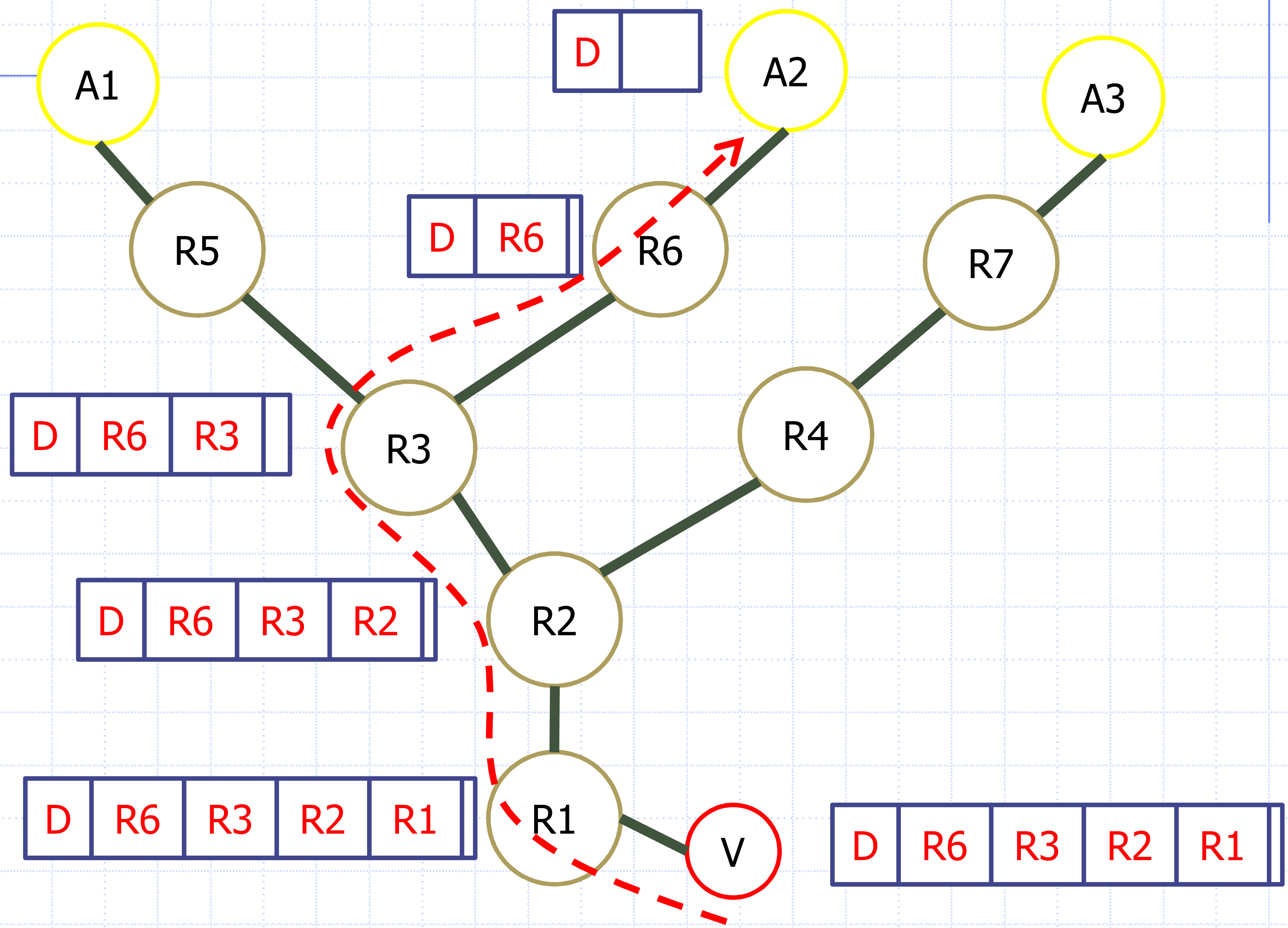

Attacker
Attacker
Attacker
MASTERS
SLAVES
REFLECTORS
Web Server
Web Server
Web Server
(a) DOS
(b) DDOS
(c) DRDOS

