Performance Analysis of Scheduling Algorithms for Switches and Data Centers: Part I

Atilla Eryilmaz, Siva Theja Maguluri, and R. Srikant

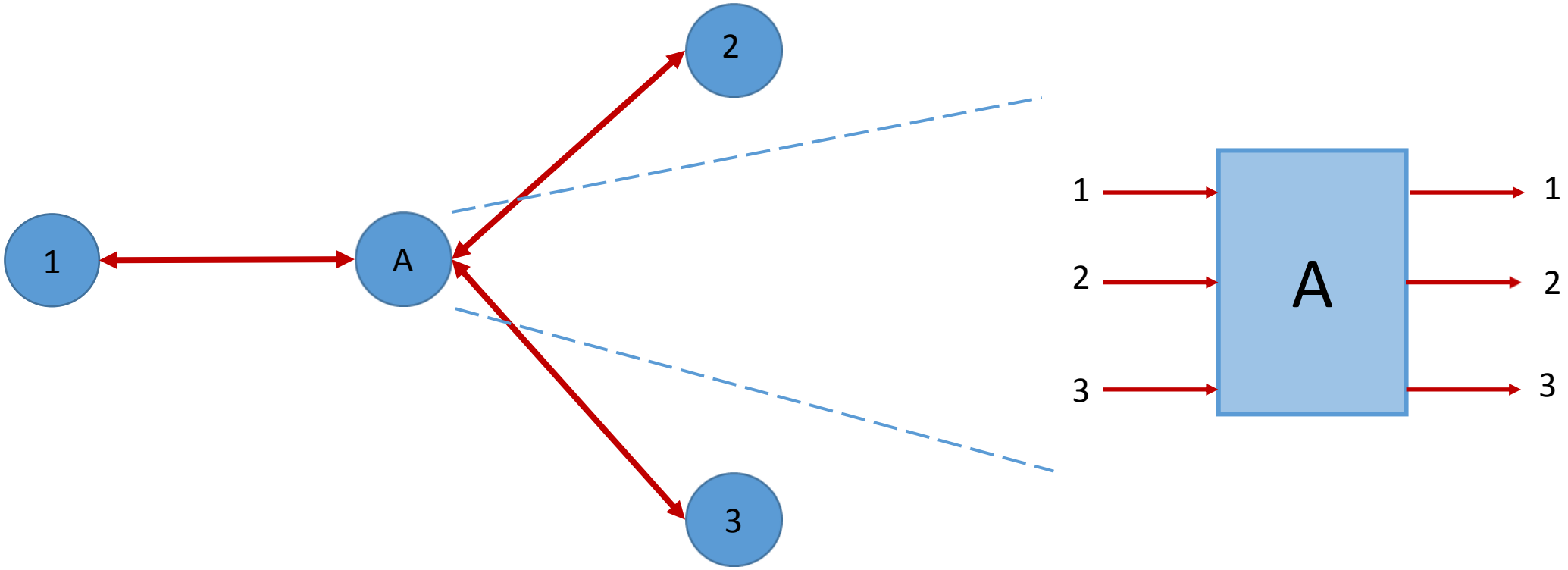
OSU, IBM TJ Watson, and UIUC

(Part I is a summary of prior work)

