Achieving 100% Throughput in an Input-Queued Switch

Nick McKeown
Department of Electrical Engineering
Stanford University, Stanford, CA 94305-4070
nickm@ee.stanford.edu

Venkat Anantharam   Jean Walrand
Department of Electrical Engineering and Computer Sciences
University of California at Berkeley, Berkeley, CA 94720
{ananth,wlr}@eecs.berkeley.edu

Abstract

It is well known that head-of-line (HOL) blocking limits the throughput of an input-queued switch with FIFO queues. Under certain conditions, the throughput can be shown to be limited to approximately 58%. It is also known that if non-FIFO queueing policies are used, the throughput can be increased. However, it has not been previously shown that if a suitable queueing policy and scheduling algorithm are used then it is possible to achieve 100% throughput for all independent arrival processes. In this paper we prove this to be the case using a simple linear programming argument and quadratic Lyapunov function. In particular, we assume that each input maintains a separate FIFO queue for each output and that the switch is scheduled using a maximum weight bipartite matching algorithm.

1 Introduction

Since Karol et al.'s paper was published in 1986, [11], it has become well known that an $N \times N$ port input-queued switch with FIFO queues can have a throughput limited to just $(2 - \sqrt{2}) \approx 58.6\%$. The conditions for this to be true are that:

1. Arrivals at each input are independent and identically distributed (i.i.d.).
2. Arrival processes at each input are independent of arrivals at other inputs.
3. All arrival processes have the same arrival rate and destinations are uniformly distributed over all outputs.
4. Arriving packets are of fixed and equal length, called cells.
5. N is large.

When conditions (1) and (2) are true we shall say that arrivals are *independent,* and when condition (3) is true we shall say that arrivals are *uniform.*

The throughput is limited because a cell can be held up by another cell ahead of it in line that is destined for a different output. This phenomenon is known as HOL blocking.

It is well documented that this result applies only to input-queued switches *with FIFO queues*. And so many techniques have been suggested for reducing HOL blocking using non-FIFO queues, for example by examining the first K cells in a FIFO queue, where K>1 [5][8][10]. In fact, HOL blocking can be eliminated entirely by using a simple buffering strategy at each input port. Rather than maintain a single FIFO queue for all cells, each input maintains a separate queue for each output [1][9][15][16][17][18], as shown in Figure 1. HOL blocking is eliminated because a cell cannot be held up by a cell queued ahead of it that is destined for a different output. This implementation is slightly more complex, requiring N FIFOs to be maintained by each input buffer. But no additional speedup is required: at most one cell can arrive and depart from each input in a cell time. During each slot time a scheduling algorithm decides the configuration of the switch by finding a matching on a bipartite graph (described below). A number of different techniques have been used for finding such a matching, for example using neural networks [2][4][20], or iterative algorithms [1][13][14]. These algorithms were designed to give high throughput while remaining simple to implement in hardware. When traffic is uniform, these algorithms perform well (>90% throughput). The *i*SLIP algorithm [13][14], for example, has been demonstrated using simulation to achieve 100% throughput when the traffic is independent and uniform. However, all of these algorithms perform less well and are unable to sustain a throughput of 100% when traffic is non-uniform.

It is worth asking the question:

What is the highest throughput that can be achieved by an input-queued switch which uses the queueing discipline shown in Figure 1?

In this paper we prove that for independent arrivals (uniform or non-uniform), a maximum throughput of 100% is achievable using a maximum weight matching algorithm.

In Section 2 we describe our model for an input-queued switch that uses the queueing discipline illustrated in Figure 1. We then consider two graph algorithms that can be used to schedule the transfer of cells through the switch. In Section 3 we describe the maximum size scheduling algorithm. Although this algorithm achieves 100% throughput for uniform traffic, we show that it can become unstable, even starve input queues, when arrivals are non-uniform. Section 4 describes the maximum weight scheduling algorithm. In conjunction with the appendix, we prove that the maximum weight scheduling algorithm is stable for all uniform and non-uniform independent arrival processes up to a maximum throughput of 100%. It is important to note that this is a theoretical result — the maximum weight matching algorithm that we propose is not readily implemented in hardware. However, our result indicates that a more practical technique that approximates this algorithm can be expected to perform well.

