Randomized switch scheduling algorithms

Balaji Prabhakar
Departments of EE and CS
Stanford University
Randomized algorithms

• Randomization is a method that can be used to simplify the implementation

• The main idea
  – base decisions upon a small, randomly chosen sample of the state/input, instead of the complete state/input
  
  – randomized algorithms are also robust to adversarial attacks (decisions depend on chance events)
An illustrative example

• Find the youngest person from a population of 1 billion

• Deterministic algorithm: linear search
  – has a complexity of 1 billion

• A randomized version: find the youngest of 30 randomly chosen people
  – has a complexity of 30

• Performance
  – linear search will find the absolute youngest person (rank = 1)
  – if R is the person found by randomized algorithm, we can say

\[
P(R \text{ has rank } < 100 \text{ million}) > 1 - \left( \frac{9}{10} \right)^{30} \approx 0.95
\]

➢ thus, we can say that the performance of the randomized algorithm is good with a high probability
Randomizing iterative schemes

- Often, we want to perform some operation iteratively

- Example: find the youngest person each year

- Say in 2007 you choose 30 people at random
  - and store the identity of the youngest person in memory
  - in 2008 you choose 29 new people at random
  - let R be the youngest person from these $29 + 1 = 30$ people

$$P(R \text{ has rank } < 100 \text{ million}) > 1 - \left( \frac{9}{10} \right)^{58}$$

$$P(R \text{ has rank } < 50 \text{ million}) > 1 - \left( \frac{9}{10} \right)^{30}$$
Randomized approximation to the Max Wt Matching algorithm

joint work with Paolo Giaccone and Devavrat Shah
Notation and definitions

- Arrivals: $A_{ij}(t)$, i.i.d. Bernoulli, $E(A_{ij}(t)\lambda_{i,j})$

$\lambda = [\lambda_{ij}]$ is admissible if: $\sum_k \lambda_{ik} < 1; \sum_k \lambda_{kj} < 1$, $\forall i, j$.

- $Q(t) = [Q_{ij}(t)]$ are backlogs at time $t$

$Q(t+1) = [Q(t) + A(t) - S(t)]^+$

- Scheduling problem: Given $Q(t)$, determine a matching, $S(t)$, of inputs and outputs to maximize throughput and minimize backlogs
Useful performance metrics

• Throughput
  - an algorithm is stable (or delivers 100% throughput) if for any admissible arrival, the average backlog is bounded; i.e.
    \[ \sup_t E[Q_{ij}(t)] < \infty, \text{ for every } i \text{ and } j. \]
    (equivalent to positive recurrence of Q(t))

• Minimize average backlogs or, equivalently, packet delays
Scheduling: Bipartite graph matching

Schedule or Matching
Scheduling algorithms

- Not stable

Practical Maximal Matchings

Max Wt Matching

Not stable

Max Size Matching

Not stable (McKeown–Ananthram–Walrand 96)

Stable

(Tassiulas–Ephremides 92, McKeown et. al. 96, Dai–Prabhakar 00)
The Maximum Weight Matching Algorithm

• MWM: performance
  – throughput: stable (Tassiulas–Ephremides 92; McKeown et al 96; Dai–Prabhakar 00)
  – backlogs: very low on average (Leonardi et al 01; Shah–Kopikare 02)

• MWM: implementation
  – has cubic worst-case complexity
    (approx. 27,000 iterations for a 30-port switch)
  – MWM algorithms involve backtracking:
    i.e. edges laid down in one iteration may be removed in a subsequent iteration
    ➢ algorithm not amenable to pipelining
Switch algorithms

Maximal matching
- Not stable

Max Size Matching
- Not stable

Max Wt Matching
- Stable and low backlogs

Better performance
Easier implementation
Randomized approximation to MWM

• Consider the following randomized approximation:
  At every time
  – sample d matchings independently and uniformly
  – use the heaviest of these d matchings to schedule packets

• Ideally we would like to use a small value of d. However,…

**Theorem.** Even with \( d = N \), this algorithm is not stable. In fact,
when \( d = N \), the throughput is at most \( 1 - \frac{1}{e} \approx 63\% \)
(Giaccone–Prabhakar–Shah 02)
Proof

- Let \( E_{ij} \) be the edge connecting input \( i \) to output \( j \), then

\[
P_{ij} = P(E_{ij} \in \text{ one of the } d \text{ randomly chosen matching})
= 1 - P(E_{ij} \notin \text{ any of } d \text{ randomly chosen matching})
= 1 - P(E_{ij} \notin \text{ one randomly chosen matching})^d
= 1 - (1 - \frac{1}{N})^d \leq 1 - (1 - \frac{1}{N})^N \text{ for } d \leq N
\leq 1 - \frac{1}{e} \approx 0.63.
\]

- Since the edge \( E_{ij} \) can be served only if it is chosen by at least one of the \( d \) matching, it follows that the throughput is at most 63%
Tassiulas’ algorithm

Previous matching $S(t-1)$

Random Matching $R(t)$

Next time

MAX

Current matching $S(t)$
Tassiulas’ algorithm

\[ S(t-1) \]
\[ W(S(t-1)) = 160 \]

\[ R(t) \]
\[ W(R(t)) = 150 \]
Performance of Tassiulas’ algorithm

**Theorem** (Tassiulas 98): The above scheme is stable under any admissible Bernoulli IID inputs.
Backlogs under Tassiulas’ algorithm
Reducing backlogs: the Merge operation

S(t-1)
W(S(t-1))=160

R(t)
W(R(t))=150

Merge

10
10
10
70
60

50
40
30
10
20

30 v/s 120
130 v/s 30
Reducing backlogs: the Merge operation

\[ W(S(t-1)) = 160 \]

\[ W(S(t)) = 250 \]

\[ W(R(t)) = 150 \]
Performance of Merge algorithm

**Theorem** (GPS): The Merge scheme is stable under any admissible Bernoulli IID inputs.
Use arrival information: Serena

\[ W(S(t-1)) = 209 \]

The arrival graph
Use arrival information: Serena

\[ S(t-1) \]
\[ W(S(t-1)) = 209 \]

The arrival graph
Use arrival information: Serena

\[ W(S(t-1)) = 209 \]

\[ W(S(t)) = 243 \]

\[ W = 121 \]
Performance of Serena algorithm

**Theorem (GPS):** The Serena algorithm is stable under any admissible Bernoulli IID inputs.
Backlogs under Serena

![Graph showing normalized load against mean IQ and length for Tassiulas, Merge, Serena, and MWM.](image)