Outline

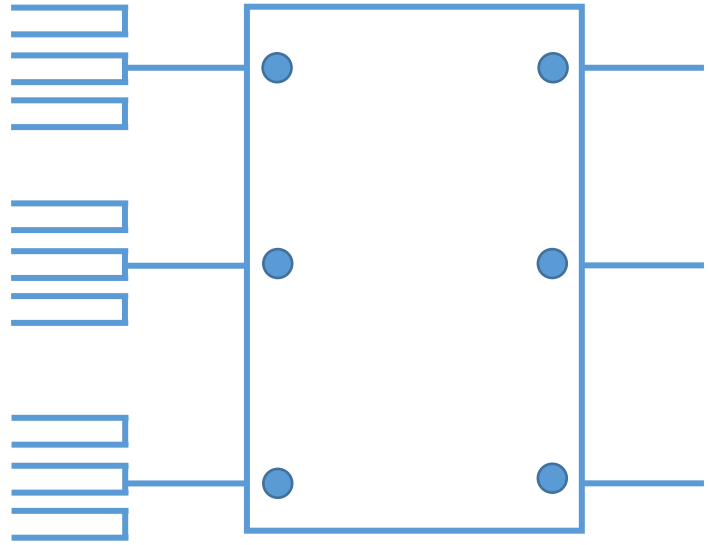
- Switch Scheduling: Where does the problem arise?
  - Why is it relevant today in the age of big data, i.e., the age of Facebook, Twitter, Google, Miscrosoft, etc.?
- MaxWeight matchings in bipartite graphs and stochastic stability
- A preview of Part II: how does the delay in a switch scale as the size of the system grows or the traffic intensity grows?