IP traceback

- Marking and Reconstruction
  - Node append vs. node sampling

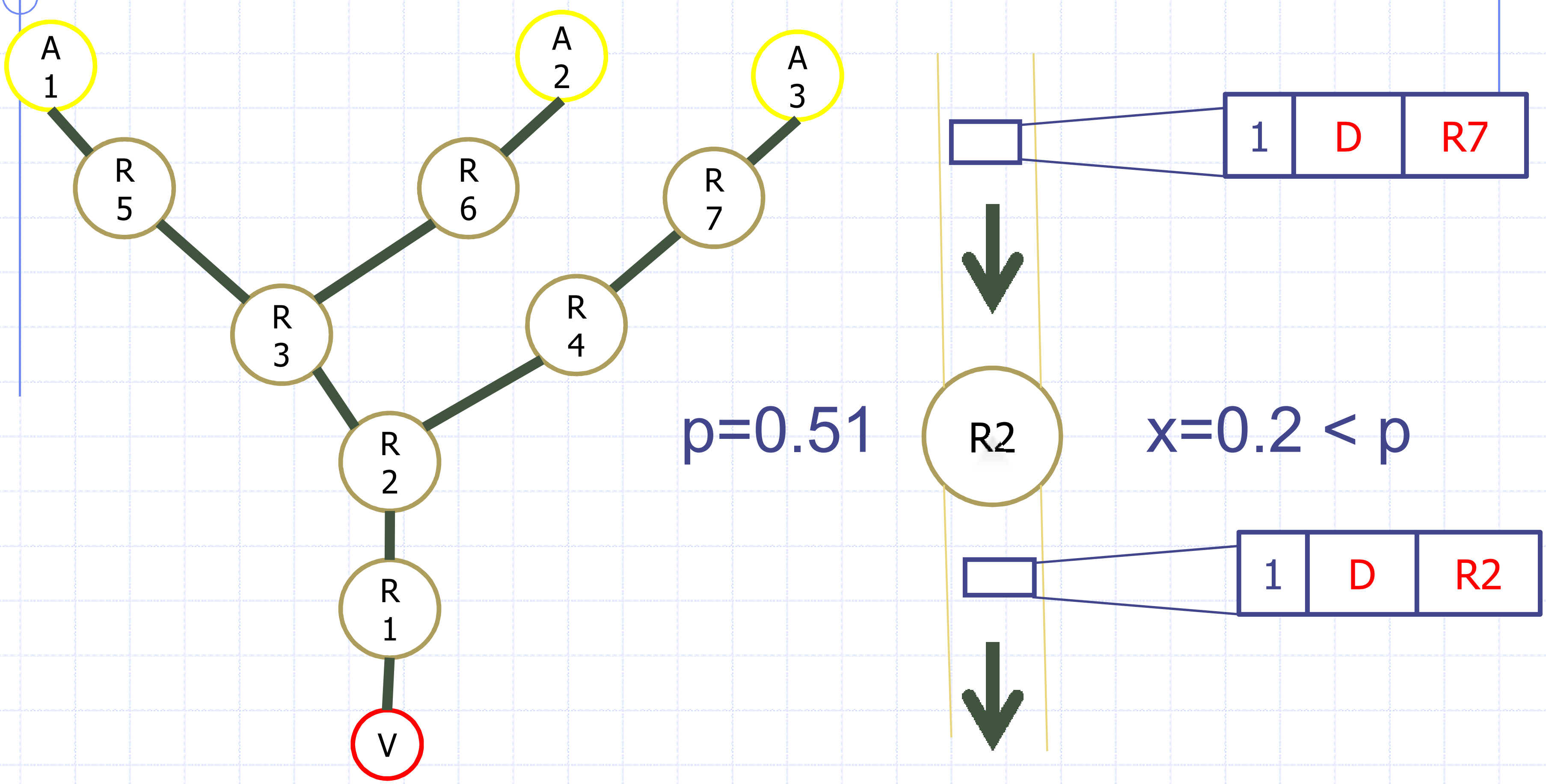


Node apend



D
A2
A1
A3
D
R6
R5
R6
R7
D
R6
R3
R3
R4
D
R6
R3
R2
R2
D
R6
R3
R2
R1
R1
V
D
R6
R3
R2
R1

Node Sampling



Expected Run-Time of QuickSort

Quicksort Algorithm:

Input: A list $S = \{x_1, \ldots, x_n\}$ of n distinct elements over a totally ordered universe.