2 Our Model

Consider the "input-queued cell switch" in Figure 1 connecting M inputs to N outputs. The stationary and ergodic arrival process $A_i(n)$ at input i, $1 \le i \le M$, is a discrete-time process of fixed sized packets, or cells. At the beginning of each time slot, either zero or one cell arrive at each input. Each cell contains an identifier that indicates which output j, $1 \le j \le N$, it is destined for. When a cell destined for output j arrives at input i it is placed in the FIFO queue $Q(i,j)$ which has occupancy $L_{i,j}(n)$. Define the following vector which represents the occupancy of all queues at time n:

$$\underline{L}(n) \equiv (L_{1,1}(n), \ldots, L_{1,N}(n), \ldots, L_{M,N}(n))^T. \qquad (1)$$

We shall define the arrival process $A_{i,j}(n)$ to be the process of arrivals at input i for output j at rate $\lambda_{i,j}$, and the set of all arrival processes $A(n) = \{A_i(n); 1 \le i \le M\}$. $A(n)$ is considered *admissible* if no input or output is oversubscribed, i.e. $\sum_i \lambda_{ij} < 1$, $\sum_j \lambda_{ij} < 1$, otherwise it is *inadmissible*.

The FIFO queues are served as follows. A scheduling algorithm selects a *match*, or *matching*, M between the in-

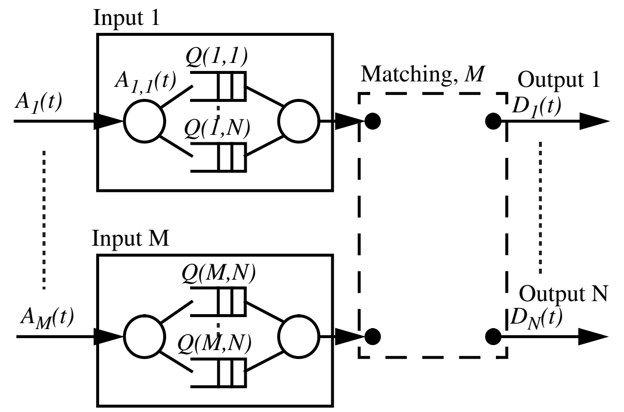


Figure 1: Components of an Input-Queued Cell-Switch.

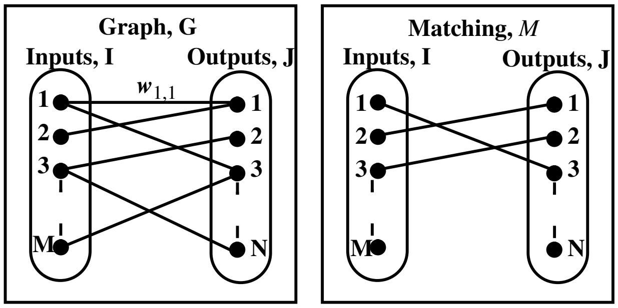
puts and outputs, defined as a collection of edges from the set of non-empty input queues to the set of outputs such that each non-empty input is connected to at most one output and each non-empty output is connected to at most one input. At the end of the time slot, if input i is connected to output j, one cell is removed from $Q(i,j)$ and sent to output j. Clearly, the departure process from output j, $D_j(t)$, rate μ_j is also a discrete-time process with either zero or one cell departing from each output at the end of each time slot. We shall define the departure process $D_{i,j}(t)$, rate $\mu_{i,j}$, as the process of departures from output j that were received from input i. Note that the departure rate may not be defined if the departure process is not stationary and ergodic.

To find a matching M, the scheduling algorithm solves a bipartite graph matching problem. An example of a bipartite graph is shown in Figure 2.