Internet Router/ Switch

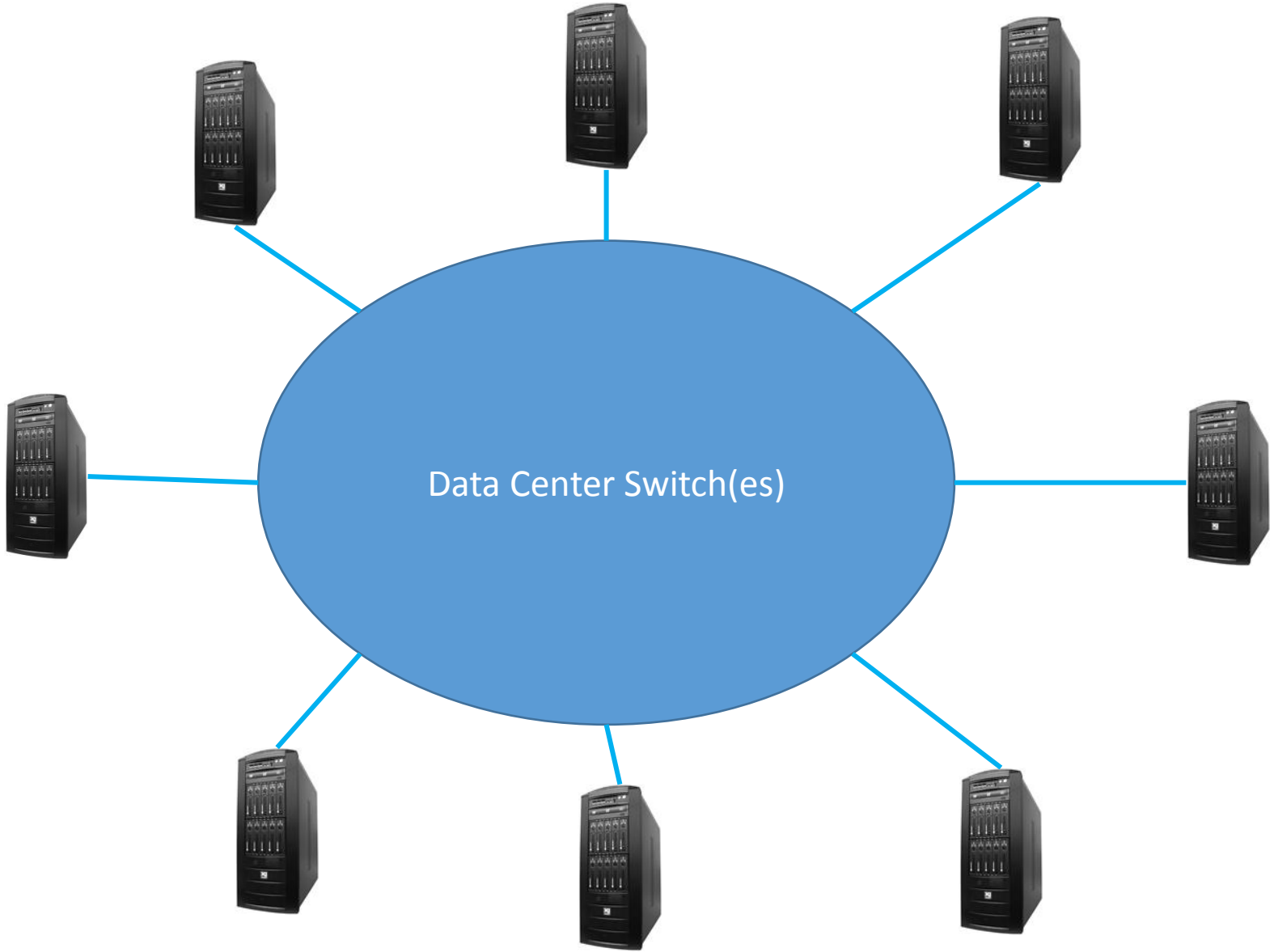


Input Queued Switch

- An abstraction to study internet switch
- n input ports and n output ports
- n queues at each input port
- Packets arrive to each queue according to some arrival process. In each time slot, at most one packet can be served from each queue
- One input port may be connected to only one output port in each time slot

