A Starvation-free Algorithm For Achieving 100% Throughput in an Input-Queued Switch

Adisak Mekkittikul Nick McKeown
Department of Electrical Engineering
Stanford University, Stanford, CA 94305-4030
Tel. (415) 723-1414, Fax. (415) 725-6949
e-mail: adisak@dara.stanford.edu

Abstract

It has been recently shown that an input-queued switch with an appropriate buffering policy and scheduling algorithm can achieve 100% throughput for independent arrival processes. This result was shown to be true for the longest queue first (LQF) algorithm — a scheduling policy that finds the maximum weight matching on a bipartite graph, and gives preference to more backlogged input queues. Despite its high performance, this algorithm can lead to the starvation of one or more input queues. In this paper we introduce an alternative algorithm: the oldest cell first (OCF), which gives preference to cells that have been waiting for the longest time. We show that like the LQF algorithm, the OCF algorithm can lead to 100% throughput for independent arrivals, and that no queue can be starved.

1: Introduction

There is a growing interest in input-queueing for high bandwidth switches. This is because input-queued switches do not require an internal "speedup". In an output-queued switch, the switching fabric and the output buffers must have a bandwidth equal to the line rate multiplied by the number of inputs, i.e., the buffers must have a speedup equal to the number of inputs. This speedup is necessary because, in the worst case, every input may have an arrival in the same cell time destined for the same output. As a result memory bandwidth limits the size and line rate of the switch. This is not a problem for today's commercial ATM switches; they are able to use output-queueing because line-rates are low and the number of inputs is small. However, the demand for bandwidth is growing rapidly and before long memory bandwidth will be insufficient for output-queueing to be practicable.

For switches with a high aggregate bandwidth, input-queueing is attractive — each input can accept at most one cell in a cell time, and can deliver at most one cell to the switch in a cell time. Despite this advantage, the long-standing view has been that input-queued switches are unsuitable because they can suffer a throughput limitation caused by head-of-line (HOL) blocking. The HOL blocking phenomenon occurs when an HOL cell blocks cells behind it in line that are destined to other outputs. For an input-queued switch with a single FIFO at each input port. Karol and et al. [3] have shown that under even benign traffic assumptions, HOL blocking results in a maximum throughput of just $(2-\sqrt{2}) \approx 58.6\,\%$.

However, Karol's result applies only to an input-queued switch *with a single FIFO at each input port*. Using a simple buffering strategy we can entirely eliminate HOL blocking. Specifically, if each input maintains a separate queue for each output (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. We call this scheme *Virtual Output Queueing* (VOQ).

When VOQ is used, a switch requires a scheduling algorithm to decide which one of the queues at each input can forward its HOL cell to the destination output. Finding a scheduling algorithm that is simple, fair, and efficient is critical in designing a high-speed input-queued switch. Through simulation results, a number of scheduling algorithms [1][2] have been shown to provide a high throughput.

Recent studies [5][6][7] have shown that a suitable scheduling algorithm can increase the throughput of an input-queued switch to 100% when arrivals are independent. In [6], an iterative algorithm called *i*SLIP is described. It can achieve 100% throughput for uniform traffic, is fair and can be easily implemented in hardware. However, the *i*SLIP algorithm cannot achieve 100% throughput when the traffic is non-uniform, or when arrivals are correlated. In [7], it was proved that an algorithm that finds a maximum weight matching on a bipartite

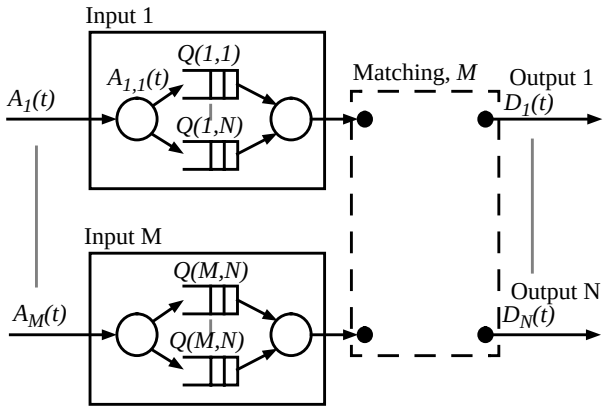


Figure 1: Components of an Input-Queued Cell-Switch.