If the queue $Q(i,j)$ is non-empty, $L_{i,j}(n) > 0$ and there is an edge in the graph G between input i and output j. We associate a weight $w_{i,j}(n)$ to each such edge. The meaning of the weights depend on the algorithm, and we consider two algorithms here:

1. **Maximum Size Matching Algorithms:** Algorithms that find the match containing the maximum number of edges.
2. **Maximum Weight Matching Algorithms:** Algorithms that find the maximum weight matching where, in this paper, we only consider algorithms for which the weight $w_{i,j}(n)$ is integer-valued, equalling the occupancy $L_{i,j}(n)$ of $Q(i,j)$.

Clearly, a maximum size match is a special case of the maximum weight matching with weight $w_{i,j}(n) = 1$.



a) Example of G for $|\mathbf{I}|$= M and $|\mathbf{J}|$= N.

b) Example of matching *M* on G.

Figure 2: Define G = [V,E] as an undirected graph connecting the set of vertices V with the set of edges E. The edge connecting vertices *i*, 1≤*i*≤*M* and *j*, 1≤*j*≤*N* has an associated weight denoted $w_{i,j}$. Graph G is bipartite if the set of inputs I = {i: 1≤*i*≤*M*} and outputs J = {*i:* 1≤*j*≤*N*} partition V such that every edge has one end in I and one end in J. Matching M on G is any subset of E such that no two edges in *M* have a common vertex. A *maximum matching algorithm* is one that finds the matching M_{max} with the maximum total size or total weight.

3 Maximum Size Matchings

The maximum size matching for a bipartite graph can be found by solving an equivalent network flow problem [19]. There exist many algorithms for solving these problems, the most efficient algorithm currently known converges in $O(N^{5/2})$ time and is described in [7].[1]

It can be demonstrated using simulation that the maximum size matching algorithm is stable for i.i.d. arrivals up to an offered load of 100% *when the traffic is uniform* [14]. It is important to note that a maximum size matching is not necessarily desirable. First, under *admissible* traffic it can lead to instability and unfairness, particularly for non-uniform traffic patterns. To demonstrate this behavior, Figure 3 shows an example of a potentially unstable 3x3 switch with just four active flows,[2] and scheduled using the maximum size matching algorithm. It is assumed that ties are broken by selecting among alternatives with equal probability. Arrivals to the switch are i.i.d. Bernoulli arrivals and each flow has arrivals at rate $(1/2) - \delta$, where $\delta > 0$. Even though the traffic is admissible, it is straightforward to show that the switch can be unstable for sufficiently small δ. Consider the event that at time *n*, both $A_{2,1}(n)$ and $A_{3,2}(n)$ have arrivals (with probability $((1/2) - \delta)^2$) and $L_{1,1}(n) > 0$, $L_{1,2}(n) > 0$, in which case input 1 receives service with probability 2/3. Therefore the total rate at which input 1 receives service is at most:

$$\frac{2}{3}\left(\frac{1}{2} - \delta\right)^2 + \left(1 - \left(\frac{1}{2} - \delta\right)^2\right)$$
$$= 1 - \frac{1}{3}\left(\frac{1}{2} - \delta\right)^2$$

But the arrival rate to input 1 is $1 - 2\delta$, so if

$$2\delta < \frac{1}{3}\left(\frac{1}{2} - \delta\right)^2,$$

(which holds for $\delta < 0.0358$), then the switch is unstable and the traffic cannot be sustained by the maximum size matching algorithm.

Second, under *inadmissible* traffic, the maximum size matching algorithm can lead to *starvation*. An example of this behavior is shown in Figure 4 for a 2x2 switch. It is clear that because all three queues are permanently occupied, the algorithm will always select the "cross" traffic: input 1 to output 2 and input 2 to output 1.It is worth noting that the most practical among the scheduling algorithms described earlier attempt to approximate a maximum size matching [1][2][4][13][20]. It is therefore not surprising that these algorithms perform well when the traffic is uniform, but perform less well when the traffic is non-uniform.