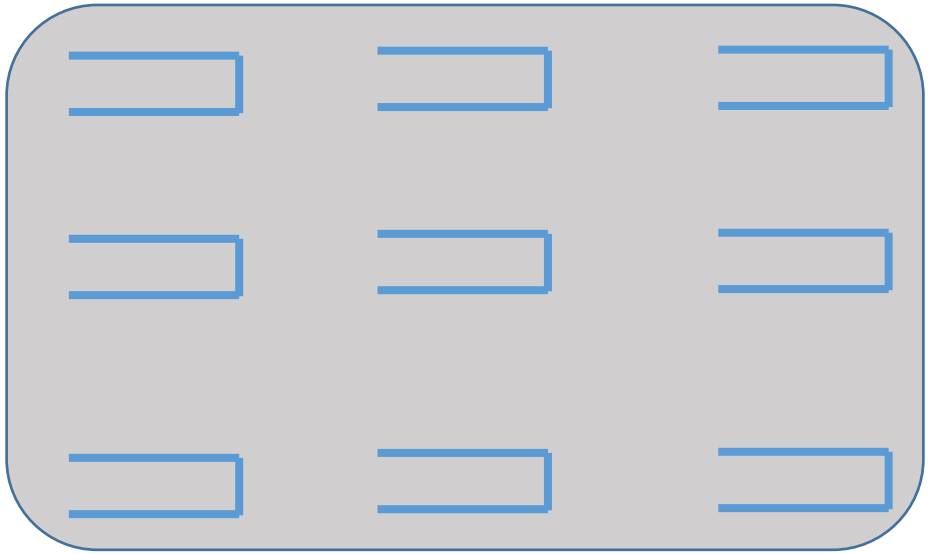


Data Centers



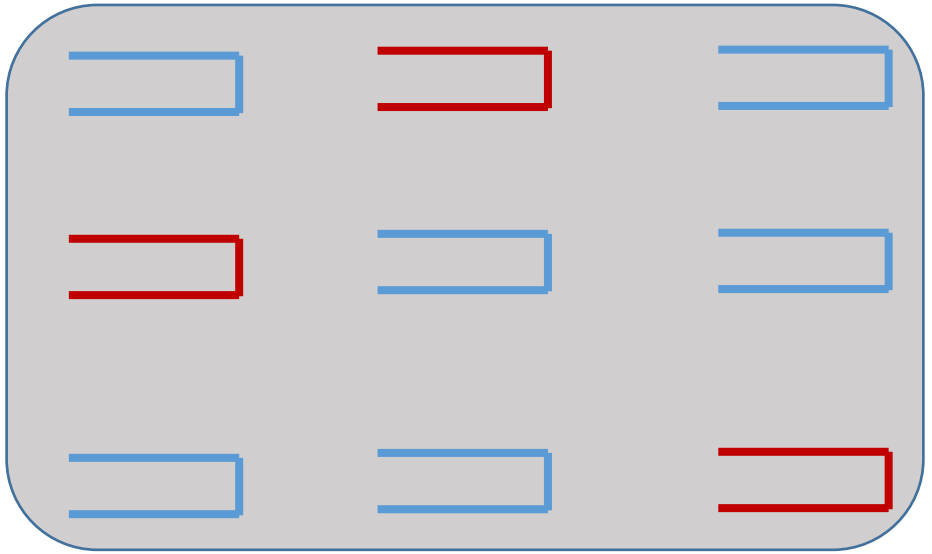
nxn Switch – another view

- A matrix of queues operating in discrete-time
- Key constraint: At most one queue from each row, and one from each column can be served in each time slot



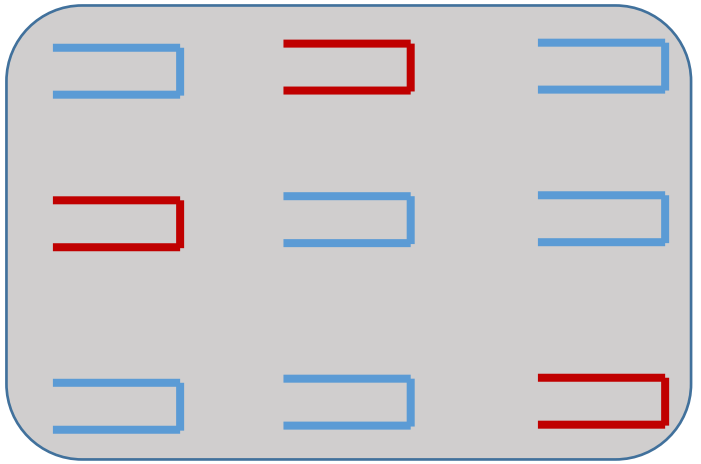
nxn Switch – Another View

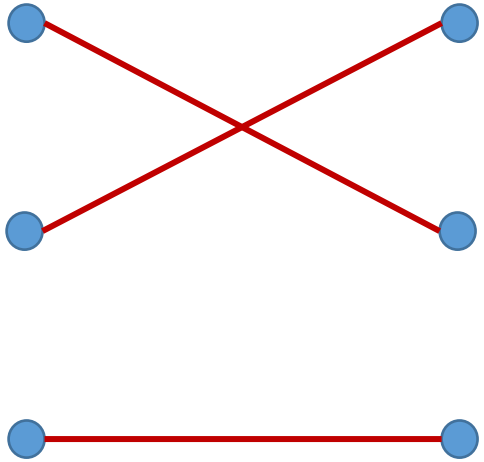
- A matrix of queues operating in discrete-time;
- Key constraint: At most one queue from each row, and one from each column can be served in each time slot
- Row corresponds to input port and column corresponds to output port



Bipartite Graph

- The row/column constraints can be captured by a bipartite graph
  - Each row is represented by a vertex on left
  - Each column by a vertex on right
- Each valid schedule is a matching
  - A collection of edges with no common node

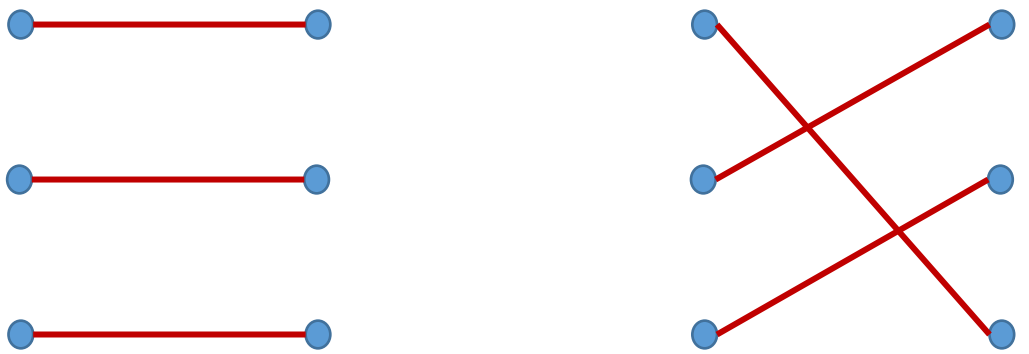




Schedules and Permutation Matrices

- Maximal Schedules are $n \times n$ permutation matrices
  - A matrix with exactly one 1 in each row and one 1 in each column, with all other entries equal to zero