graph can achieve 100% throughput for non-uniform traffic by giving preference to queues with a larger occupancy (the LQF algorithm). This result is a special case of Tassiulas' analysis in [8]. Unfortunately, the LQF algorithm may starve a low rate queue under a very heavy load. In this paper, we consider a different maximum weight matching algorithm: the oldest cell first (OCF). The OCF algorithm never starves any queue, and as we will show, is stable for all independent arrival processes.

We organize the paper as follows. Section 2 describes our input-queued switch model. Section 3 explains the two scheduling algorithms: LQF and OCF. Section 4 outlines the proofs for the stabilities of the LQF and OCF algorithms. Section 5 concludes our proof and summarizes our finding. Appendix A contains the detailed proofs of our theorems.

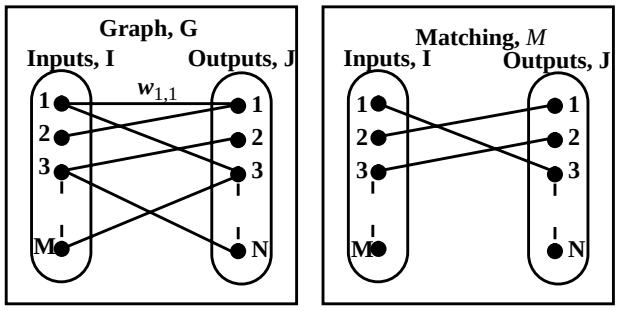
2: Input-queued switch model

The input-queued switch model shown in Figure 1 consists of M inputs, N outputs, a nonblocking switch fabric, and a scheduler. Each input maintains N FIFO queues to buffer arrivals. $Q_{i,j}$ denotes the queue at input i for cells destined to output j. Arrivals are sorted and placed in the corresponding queues, awaiting the scheduler's decision to send them to their destination.

Since arrivals are fixed size cells, time is slotted into cell times. During any given slot, there is at most one arrival to and at most one departure from each input and output. $A_{i,j}(n)$ is the arrival process of cells to input i destined to output j at rate $\lambda_{i,j}$. Consequently, $A_i(n)$ is the aggregate process of all arrivals to input i at rate $\lambda_i = \sum_{j=1}^{N} \lambda_{i,j}$. An arrival process is said to be *admissible* when no input or output is oversubscribed, i.e., when $\sum_{i=1}^{M} \lambda_{i,j} \le 1, \sum_{j=1}^{N} \lambda_{i,j} \le 1, \lambda_{i,j} \ge 0$.

In this paper, we consider only independent arrival processes; i.e., during any cell time, an arrival occurs with fixed probability p, independently of whether an arrival occurred in the previous cell time.

To get permission to send, all non-empty queues make requests to the scheduler which makes grants based on its scheduling algorithm. Granting can be viewed as solving a bipartite graph matching problem, an example of which is shown in Figure 2. Each edge of the bipartite graph G represents a request from an input-queue to an output. Each edge in the matching M corresponds to a granted request; each node in M can have at most one edge incident on it. After scheduling is complete, the queues with granted requests forward their HOL cells to the destination outputs.



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, $1 \le i \le M$ and j, $1 \le j \le N$ has an associated weight denoted $w_{i,j}$. Matching M on G is any subset of E such that no two edges in M have a common vertex. A *maximum matching algorithm* finds the matching M_{max} with the maximum total size or total weight.

3: Scheduling algorithms

The task of the scheduler is to find a conflict-free matching based on input requests. Our study considers two types of maximum weight matching algorithms: the previously described longest queue first (LQF) and the novel oldest cell first (OCF).