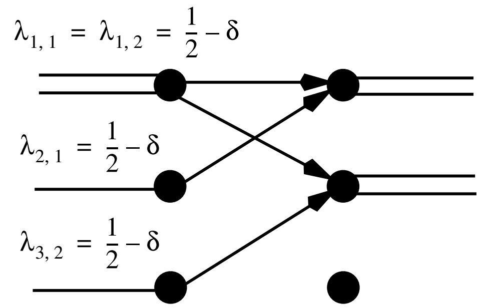


Figure 3: Example of *instability* under admissible traffic using a maximum size matching algorithm for a 3x3 switch with non-uniform traffic.

1.This algorithm is equivalent to Dinic's algorithm [6].

2.It can also be shown that a 2x2 switch with non-uniform traffic can be unstable when scheduled using a maximum size matching algorithm. However, our proof is more complex and is omitted here.

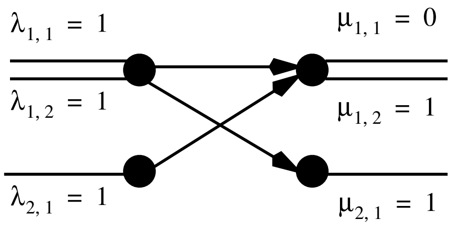


Figure 4: Under an *inadmissible* workload, the maximum size match will always serve just two queues, *starving* the flow from input 1 to output 1.

4 Maximum Weight Matchings

The maximum weight matching M for a bipartite graph is one that maximizes $\sum_{(i,j) \in M} w_{i,j}$ and can be found by solving an equivalent network flow problem. The most efficient known algorithm for solving this problem converges in $O(N^3 \log N)$ running time [19].

The maximum *size* matching algorithm described above knows only whether an input queue $Q(i, j)$ is empty or non-empty. Therefore, if the traffic is non-uniform and the occupancy of some queues begins to increase, this algorithm does not know to favor those queues and reduce their backlog.

On the other hand, the maximum *weight* matching algorithm knows the occupancy of each queue, $L_{i,j}(n)$, and can thus give preference to queues with greater occupancy. In fact, as the following theorem shows, a maximum throughput 100% is possible for independent and either uniform or non-uniform arrivals.

4.1 Main Result

Theorem: *The maximum weight matching algorithm is stable for all admissible i.i.d. arrival processes.*

Proof: The proof is given in the Appendix. In summary, we show that for an $M \times N$ switch scheduled using the *maximum weight matching* algorithm, there is a negative expected single-step drift in the sum of the squares of the occupancy. In other words,

$$E\,[\underline{L}^T(n+1)\underline{L}(n+1) - \underline{L}^T(n)\underline{L}(n) \mid \underline{L}(n)] \\ \leq -\varepsilon \|\underline{L}(n)\| + k$$

where, $k > 0, \varepsilon > 0$.

$V(n) = \underline{L}^T(n)\underline{L}(n)$ is a 2nd order Lyapunov function and, using the result of Kumar and Meyn [12] we show that the system is stable. The term $-\varepsilon\|\underline{L}(n)\|$ indicates that whenever the occupancy of the input queues is large enough, the expected drift is negative; should $\|\underline{L}(n)\|$ become very large, the downward drift also becomes large.

5 Conclusion

We have shown that if an input-queued switch maintains a separate FIFO queue for each output at each input, then the maximum throughput is 100% for independent arrivals. If a maximum sized matching algorithm is used to schedule cells, then we demonstrate that a throughput of 100% is possible only when arrivals are uniform. However, if a maximum weight matching algorithm is used, we have proved that a throughput of 100% is achievable for both uniform and non-uniform arrivals.

6 References

[1] Anderson, T.; Owicki, S.; Saxe, J.; and Thacker, C. “High speed switch scheduling for local area networks,” *ACM Trans. on Computer Systems.* Nov 1993 pp. 319-352.

[2] Ali, M.; Nguyen, H. “A neural network implementation of an input access scheme in a high-speed packet switch,” *Proc. of GLOBECOM 1989*, pp.1192-1196.

