

CS6811: Research Seminar on Reconfigurable Hardware

Lockwood : Spring 2002

Review of: Efficient Randomized Algorithms for Input-Queued Switch Scheduling

- Paper by:

- Devavrat Shah (Stanford University)
- Paolo Giaccone (Politecnico Di Torino)
- Balaji Prabhakar (Stanford University)

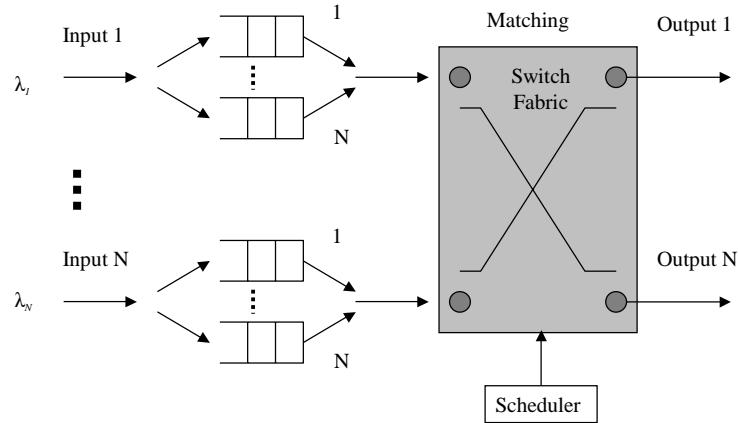
– Hot Interconnect, 2001

- Summarized by: Sarang Dharmapurikar

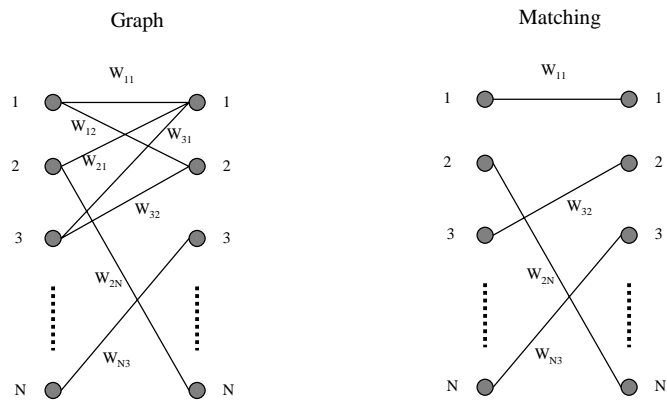
Style of Writing

- Explain the problem
- Survey the existing algorithms
- Develop the background for a new algorithm
- Explain the new algorithms
 - Dropped the proofs
- Performance comparison
 - Only software simulations

Input Queued Switches



Problem Model



Various solutions

- **Maximum Weight Matching (MWM)**
 - Maximize the sum of edges in a matching
 - $O(N^3)$ complex
 - delivers “up to” 100% throughput
- **Approximations**
 - iSLIP
 - Iterative Longest Queue First (iLQF)
 - Reservation with preemption and acknowledgement(RPA)
 - Matrix Unit Cell Scheduler
 - Complexity $O(N^2)$, can be reduced further in HW
 - Throughput bad in non-uniform traffic
 - Don't scale well with N

Background of Randomized Algorithms

- Use the matching at t for finding matching at $t+1$
 - Weights of the edges don't change much between two consecutive time slots
- Remember only few heavy edges at time t
 - The few heavy weight edges contain the bigger fraction of the total weight of matching
- Produce some randomized matchings to improve the previous matching

Simulation Settings

- **Switch**
 - $N = 32$, $Q_{ij} = 10000$ packets
 - The switch does not share the buffer
- **Input Traffic**
 - Uniform : $\lambda_{ij} = \rho/N$
 - Diagonal : $\lambda_{ii} = 2\rho/3$, $\lambda_{i|i+1|} = \rho/3$, $\lambda_{ij} = 0$ otherwise
 - Input i Sends traffic to output i and $i+1$ only
 - Log Diagonal : $\lambda_{ij} = 2\lambda_{i|i+1|}$, $\rho \lambda_{ij} = \rho$
 - Example : $\lambda_{ij} = 2^{N-j} \rho / (2^N - 1)$
 - Lesser traffic for farther ports