3.1: LQF algorithm

In the LQF algorithm, each weight $w_{i,j}(n)$ is equal to its corresponding queue occupancy, $L_{i,j}(n)$. Let $S_{i,j}(n)$ be a service indicator; a value of one indicates that the

request is granted and zero indicates otherwise. The algorithm picks a match such that the sum of serviced queues' occupancies is maximized, i.e., $\arg\max_{S}\left(\sum_{i,j} S_{i,j}(n)L_{i,j}(n)\right)$, such that $\sum_{i=1}^{M} S_{i,j}(n) \le 1$ and $\sum_{j=1}^{N} S_{i,j}(n) \le 1$. The LQF algorithm is known to be stable for all admissible arrival processes [7].

Unfortunately, although the LQF algorithm performs well, it can lead to starvation. A simple example is illustrated in Figure 3 for a 2×2 switch. Assume that both

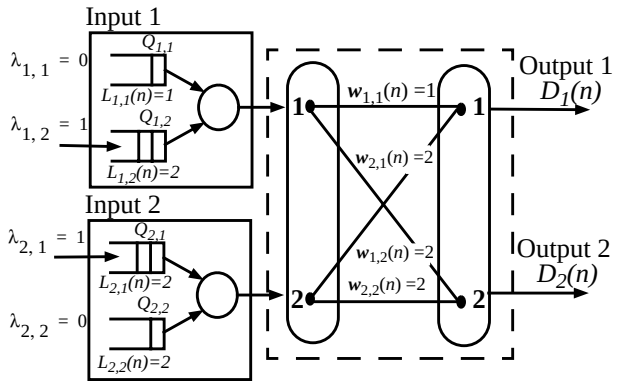


Figure 3: Example of 2x2 switch for which, using the LQF algorithm, an input queue may be starved.

$Q_{1,2}$ and $Q_{2,1}$ have exactly two cells waiting and that $Q_{1,1}$ and $Q_{2,2}$ have only one cell waiting. If there is an arrival at each input during every slot, but neither is for $Q_{1,1}$ or $Q_{2,2}$, these two queues are never served.

3.2: OCF algorithm

The OCF algorithm uses the waiting times of HOL cells as requesting weights. Let $W_{i,j}(n)$ denote the waiting time of the cell at HOL of $Q_{i,j}$. The OCF algorithm selects a match such that the sum of all serviced queue waiting times is maximized, i.e., $\arg\max_{S}\left(\sum_{i,j} S_{i,j}(n)W_{i,j}(n)\right)$, such that $\sum_{i=1}^{M} S_{i,j}(n) \le 1$ and $\sum_{j=1}^{N} S_{i,j}(n) \le 1$.

Unlike the LQF algorithm, the OCF algorithm does not starve any queue. Every slot unserved HOL cells get older and will eventually become old enough to be served.

4: Stability

To prove that the LQF and OCF algorithms can achieve 100% throughput, we use the notion of *stability*. We define a switch to be stable for a particular arrival process if the expected length of the input queues does not grow without bound, i.e.

$$E\left[\sum_{i,j} L_{i,j}(n)\right] < \infty, \forall n. \quad (1)$$

Definition: *If a switch is stable for all independent and admissible arrivals, then we say that the switch can achieve 100% throughput.*

In this section we prove that the OCF algorithm is stable for all independent arrival processes. But first, we summarize the result from [7] which shows that under these conditions the LQF algorithm is stable.

4.1: Stability of the LQF algorithm

Consider the queue occupancy vector, $\underline{L}(n)$, whose elements contain all of the queue occupancies, as the state vector of an input-queued switch. By choosing a 2nd order Lyapunov function, $V(n) = \underline{L}^T(n)\underline{L}(n)$, it can be shown for LQF that:

$$\mathrm{E}\left[\underline{L}^T(n+1)\underline{L}(n+1) - \underline{L}^T(n)\underline{L}(n) \,|\, \underline{L}(n)\right] \le -\varepsilon\|\underline{L}(n)\| + k, \quad \text{where } k > 0, \varepsilon > 0. \quad (2)$$

Theorem 1: *The LQF algorithm is stable for all admissible and independent arrival processes.*

Proof: The theorem is proved in [7]. ❒

4.2: Stability of the OCF algorithm

Our proof has two stages. First, we prove the stability of the waiting time, and then we show that the stability of the waiting time implies the stability of the queue occupancy.