[3] Birkhoff, G.; “Tres observaciones sobre el algebra lineal,” *Univ. Nac. Tucumán Rev. Ser. A5 (1946),* pp. 147-150.

[4] Brown, T.X; Liu, K.H. “Neural network design of a Banyan network controller,” *IEEE J. Selected Areas Communications*, Vol.8, pp.1289-1298, Oct. 1990.

[5] Chen, M.; Georganas, N.D., “A fast algorithm for multi-channel/port traffic scheduling” *Proc. IEEE Supercom/ICC ‘94*, pp.96-100.

[6] Dinic, E.A. “Algorithm for solution of a problem of maximum flow in a network with power estimation,” *Soviet Math. Dokl.* Vol.11, pp. 1277-1280, 1970.

[7] Hopcroft, J.E.; Karp, R.M. “An $n^{5/2}$ algorithm for maximum matching in bipartite graphs,” *Society for Industrial and Applied Mathematics J. Comput.,* 2 (1973), pp.225-231.

[8] Huang, A.; Knauer, S. “Starlite: A wideband digital switch,” *Proc. GLOBECOM ‘84* (1984), pp.121-125.

[9] Karol, M.; Eng, K.; Obara, H. “Improving the performance of input-queued ATM packet switches,” *INFOCOM ‘92*, pp.110-115.

[10] Karol, M.; Hluchyj, M. “Queueing in high-performance packet-switching,” *IEEE J. Selected Area Communications*, Vol.6, pp.1587-1597, Dec. 1988.

[11] Karol, M.; Hluchyj, M.; and Morgan, S. “Input versus output queueing on a space division switch,” *IEEE Trans. Communications*, 35(12) (1987) pp.1347-1356.

[12] Kumar, P.R.; Meyn, S.P.; "Stability of Queueing Networks and Scheduling Policies", *IEEE Transactions on Automatic Control*, Vol.40, No.2, Feb. 1995.

[13] McKeown, N.; Walrand, J.; and Varaiya, P.; "Scheduling Cells in an Input-Queued Switch." *IEE Electronics Letters,* Dec 9th 1993, pp.2174-5.

[14] McKeown, N.; "Scheduling Algorithms for Input-Queued Cell Switches," PhD Thesis. University of California at Berkeley, 1995.

[15] Obara, H. "Optimum architecture for input queueing ATM switches," *IEE Electronics Letters*, pp.555-557, 28th March 1991.

[16] Obara, H.; Hamazumi, Y. "Parallel contention resolution control for input queueing ATM switches," *IEE Electronics Letters*, Vol.28, No.9, pp.838-839, 23rd April 1992.

[17] Obara, H.; Okamoto, S.; and Hamazumi, Y. "Input and output queueing ATM switch architecture with spatial and temporal slot reservation control" *IEE Electronics Letters*, pp.22-24, 2nd Jan 1992.

[18] Tamir, Y.; Frazier, G. "High performance multi-queue buffers for VLSI communication switches," *Proc. of 15th Ann. Symp. on Comp. Arch.*, June 1988, pp.343-354.

[19] Tarjan, R.E. "Data structures and network algorithms," *Society for Industrial and Applied Mathematics,* Pennsylvania, Nov 1983.

[20] Troudet, T.P.; Walters, S.M. "Hopfield neural network architecture for crossbar switch control," *IEEE Trans. Circuits and Systems,* Vol.CAS-38, pp.42-57, Jan.1991.