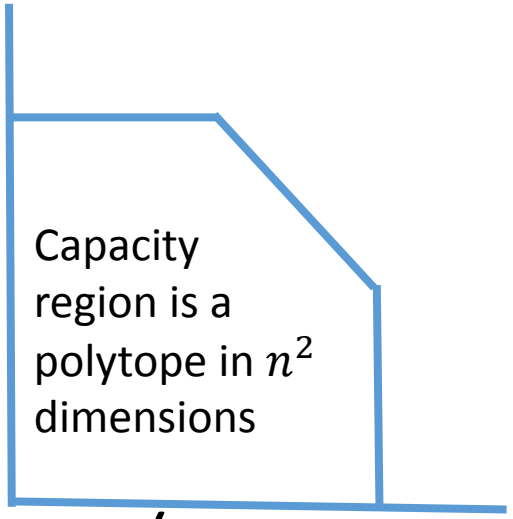
$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$



Throughput Maximization

- λ_{ij}: Arrival rate of packets to Queue (i,j).
- Maximum supportable set of arrival rates must satisfy:

$$\sum_j \lambda_{ij} \leq 1, \qquad \sum_i \lambda_{ij} \leq 1.$$



- Total number of arrivals in each time slot in each row (or column) must be less than 1.

Another view of the capacity

- Let α_P be the fraction of times that permutation matrix P is used
  - Recall $P_{ij} = 1$ if and only if Queue (i,j) is scheduled

- So the fraction of times, Queue (i,j) is scheduled is given by

$$\sum_{P:P_{ij}=1} \alpha_P$$

- Thus,

$$\lambda_{ij} \leq \sum_{P:P_{ij}=1} \alpha_P$$

Another View of the Capacity Region

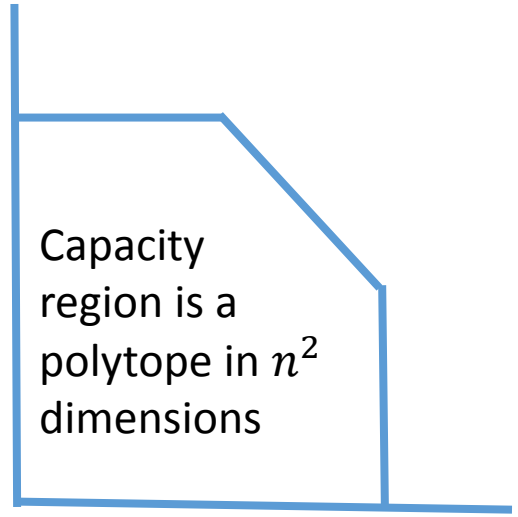
- Λ: Matrix of arrival rates; λ_{ij} is the arrival rate to Queue (i,j)