4.2.1: Basic definitions

Consider Figure 4. $C_{i,j}(n)$ is the HOL cell of $Q_{i,j}$ at slot n which arrived at slot $n - W_{i,j}(n)$, and thus has been waiting in the queue for $W_{i,j}(n)$ slots. If the cell doesn't leave the queue, i.e., doesn't get served, its waiting time increases by one every slot:

$$W_{i,j}(n+1) = W_{i,j}(n) + 1. \quad (3)$$

If the cell leaves the queue, the cell behind advances to the head of the queue, becoming the new HOL cell. From Figure 4, it can be seen that the waiting time of the new HOL is the difference of the previous HOL cell wait-

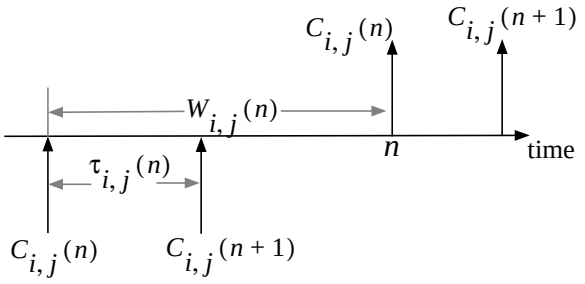


Figure 4: Arrivals and departures time line. Arrivals are shown below the line, departures are shown above the line.

ing time and the interarrival time between the two cells, $\tau_{i,j}(n)$:

$$W_{i,j}(n+1) = W_{i,j}(n) - \tau_{i,j}(n), \quad \forall \tau_{i,j}(n) \le W_{i,j}(n) \quad (4)$$

In the case that there is no arrival while $C_{i,j}(n)$ is waiting in the queue, the queue becomes empty when $C_{i,j}(n)$ leaves the queue. Hence, the waiting time in the next slot is zero:

$$W_{i,j}(n+1) = 0, \qquad \forall \tau_{i,j}(n) > W_{i,j}(n) \quad (5)$$

4.2.2: Theorems

Theorem 2: *Under the OCF algorithm, waiting times are stable for all admissible independent arrival processes, i.e.,* $E[\|\underline{W}(n)\|] < \infty, \forall n$.

Proof: The theorem is proved in Appendix A. In summary, the system is said to be stable if the following condition is met:

$$E\left[\underline{W}^T(n+1)Q\underline{W}(n+1) - \underline{W}^T(n)Q\underline{W}(n) \,\middle|\, \underline{W}(n)\right] \le -\varepsilon\|\underline{W}(n)\| + K \quad (6)$$

where $K > 0$, $\varepsilon > 0$. ❐

In other words, as the size of the waiting time vector increases, the right side of equation (6) becomes more negative, as does the expected single step drift represented by the left side. The negative drift controls the size of the waiting times, hence keeping the system stable.

Theorem 3: *Under the OCF algorithm, queue occupancies are stable for all admissible independent arrival processes, i.e.,* $E[\|\underline{L}(n)\|] < \infty, \forall n$.

Proof: From Fact 2 in Appendix A, any given queue occupancy is always less than or equal to the corresponding waiting time. Therefore, bounded waiting times imply bounded queue occupancies; thus the queue occupancies are stable. ❐

Corollary 4: *The maximum throughput of an input-queued switch under the OCF algorithm is 100%.*

5: Conclusion

Before long, output-queueing will be impracticable for high bandwidth switches. This is because of the need for internal speedup, which will quickly surpass the bandwidth of commercially available memories. Input-queued switches have been unpopular until now because, among other reasons, they have apparently limited throughput.