Appendix A.1: Proof of Theorem

A.1.1 Definitions

In this appendix we use the following definitions for an $M \times N$ switch:

1. The rate matrix of the stationary arrival processes:

   $\Lambda \equiv [\lambda_{i,j}]$, where: $\sum_{i=1}^{M} \lambda_{i,j} \le 1, \sum_{j=1}^{N} \lambda_{i,j} \le 1, \lambda_{i,j} \ge 0$

   and associated rate vector:

   $$\underline{\lambda} \equiv (\lambda_{1,1}, \ldots, \lambda_{1,N}, \ldots, \lambda_{M,1}, \ldots, \lambda_{M,N})^T. \qquad (2)$$

2. The arrival matrix, representing the sequence of arrivals into each queue:

   $\mathbf{A}(n) \equiv [A_{i,j}(n)]$

   where:

   $$A_{i,j}(n) \equiv \begin{cases} 1 & \text{if arrival occurs at } Q(i,j) \text{ at time } n \\ 0 & \text{else} \end{cases},$$

   and associated arrival vector:

   $\underline{A}(n) \equiv (A_{1,1}(n), \ldots, A_{1,N}(n), \ldots, A_{M,N}(n))^T$.

3. The service matrix, indicating which queues are served at time n:

   $\mathbf{S}(n) \equiv [S_{i,j}(n)]$, where:

   $$S_{i,j}(n) = \begin{cases} 1 & \text{if } Q(i,j) \text{ is served at time } n \\ 0 & \text{else} \end{cases},$$

   and $\mathbf{S}(n) \in \mathbf{S}$, the set of service matrices.

   Note that: $\sum_{i=1}^{M} S_{i,j}(n) = \sum_{j=1}^{N} S_{i,j}(n) = 1$, and hence if $M = N$, $\mathbf{S}(n) \in \mathbf{S}$ is a *permutation matrix*. If $M \ne N$, we say that $\mathbf{S}(n) \in \mathbf{S}$ is a *quasi-permutation matrix*. We define the associated service vector:

   $\underline{S}(n) \equiv (S_{1,1}(n), \ldots, S_{1,N}(n), \ldots, S_{M,N}(n))^T$,

   hence $\|\underline{S}(n)\|^2 = \sqrt{NM}$.

4. The *approximate* next-state vector:

   $\tilde{\underline{L}}(n+1) \equiv \underline{L}(n) - \underline{S}(n) + \underline{A}(n)$, which approximates the exact next-state of each queue

   $$L_{i,j}(n+1) = [L_{i,j}(n) - S_{i,j}(n)]^+ + A_{i,j}(n). \qquad (3)$$

A.1.2 Proof of Theorem.

Before proving the theorem, we first state the following fact and prove the subsequent lemmas.

Fact 1: *(Birkhoff's Theorem) The doubly sub-stochastic $N \times N$ square matrices form a convex set, C, with the set of extreme points equal to permutation matrices,* $\mathbf{S}$.

This is proved in [3].

Lemma 1: *The doubly sub-stochastic $M \times N$ non-square matrices form a convex set, C, with the set of extreme points equal to quasi-permutation matrices,* $\mathbf{S}$

Proof: Observe that we can add *N* - *M* rows to any non-square sub-stochastic matrix and introduce new entries so that the row sums of the new rows equal one and further that the column sums are also each 1. We can use Birkhoff's Theorem to write the augmented matrix as a convex combination of $N \times N$ permutation matrices. The first M rows of the permutation matrix is an $M \times N$ matrix which forms a permutation matrix with some M of the N columns.■

Lemma 2: $\underline{L}^T(n)(\underline{\lambda} - \underline{S}^*(n)) \le 0, \quad \forall (\underline{L}(n), \underline{\lambda})$ *, where*

$\underline{S}^*(n) = \arg\max_{\underline{S}(n)} (\underline{L}^T(n)\underline{S}(n))$ *, the service matrix selected* *by the maximum weight matching algorithm to maximize* $\underline{L}^T(n)\underline{S}(n)$.