- Convex combination of permutation matrices

$$\Lambda \leq \sum_{P} \alpha_P P$$

with

$$0 \leq \alpha_P \leq 1, \qquad \sum_{P} \alpha_P = 1$$



$$\sum_{i} \lambda_{ij} \leq 1, \qquad \sum_{j} \lambda_{ij} \leq 1.$$

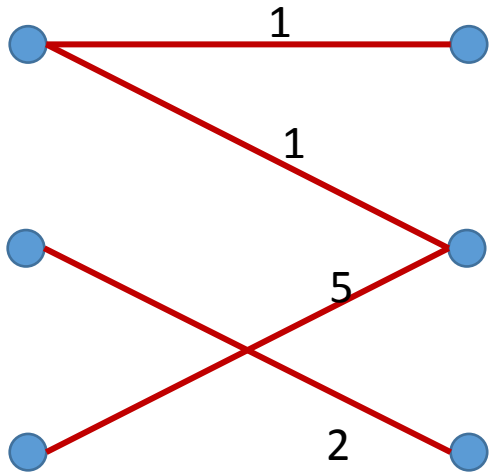
Goal: Throughput Maximization

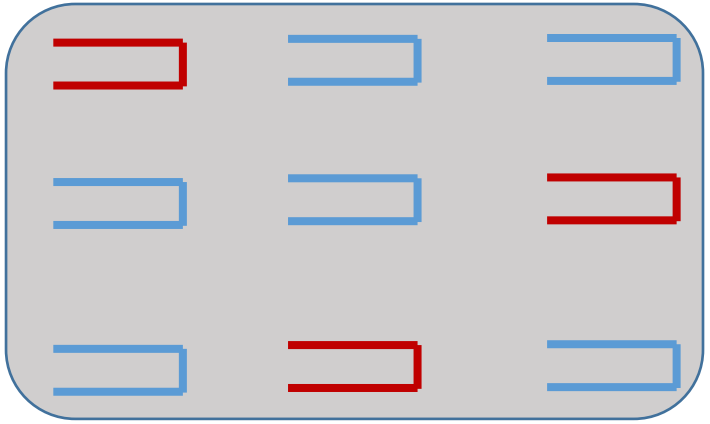
- Find a schedule (i.e., matchings or permutation matrices) in every time slot to stabilize the queues in the network, as long as the arrival rate lies within the capacity region
  - The long-term average rate at which packets are removed from each queue must be greater than or equal to the rate at which packets arrive at the queue

- Ideally, we would like to do this without knowledge of the arrival rates; we only know that they lie within the capacity region (Why?)

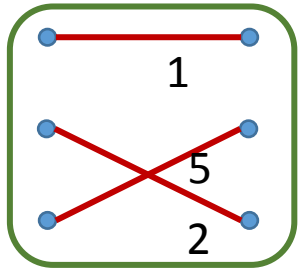
MaxWeight Scheduling Algorithm

- Weighted Bipartite graph
  - Weight of an edge from node i to node j on the right equals to q_{ij}
- Find a matching with the largest weight
- Schedule the corresponding queues

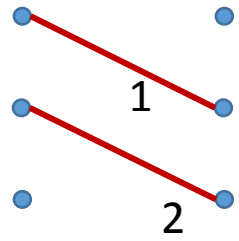




Weight: 8



Weight: 3



Linear Program

- The MaxWeight schedule is the solution to

$$\max_{P} < q, P >$$

where q is the matrix of queue lengths and P denotes a permutation matrix

- Equivalent LP:

$$\max_{P} < q, P > = \max_{\mu \in C} < q, \mu >$$

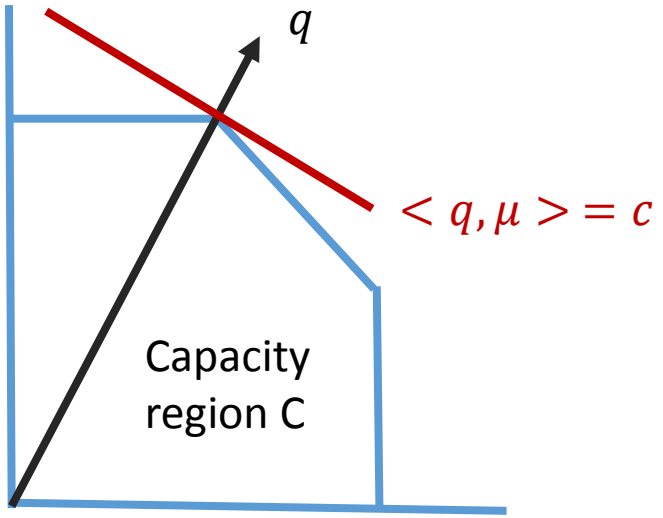
MaxWeight Algorithm

- The MaxWeight schedule is the solution to

$$\max_{\mu \in C} < q, \mu >,$$

where q is the matrix of queue lengths and μ is the average service

- MaxWeight algorithm is throughput-optimal (Tassiulas-Ephremides 1992, McKeown-Anantharam-Walrand 1996)