It has been previously shown that an input-queued switch employing VOQ and using the LQF algorithm can achieve 100% throughput for all admissible independent arrival processes. Unfortunately, the LQF algorithm can lead to starvation of one or more of the input queues. So instead, we have introduced the OCF algorithm that gives preference to cells that have been waiting the longest, and hence will never starve a queue. We have proved that like the LQF algorithm, the OCF algorithms can achieve 100% throughput for all independent arrival processes.

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] Karol, M.; Eng, K.; Obara, H. "Improving the performance of input-queued ATM packet switches," *INFOCOM '92*, pp.110-115.

[3] 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.

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

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

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

[7] McKeown, N.; Anantharam, V.; and Walrand, J.; "Achieving 100% Throughput in an Input-Queued Switch", *Proc.* INFOCOMM '96 (1996),

[8] Tassiulas, L.; Ephremides, A. "Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks," *IEEE Trans. Automatic Control*, Vol. 37, No. 12, pp.1936-1948, Dec. 1992.

Appendix A: Stability proof

A.1 Definitions

1. $Q_{i,j}$ denotes a queue for an arrival at input i for output j.
2. $C_{i,j}(n)$ denotes the HOL cell of $Q_{i,j}$ at slot n.

3. $\tau_{i,j}(n)$ denotes the interarrival time between $C_{i,j}(n)$ and the cell behind it in line.
4. $W_{i,j}(n)$ denotes the waiting time of $C_{i,j}(n)$ at slot *n*.
5. $L_{i,j}(n)$ denotes the occupancy of $Q_{i,j}$ at slot *n*.
6. The service indicator:

$$S_{i,j}(n) = \begin{cases} 1 & \text{if } Q_{i,j} \text{ is served at slot } n \\ 0 & \text{otherwise} \end{cases} \quad (7)$$

7. The waiting time vector:

$$\underline{W}(n) \equiv (W_{1,1}(n), \ldots W_{1,N}(n), \ldots W_{M,1}(n), \ldots W_{M,N}(n)) . \quad (8)$$

8. The queue occupancy vector:

$$\underline{L}(n) \equiv (L_{1,1}(n), \ldots L_{1,N}(n), \ldots L_{M,1}(n), \ldots L_{M,N}(n)) . \quad (9)$$

9. The arrival matrix:

$$\Lambda \equiv \left[\lambda_{i,j}\right], \text{ where } \sum_{i=1}^{M} \lambda_{i,j} \le 1, \sum_{j=1}^{N} \lambda_{i,j} \le 1, \lambda_{i,j} \ge 0 , \quad (10)$$

and associated rate vector:

$$\underline{\lambda} \equiv (\lambda_{1,1}, \ldots \lambda_{1,N}, \ldots \lambda_{M,1}, \ldots \lambda_{M,N}) . \quad (11)$$

10. The interarrival time vector:

$$\underline{\tau}(n) \equiv (\tau_{1,1}(n), \ldots \tau_{1,N}(n), \ldots \tau_{M,1}(n), \ldots \tau_{M,N}(n)) . \quad (12)$$

11. The service vector, indicating which queues are served at slot *n*:

$$\underline{S}(n) \equiv (S_{1,1}(n), \ldots S_{1,N}(n), \ldots S_{M,1}(n), \ldots S_{M,N}(n)) . \quad (13)$$

12. The service matrix, indicating which queues are served at slot *n*:

$$\mathbf{S}(n) \equiv [S_{i,j}(n)] , \quad (14)$$

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

Note that: $\sum_{i=1}^{M} S_{i,j}(n) \le 1, \sum_{j=1}^{N} S_{i,j}(n) \le 1$,

and hence $\mathbf{S}(n) \in \mathbf{S}$ is a *permutation matrix*.