Proof: Consider the linear programming problem:

$$\max(\underline{L}^T(n)\underline{\lambda})$$
$$\text{s.t.}\quad \sum_{i=1}^{M}\lambda_{i,j} \le 1,\ \sum_{j=1}^{N}\lambda_{i,j} \le 1,\ \lambda_{i,j} \ge 0$$

which has a solution equal to an extreme point of the convex set, C. Hence,

$$\max(\underline{L}^T(n)\underline{\lambda}) \le \max(\underline{L}^T(n)\underline{S}(n)),$$

and so $\underline{L}^T(n)\underline{\lambda} - \max(\underline{L}^T(n)\underline{S}(n)) \le 0$. ■

Lemma 3:

$$\mathrm{E}\,[\tilde{\underline{L}}^T(n+1)\tilde{\underline{L}}(n+1) - \underline{L}^T(n)\underline{L}(n) |\ \underline{L}(n)] \le 2\sqrt{NM},\ \ \forall\underline{\lambda}$$

Proof:

$$\begin{aligned}
&\tilde{\underline{L}}^T(n+1)\tilde{\underline{L}}(n+1) - \underline{L}^T(n)\underline{L}(n)\\
&= (\underline{L}(n) - \underline{S}(n) + \underline{A}(n))^T(\underline{L}(n) - \underline{S}(n) + \underline{A}(n)) - \underline{L}^T(n)\underline{L}(n)\\
&= 2\underline{L}^T(n)(\underline{A}(n) - \underline{S}(n)) + (\underline{S}(n) - \underline{A}(n))^T(\underline{S}(n) - \underline{A}(n))\\
&= 2\underline{L}^T(n)(\underline{A}(n) - \underline{S}(n)) + k,
\end{aligned}$$

where $0 \le k \le 2N$. $k \ge 0$ because $\underline{S}(n) - \underline{A}(n)$ is a real vector, and $k \le 2\sqrt{NM}$ because $\|\underline{S}(n) - \underline{A}(n)\|^2 \le 2\sqrt{NM}$.

Taking the expected value:

$$\begin{aligned}
&E\,[\tilde{\underline{L}}^T(n+1)\tilde{\underline{L}}(n+1) - \underline{L}^T(n)\underline{L}(n) |\ \underline{L}(n)]\\
&\le E\,[2\underline{L}^T(n)(\underline{A}(n) - \underline{S}(n))]\\
&= 2\underline{L}^T(n)(\underline{\lambda} - \underline{S}^*(n)) + 2\sqrt{NM}.
\end{aligned}$$

From Lemma 2 we know that $2\underline{L}^T(n)(\underline{\lambda} - \underline{S}^*(n)) \le 0$, proving the lemma. ■

Lemma 4: *For any* $\varepsilon > 0$ *:*

$$\begin{aligned}
&\mathrm{E}\,[\tilde{\underline{L}}^T(n+1)\tilde{\underline{L}}(n+1) - \underline{L}^T(n)\underline{L}(n) |\ \underline{L}(n)]\\
&\qquad\le -\varepsilon\|\underline{L}(n)\| + 2\sqrt{NM}.
\end{aligned}$$

$\forall\underline{\lambda} \le (1-\beta)\underline{\lambda}_m, 0 < \beta < 1$, where $\underline{\lambda}_m$ is any rate vector such that $\|\underline{\lambda}_m\|^2 = \sqrt{NM}$.

Proof:

$$\begin{aligned}
&\underline{L}^T(n)(\underline{\lambda} - \underline{S}^*(n))\\
&\le \underline{L}^T(n)\{\underline{\lambda}_m - \underline{S}^*(n)\} - \underline{L}^T(n)(\beta\underline{\lambda}_m)\\
&\le 0 - \beta\|\underline{L}(n)\| \cdot \|\underline{\lambda}_m\|\cos\theta
\end{aligned}$$

where θ is the angle between $\underline{L}(n)$ and $\underline{\lambda}_m$.