Optimization Interpretation

- Consider the following "optimization" problem (feasibility problem)

$$\max_{\mu \in C} \quad 0$$

s.t.

$$\lambda_{ij} \leq \mu_{ij} \quad \forall i, j$$

- Lagrangian:

Lagrange multipliers/ Dual variables

$$\max_{\mu \in C} \quad L(p, \mu) = -\sum_{ij} p_{ij}(\lambda_{ij} - \mu_{ij})$$

s.t.

$$p_{ij} \geq 0 \quad \forall i, j$$

MaxWeight as a Dual Algorithm

$$\max_{\mu \in C} \quad L(p,\mu) = -\sum_{ij} p_{ij}(\lambda_{ij} - \mu_{ij})$$

s.t.

$$p_{ij} \geq 0 \ \ \forall i,j$$

- Suppose the optimal dual variables are known, then the optimal solution is to pick service μ that solves $\max_{\mu \in C} < p, \mu >$

- In order to find the optimal dual variables μ, we consider a dual gradient algorithm

$$p_{ij}(k+1) = \left[p_{ij}(k) + \lambda_{ij} - \mu_{ij}\right]^{+}$$

- Queue lengths are like dual variables

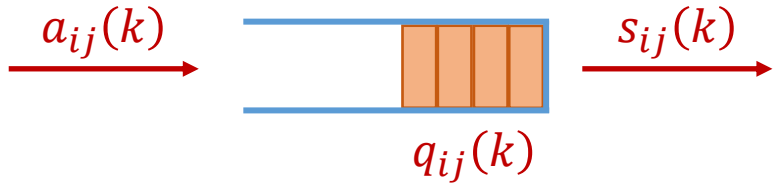
Queue Lengths and Stability

- Queue lengths evolves as:

$$q_{ij}(k+1) = \left[q_{ij}(k) + a_{ij}(k) - s_{ij}(k)\right]^{+}$$

$$q_{ij}(k+1) = \left[q_{ij}(k) + a_{ij}(k) - s_{ij}(k) + u_{ij}(k)\right]$$

Unused Service



- Stability roughly means that all the queue lengths are finite.

Stability Definition 1

- Definition (A Weak Version): The average expected queue length is bounded

$$\lim_{K\to\infty}\frac{1}{K}\sum_{k=1}^{K}E\left(|\boldsymbol{q}(k)|\right)\leq M<\infty$$

- Implication: Since $P(|\boldsymbol{q}(k)|\geq C)\leq\frac{E(|\boldsymbol{q}(k)|)}{C}$, we also have

$$\lim_{K\to\infty}\frac{1}{K}\sum_{k=1}^{K}P(|\boldsymbol{q}(k)|\geq C)\leq\frac{M}{C},\qquad\forall C>0$$

Stability Definition 2

- Let $\pi_i = \lim_{k\to\infty} P(|\boldsymbol{q}(k)| = i)$ be the steady-state probability that the total queue length is equal to i.

- Definition 2: Stable if

$$\sum_{i=0}^{\infty} \pi_i = 1$$

- Note: If $\sum_{i=0}^{\infty} \pi_i < 1$, then there is a non-zero probability of the queue being infinity. The above definition simply avoids it.