Output: The elements of S in sorted order.

1. If S has one or zero elements, return S. Otherwise continue.
2. Choose an element of S as a pivot; call it x.
3. Compare every other element of S to x in order to divide the other elements into two sublists:
   - **(a)** S_1 has all the elements of S that are less than x;
   - **(b)** S_2 has all those that are greater than x.
4. Use Quicksort to sort S_1 and S_2.
5. Return the list S_1, x, S_2.

Analysis

- Worst-case: n^2.
- Depends on how we choose the pivot.
- Good pivot (divide the list in two nearly equal length sub-lists) vs. Bad pivot.
- In case of good pivot -> nlg(n). [by solving recurrence]

- If we choose pivot point randomly, we will have a randomized version of QuickSort.

Analysis

- X_{ij} be a random variable that
  - Takes value 1 if y_i and y_j are compared with each other
  - 0 if they are not compared.

- $E[X] = \sum\sum E[X_{ij}]$
- $E[X_{ij}] = 2/\ (j-i+1)$
  - Consider when the set $Y_{ij}=\{y_i, y_{i+1}, \ldots, y_j\}$ is "touched" by a pivot the first time. If this pivot is either y_i or y_j then the two will be compared, otherwise Never – a (S_1,S_2) split

- Using $k = j-i+1$, we can compute $E[X] = 2n\ln(n)$

Detail analysis

$$\begin{aligned}
\mathbf{E}[X] &= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2}{j-i+1} \\
&= \sum_{i=1}^{n-1} \sum_{k=2}^{n-i+1} \frac{2}{k} \\
&= \sum_{k=2}^{n} \sum_{i=1}^{n+1-k} \frac{2}{k} \\
&= \sum_{k=2}^{n} (n+1-k) \frac{2}{k} \\
&= \left((n+1) \sum_{k=2}^{n} \frac{2}{k} \right) - 2(n-1) \\
&= (2n+2) \sum_{k=1}^{n} \frac{1}{k} - 4n.
\end{aligned}$$

Birthday "Paradox"

What is the probability that two persons in a room of 30 have the same birthday?

Birthday Paradox

- Ways to assign k different birthdays without duplicates:

$$N = 365 * 364 * \ldots * (365 - k + 1)$$
$$= 365! / (365 - k)!$$

- Ways to assign k different birthdays with possible duplicates:

$$D = 365 * 365 * \ldots * 365 = 365^k$$

Birthday "Paradox"

Assuming real birthdays assigned randomly:

N/D = probability there are no duplicates

1 - N/D = probability there is a duplicate

$$= 1 - 365! \,/\, ((365 - k)!(365)^k)$$

Generalizing Birthdays

$$P(n, k) = 1 - n!/(n-k)!n^k$$

Given k random selections from n possible values, $P(n, k)$ gives the probability that there is at least 1 duplicate.

Birthday Probabilities

P(no two match) = 1 – P(all are different)

P(2 chosen from N are different)

$= 1 - 1/N$

P(3 are all different)

$= (1 - 1/N)(1 - 2/N)$

P(n trials are all different)

$= (1 - 1/N)(1 - 2/N) \ldots (1 - (n - 1)/N)$

ln (P)

$= \ln (1 - 1/N) + \ln (1 - 2/N) + \ldots \ln (1 - (k - 1)/N)$

Happy Birthday Bob!