13. The positive-definite diagonal matrix, Q, whose diagonal elements are $\{\lambda_{1,1}, \ldots \lambda_{1,N}, \ldots \lambda_{M,1}, \ldots \lambda_{M,N}\}$.
14. $[\underline{a} \cdot \underline{b} \cdot \underline{c}]$ denotes a vector in which each element is a product of the corresponding elements of the vectors: $\underline{a}$, $\underline{b}$, and $\underline{c}$, i.e., $a_{i,j} \cdot b_{i,j} \cdot c_{i,j}$.
15. The approximate waiting time next-state vector:

$$\tilde{\underline{W}}(n+1) \equiv \underline{W}(n) + \underline{1} - [\underline{S}(n) \cdot \underline{\tau}(n)] . \quad (15)$$

A.2 Analysis

Fact 1: An interarrival time, $\tau_{i,j}(n)$ is independent of a waiting time, $W_{i,j}(n), \forall i, j, n$.

Fact 2: $\tau_{i,j}(n) \ge 1$. Since there is only at most one arrival per slot, the arrival time of any two consecutive cells must be at least one slot apart.

Fact 3: $W_{i,j}(n) \ge L_{i,j}(n), \forall i, j, n$ because there is at most one arrival per slot.

Fact 4: The sum of the weights of all granted requests, $\sum_{(i,j) \in M} W_{i,j}(n)$, is equal to $\underline{W}^T(n)\underline{S}(n)$.

Fact 5: For any queue whose arrival rate is zero, $\lambda_{i,j} = 0$, $L_{i,j}(n) = 0$, thus $W_{i,j}(n) = 0, \forall n$. Considering the fact that a zero waiting time does not contribute to the sum value, $\underline{W}^T(n)\underline{S}(n)$, without loss of generality, we can set the corresponding service indicator, $S_{i,j}(n)$ to zero for all time, $S_{i,j}(n) = 0, \forall n$.

Lemma 1: $\underline{W}^T(n)\underline{\lambda} - \underline{W}^T(n)\underline{S}^*(n) \le 0, \quad \forall \underline{W}(n), \underline{\lambda}$,

where $\underline{S}^*(n)$ *is s.t.* $\underline{W}^T(n)\underline{S}^*(n) = max(\underline{W}^T(n)\underline{S}(n))$.

Proof: Consider the linear programming problem:

$$max(\underline{W}^T(n)\underline{\lambda}) \quad (16)$$

$$\text{s.t. } \sum_{i=1}^{M} \lambda_{i,j} \le 1, \sum_{j=1}^{N} \lambda_{i,j} \le 1, \lambda_{i,j} \ge 0 \quad (17)$$

Λ is a doubly stochastic matrix and forms a convex set, C, with the set of extreme points equal to permutation matrices, S [7]. Therefore, the following are true.

$$max(\underline{W}^T(n)\underline{\lambda}) \le \underline{W}^T(n)\underline{S}^*(n) \ \square \quad (18)$$

Lemma 2:

$$E\left[\tilde{\underline{W}}^T(n+1)Q\tilde{\underline{W}}(n+1) - \underline{W}^T(n)Q\underline{W}(n) \,\middle|\, \underline{W}(n)\right] \le K, \forall \underline{\lambda}$$

where K is a constant.

Proof: By expansion:

$$\begin{aligned} &\tilde{\underline{W}}^T(n+1)Q\tilde{\underline{W}}(n+1) \\ &= (\underline{W}(n) + \underline{1} - [\underline{S}^*(n) \cdot \underline{\tau}(n)])^T Q \\ &\quad (\underline{W}(n) + \underline{1} - [\underline{S}^*(n) \cdot \underline{\tau}(n)]) \\ &= \underline{W}^T(n)Q\underline{W}(n) + 2\underline{W}^T(n)\underline{\lambda} \\ &\quad -2\underline{W}^T(n)[\underline{S}^*(n) \cdot \underline{\tau}(n) \cdot \underline{\lambda}] \\ &\quad + \sum_{i,j} \lambda_{i,j} - 2\sum_{i,j} S^*_{i,j}(n) \cdot \tau_{i,j}(n) \cdot \lambda_{i,j} \\ &\quad + \sum_{i,j} S^*_{i,j}(n) \cdot \tau^2_{i,j}(n) \cdot \lambda_{i,j} \end{aligned} \quad (19)$$