We now show that $\cos\theta > \delta$ for some $\delta > 0$ whenever $\underline{L}(n) \ne \underline{0}$. First, we show that $\cos\theta > 0$. We do this by contradiction: suppose that $\cos\theta = 0$, i.e. $\underline{L}(n)$ and $\underline{\lambda}_m$ are orthogonal. This can only occur if $\underline{L}(n) = \underline{0}$, or if for some i, j, both $\lambda_{i,j} = 0$ and $L_{i,j}(n) > 0$, which is not possible: for arrivals to have occurred at queue $Q(i, j)$, $\lambda_{i,j}$ must be greater than zero. Therefore, $\cos\theta > 0$ unless $\underline{L}(n) = \underline{0}$. Now we show that $\cos\theta$ is bounded away from zero, i.e. that $\cos\theta > \delta$ for some $\delta > 0$. Because $\lambda_{i,j} > 0$ wherever $L_{i,j}(n) > 0$, and because $\|\underline{\lambda}\|^2 \le \sqrt{NM}$,

$$\cos\theta = \frac{\underline{L}^T(n)\underline{\lambda}}{\|\underline{L}(n)\|\|\underline{\lambda}\|} \ge \frac{L_{max}(n)\lambda_{min}}{\|\underline{L}(n)\|\,(NM)^{1/4}}, \qquad (4)$$

where $\lambda_{min} = \min(\lambda_{i,j}, 1 \le i \le M, 1 \le j \le N)$, and $L_{max}(n) = \max(L_{i,j}(n), 1 \le i \le M, 1 \le j \le N)$.

Also, $\|\underline{L}(n)\| \le [NML^2_{max}(n)]^{1/2} = \sqrt{NM}L_{max}(n)$, and so $\cos\theta$ is bounded by

$$\cos\theta \ge \frac{\lambda_{min}}{(NM)^{3/4}} \qquad (5)$$

Therefore

$$\begin{aligned}
&\mathrm{E}\,[\tilde{\underline{L}}^T(n+1)\tilde{\underline{L}}(n+1) - \underline{L}^T(n)\underline{L}(n) |\ \underline{L}(n)]\\
&\le -\frac{\beta\lambda_{min}}{\sqrt{N}}\|\underline{L}(n)\| + 2\sqrt{NM}
\end{aligned}$$

■

Lemma 5: *For all* $\varepsilon > 0$ *,*

$$\begin{aligned}
&\mathrm{E}\,[\underline{L}^T(n+1)\underline{L}(n+1) - \underline{L}^T(n)\underline{L}(n) |\ \underline{L}(n)]\\
&\qquad\le -\varepsilon\|\underline{L}(n)\| + NM + 2\sqrt{NM}
\end{aligned}$$

$\forall\ \underline{\lambda} \le (1-\beta)\underline{\lambda}_m(n),\quad 0 < \beta < 1$.

$$L_{i,j}(n+1) = \tilde{L}_{i,j}(n+1) + \begin{cases} 1 & \text{if } L_{i,j}(n) = 0, S_{i,j}(n) = 1 \\ 0 & \text{else} \end{cases},$$

therefore

$$\underline{L}^T(n+1)\underline{L}(n+1) - \tilde{\underline{L}}^T(n+1)\tilde{\underline{L}}(n+1) \le NM, \qquad (6)$$

and so

$$\begin{aligned} &\mathrm{E}\,[\underline{L}^T(n+1)\underline{L}(n+1) - \underline{L}^T(n)\underline{L}(n) \,|\, \underline{L}(n)] \\ &\le \mathrm{E}\,[\tilde{\underline{L}}^T(n+1)\tilde{\underline{L}}(n+1) - \underline{L}^T(n)\underline{L}(n) \,|\, \underline{L}(n)] + NM \end{aligned}$$

Using Lemma this concludes the proof.■

Lemma 6: *There exists a* $V(\underline{L}(n))$ *s.t.* $\mathrm{E}\,[V(\underline{L}(n+1)) - V(\underline{L}(n)) \,|\, \underline{L}(n)] \le -\varepsilon \|\underline{L}(n)\| + k$, *where* $k, \varepsilon > 0$.

Proof: $V(\underline{L}(n)) = \underline{L}^T(n)\underline{L}(n)$ and $k = NM + 2\sqrt{NM}$ in Lemma 5.■

We are now ready to prove the main theorem. $V(\underline{L}(n))$ in the main Theorem is a quadratic Lyapunov function and, according to the argument of Kumar and Meyn [12], it follows that the switch is stable.