$\ln(P) = \ln(1 - 1/N) + \ldots + \ln(1 - (k-1)/N)$

For $0 < x < 1$: $\ln(1 - x) \le x$

$$\ln(P) \le -(1/N + 2/N + \ldots + (n-1)/N)$$

Gauss says:

$$1 + 2 + 3 + 4 + \ldots + (n-1) + n = ½\, n\,(n+1)$$

So,

$$\ln(P) \le ½\,(k\text{-}1)\,k/N$$

$$P \le e^{½\,(k\text{-}1)k\,/\,N}$$

Probability of match $\ge 1 - e^{½\,(k\text{-}1)k\,/\,N}$

Applying Birthdays

$$P(n, k) > 1 - e^{-k*(k-1)/2n}$$

- For $n = 365$, $k = 20$:

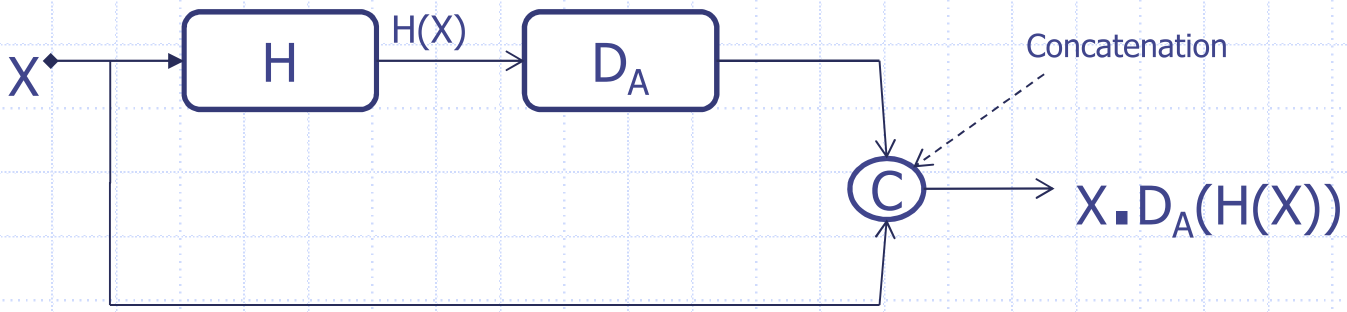
$$P(365, 20) > 1 - e^{-20*(19)/2*365}$$
$$P(365, 20) > .4058$$

- For $n = 2^{64}$, $k = 2^{32}$: $P(2^{64}, 2^{32}) > .39$
- For $n = 2^{64}$, $k = 2^{33}$: $P(2^{64}, 2^{33}) > .86$
- For $n = 2^{64}$, $k = 2^{34}$: $P(2^{64}, 2^{34}) > .9996$
- Application: Digital Signatures

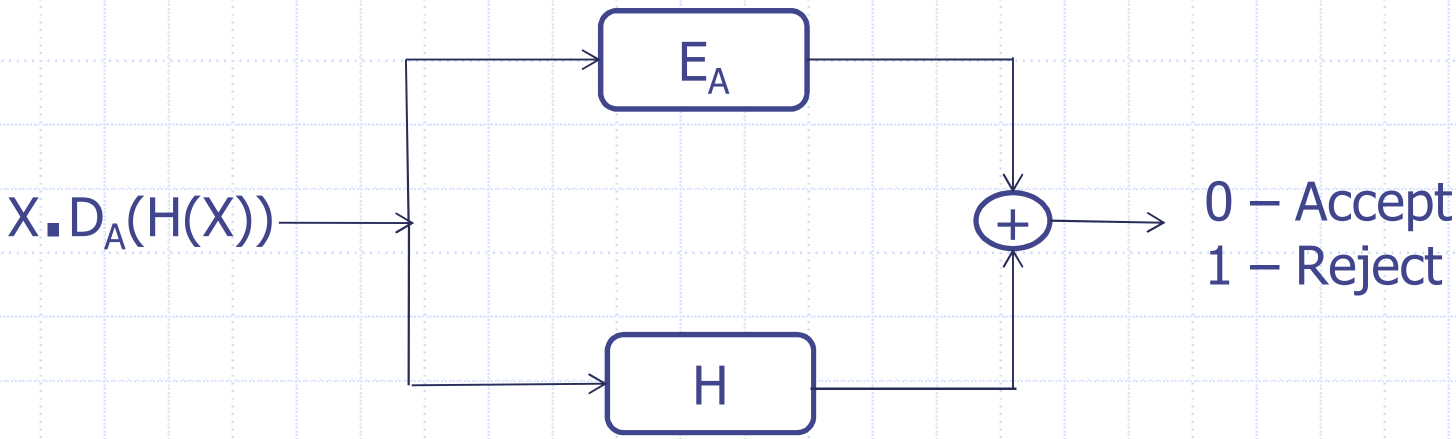
Digital Signature Scheme: Using Hash Functions