Stability Definition 3

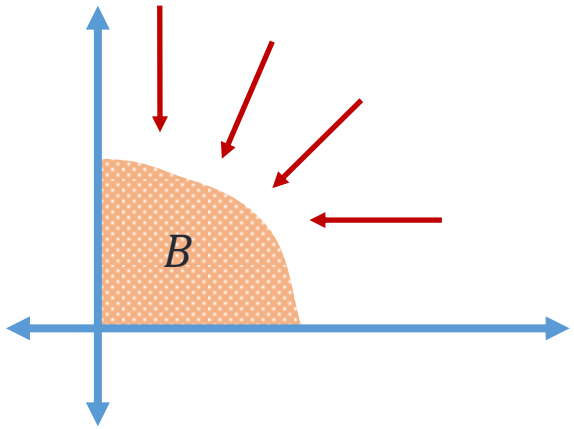
- Definition 3: The steady-state expected queue length is finite:

$$\lim_{k\to\infty} E(|\boldsymbol{q}(k)|) < \infty$$

- Often, this form of stability needed in practice. If the expectation is finite, then one can compare different stable scheduling algorithms in terms of their steady-state expected queue lengths

How to prove stability?

- Lyapunov function $V(\boldsymbol{q})$
  - Nonnegative function



- Negative drift outside a finite set B; Bounded drift otherwise

$$E\left[V\left(\boldsymbol{q}(k+1)\right) - V\left(\boldsymbol{q}(k)\right)\middle|\boldsymbol{q}(k)\right] \leq -\delta 1_{\{\boldsymbol{q}\in B^c\}} + M 1_{\{q\in B\}}$$

- Then, $\boldsymbol{q}$ is stable
  - Foster-Lyapunov Theorem

Stability in Switches

- Example: $\lambda_{ij} = \frac{1-\epsilon}{n} \quad \forall i, j$
- Using $V(\boldsymbol{q}) = \sum_{ij} q_{ij}^2$ as the Lyapunov function, can show

$$E\left[V\left(\boldsymbol{q}(k+1)\right) - V\left(\boldsymbol{q}(k)\right) \middle| \boldsymbol{q}(k)\right] \leq -\frac{\epsilon}{n}\sum_{ij} q_{ij} + n$$

- This shows the stability of the MaxWeight algorithm; true $\forall \lambda \in C$

Queue Length Upper Bound

- Left-hand size is zero in steady-state:

$$E\left[V\left(\boldsymbol{q}(k+1)\right) - V\left(\boldsymbol{q}(k)\right)\right] \leq -\frac{\epsilon}{n} E\left[\sum_{ij} q_{ij}\right] + n$$

- So, we get

$$E\left[\sum_{ij} q_{ij}\right] \leq \frac{n^2}{\epsilon}$$

Conjectured to be loose

Part II: Scaling Questions

- Let the total arrival rate to each row and each column be $1 - \epsilon$:

$$\sum_i \lambda_{ij} = 1 - \epsilon, \qquad \sum_j \lambda_{ij} = 1 - \epsilon$$

- Increase the size of the switch n and/or decrease ϵ, how does the expected steady-state queue length scale?

Conjecture 1 (Stolyar, 2005)

$$\frac{K_1 n}{\epsilon} \leq E\left(\sum_{ij} q_{ij}\right) \leq \frac{K_2 n}{\epsilon}$$

- For a fixed ϵ, the delay is $\Theta(1)$, independent of the size of the network

- Unsolved for the MaxWeight algorithm. Why does it matter?

Conjecture 2: Heavy Traffic (Shah, Tsitsiklis, Zhong 2012)

$$\lim_{\epsilon \to 0} \epsilon E\left(\sum_{ij} q_{ij}\right) = \Theta(n)$$

- In other words,

$$E\left(\sum_{ij} q_{ij}\right) = \frac{\Theta(n)}{\epsilon} + o\left(\frac{1}{\epsilon}\right)$$

- In this case, it is also useful to know the exact expression for the leading term, and if it is optimal.

Conjecture 3 (Shah, Tsitsiklis, Zhong, 2012)

- Suppose $\epsilon = \frac{1}{n^{\beta}}$
- Then,

$$E\left(\sum_{ij} q_{ij}\right) = \Theta\left(\frac{n}{\epsilon}\right) = \Theta(n^{\beta+1})$$

- We will address Conjectures 2 and 3.