Basic Randomized Algorithms

- **Random I**
 - For every time, pick matching R uniformly and randomly from all possible $N!$ matching
 - Use R as the schedule
- **Random II**
 - Choose d matchings uniformly and randomly in each time slot.
 - Use the highest weighted of these matchings as the schedule

Random III

- Let $S(t)$ be the schedule used at time t .
- At time $t+1$, choose a matching $R(t+1)$ uniformly and randomly from the set of all $N!$ possible matchings.
- Let the schedule at time $t+1$, $S(t+1)$, be the heavier weighted of $S(t)$ and $R(t+1)$
 - Gives significantly better throughput than Random I and Random II
 - Large delays compared to MWM

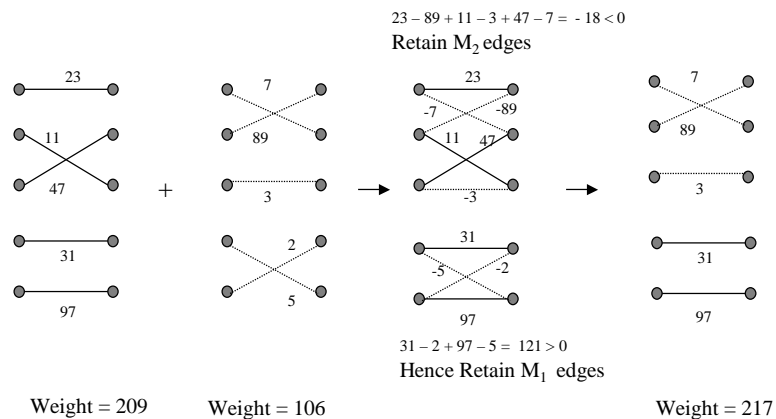
Random IV

- Observe that for non-uniform loading, just the few edges carry most of the weight of the randomly chosen matching
- $F_\eta(M)$ = minimum set of edges in M carrying at least an η fraction of the total weight of M . (Hence $0 < \eta \leq 1$)
- Let $S(t)$ be the matching at time t
- Compute $F_\eta[S(t)]$
- At time $t+1$, let $R(t+1)$ be the matching that first uses the edges in $F_\eta[S(t)]$. This leaves $N - F_\eta[S(t)]$ input/output nodes unmatched. $R(t+1)$ connects these unmatched input/output nodes using a randomly chosen matching
- Let $S(t+1)$ equal the heavier weighted of $R(t+1)$ and $S(t)$
- Improvement: Use m matchings from the past and r random matchings

Laura

- Retain M_1, M_2, \dots, M_S be the S distinct matchings from the past
- Obtain X_1, X_2, \dots, X_v matchings using the "Random" procedure
- Do a "Merge" on each pair $R_{ab} = \text{Merge}(M_a, X_b)$ for all $a = 1, 2, \dots, S$ and $b = 1, 2, \dots, v$ and thus produce SV number of new and better schedules
- Find the $M = \text{Max}(R_{ab})$
- Use M as the schedule
- Retain only highest S matchings of all the R_{ab} and use those as the M_1, M_2, \dots, M_S for the next iteration
- Runtime complexity $O(VIN \log_2 N + SVN)$

Merging



Serena

- Retain M_1, M_2, \dots, M_S be the S distinct matchings from the past
- Consider only those edges which had some arrival in previous time slot
 - The edges having more arrivals will have more weight
- Merge these edges with the past matching using *Arrival-Merge* procedure to get the higher weight matching

Conclusion

- Randomized Algorithms is a potential solution for distributed Input-Queuing switches
 - Lower runtime complexity
 - Scalable algorithms
- Performance can be enhanced with the knowledge of past
- Can be implemented in hardware
 - See "An Implementable Parallel Scheduler For Input-Queued Switches"