- A hash function H maps a message of variable length n bits to a fingerprint of fixed length m bits, with $m < n$.
  - This hash value is also called a digest (of the original message).
  - Since $n>m$, there exist many X which are map to the same digest ➔ collision.

DS schemes with hash functions



Signature Generator



Signature Verifier

Main properties

Given a hash function $H: X \rightarrow Y$

- Long message → short, fixed-length hash
- One-way property: given $y \in Y$

  it is computationally infeasible to find a value $x \in X$ s.t. $H(x) = y$
- Collision resistance (collision-free)

  it is computationally infeasible to find any two distinct values $x', x \in X$ s.t. $H(x') = H(x)$

  - This property prevent against signature forgery

Collisions

- Avoiding collisions is theoretically impossible
  - Dirichlet principle: n+1 rabbits into n cages → at least 2 rabbits go to the same cage
  - This suggest exhaustive search: try |Y|+1 messages then must find a collision (H:X→Y)
- In practice
  - Choose |Y| large enough so exhaustive search is computational infeasible.
    - |Y| not too large or long signature and slow process
  - However, collision-freeness is still hard

Birthday attack

- Can hash values be of 64 bits?
  - Look good, initially, since a space of size 2^{64} is too large to do exhaustive search or compute that many hash values
  - However a birthday attack can easily break a DS with a 64-bit hash function
    - In fact, the attacker only need to create a bunch of 2^{32} messages and then launch the attack with reasonably high probability for success.

How is the attack

- Goal: given H, find x, x′ such that H(x)=H(x′)
- Algorithm:
  - pick a random set S of q values in X
  - for each $x \in S$, computes $h_x = H(x)$
  - if $h_x = h_{x'}$ for some $x' \neq x$ then collision found: (x,x′), else fail
- The average success probability is

  $\varepsilon = 1-\exp(^{q(q-1)}/_{2|Y|})$
  - Suppose Y has size 2^m, choose $q \approx 2^{m/2}$ then ε is almost 0.5!

Balls into Bins

- We have m balls that are thrown into n bins, with the location of each ball chosen independently and uniformly at random from n possibilities.
- What does the distribution of the balls into the bins look like
  - "Birthday paradox" question: is there a bin with at least 2 balls
  - How many of the bins are empty?
  - How many balls are in the fullest bin?

Answers to these questions give solutions to many problems in the design and analysis of algorithms

The maximum load

- When n balls are thrown independently and uniformly at random into n bins, the probability that the maximum load is more than 3 ln*n*/lnln*n* is at most 1/*n* for *n* sufficiently large.
  - By Union bound, Pr [bin 1 receives ≥ M balls] ≤ $\binom{n}{M}\left(\frac{1}{n}\right)^M$.
  - Note that:

$$\binom{n}{M}\left(\frac{1}{n}\right)^M \le \frac{1}{M!} \le \left(\frac{e}{M}\right)^M.$$

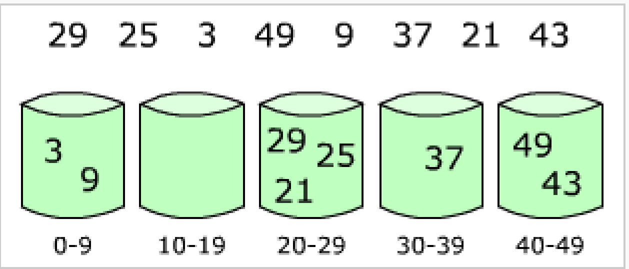
  - Now, using Union bound again, Pr [any ball receives ≥ M balls] is at most

$$n\left(\frac{e}{M}\right)^M \le n\left(\frac{e \ln\ln n}{3\ln n}\right)^{3\ln n/\ln\ln n}$$

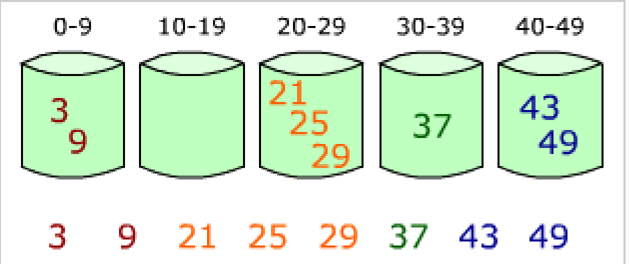
which is ≤ 1 / n

Application: Bucket Sort

- A sorting algorithm that breaks the Ω(nlogn) lower bound under certain input assumption
- Bucket sort works as follows:
  - Set up an array of initially empty "buckets."
  - Scatter: Go over the original array, putting each object in its bucket.
  - Sort each non-empty bucket.
  - Gather: Visit the buckets in order and put all elements back into the original array.



Elements are distributed among bins



Then, elements are sorted within each bin

- A set of $n = 2^m$ integers, randomly chosen from $[0,2^k), k \geq m$, can be sorted in expected time $O(n)$
  - Why: will analyze later!

The Poisson Distribution

- Consider m balls, n bins
  - Pr [a given bin is empty] = $\left(1-\frac{1}{n}\right)^m \approx e^{-m/n};$
  - Let X_j is a indicator r.v. that os 1 if bin j empty, 0 otherwise
  - Let X be a r.v. that represents # empty bins

$$\mathbf{E}[X] = \mathbf{E}\left[\sum_{i=1}^{n} X_i\right] = \sum_{i=1}^{n} \mathbf{E}[X_i] = n\left(1-\frac{1}{n}\right)^m \approx n e^{-m/n}.$$

  - Generalizing this argument, Pr [a given bin has r balls] =

$$\binom{m}{r}\left(\frac{1}{n}\right)^r\left(1-\frac{1}{n}\right)^{m-r} = \frac{1}{r!}\frac{m(m-1)\cdots(m-r+1)}{n^r}\left(1-\frac{1}{n}\right)^{m-r}.$$

  - Approximately, $p_r \approx \dfrac{e^{-m/n}(m/n)^r}{r!}$
  - So: **Definition 5.1:** *A discrete* Poisson random variable *X with parameter μ is given by the following probability distribution on $j = 0, 1, 2, \ldots$:*

$$\Pr(X = j) = \frac{e^{-\mu}\mu^j}{j!}.$$

Limit of the Binomial Distribution

We have shown that, when throwing m balls randomly into b bins, the probability p_r that a bin has r balls is approximately the Poisson distribution with mean m/b. In general, the Poisson distribution is the limit distribution of the binomial distribution with parameters n and p, when n is large and p is small. More precisely, we have the following limit result.

Theorem 5.5: *Let X_n be a binomial random variable with parameters n and p, where p is a function of n and $\lim_{n\to\infty} np = \lambda$ is a constant that is independent of n. Then, for any fixed k,*

$$\lim_{n\to\infty} \Pr(X_n = k) = \frac{e^{-\lambda}\lambda^k}{k!}.$$

This theorem directly applies to the balls-and-bins scenario. Consider the situation where there are m balls and b bins, where m is a function of b and $\lim_{n\to\infty} m/b = \lambda$. Let X_n be the number of balls in a specific bin. Then X_n is a binomial random variable with parameters m and $1/b$. Theorem 5.5 thus applies and says that

$$\lim_{n\to\infty} \Pr(X_n = r) = \frac{e^{-m/n}(m/n)^r}{r!},$$