Substracting $\underline{W}^T(n)Q\underline{W}(n)$ from both sides:

$$E\left[\tilde{\underline{W}}^T(n+1)Q\tilde{\underline{W}}(n+1) - \underline{W}^T(n)Q\underline{W}(n)\middle|\underline{W}(n)\right]$$
$$= 2\left(\underline{W}^T(n)\underline{\lambda} - \underline{W}^T(n)\underline{S}^*(n)\right) \quad (20)$$
$$+ \sum_{i,j}\lambda_{i,j} - 2\sum_{i,j}S^*_{i,j}(n) + \sum_{i,j}\frac{S^*_{i,j}(n)}{\lambda_{i,j}}$$

After imposing the admissibility constraints and the scheduling algorithm properties, we obtain the following inequalities:

$$\sum_{i,j}\lambda_{i,j} \le N, \sum_{i,j}S^*_{i,j}(n) \ge 0, \sum_{i,j}\frac{S^*_{i,j}(n)}{\lambda_{i,j}} \le L < \infty, \quad (21)$$

where L is a non-negative constant.

From (20), we obtain:

$$E\left[\tilde{\underline{W}}^T(n+1)Q\tilde{\underline{W}}(n+1) - \underline{W}^T(n)Q\underline{W}(n)\middle|\underline{W}(n)\right]$$
$$\le 2\left(\underline{W}^T(n)\underline{\lambda} - \underline{W}^T(n)\underline{S}^*(n)\right) + L + N \quad (22)$$

Recall Lemma 1, $\underline{W}^T(n)\underline{\lambda} - \underline{W}^T(n)\underline{S}^*(n) \le 0$. Hence, we prove Lemma 2, where $K = L + N > 0$. ❐

Lemma 3:

$E\left[\tilde{\underline{W}}^T(n+1)Q\tilde{\underline{W}}(n+1) - \underline{W}^T(n)Q\underline{W}(n)\middle|\underline{W}(n)\right] \le -\varepsilon\|\underline{W}(n)\| + K$
$\forall\underline{\lambda} \le (1-\beta)\underline{\lambda}_m, 0 < \beta < 1$, *where* $\underline{\lambda}_m$ *is any rate vector such that* $\|\underline{\lambda}_m\|^2 = N$, *and* $\varepsilon > 0$.

Proof:

$$\underline{W}^T(n)\underline{\lambda} - \underline{W}^T(n)\underline{S}^*(n) \le \underline{W}^T(n)(1-\beta)\underline{\lambda}_m - \underline{W}^T(n)\underline{S}^*(n) \quad (23)$$

Applying Lemma 1,

$$\underline{W}^T(n)\underline{\lambda} - \underline{W}^T(n)\underline{S}^*(n) \le -\beta\underline{W}^T(n)\underline{\lambda}_m. \quad (24)$$

$$\underline{W}^T(n)\underline{\lambda} - \underline{W}^T(n)\underline{S}^*(n) \le -\beta\|\underline{W}(n)\| \cdot \|\underline{\lambda}_m\| \cdot \cos\theta, \quad (25)$$

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

Any non-zero waiting time, $W_{i,j}(n) > 0$, can occur iff the corresponding rate is non-zero, $\lambda_{i,j} > 0$. Hence, it is sufficient to say that $\cos\theta > 0$. Furthermore, since $\|\underline{\lambda}\| \le \sqrt{N}$,

$$\cos\theta = \frac{\underline{W}^T(n)\underline{\lambda}}{\|\underline{W}(n)\| \cdot \|\underline{\lambda}\|} \ge \frac{W_{max}(n)\lambda_{min}}{\|\underline{W}(n)\|\sqrt{N}} \quad (26)$$

where $W_{max}(n) = max(W_{i,j}(n))$ and $\lambda_{min} = min(\lambda_{i,j})$.

Since $\|\underline{W}(n)\| \le NW_{max}$,

$$\cos\theta \ge \frac{\lambda_{min}}{N\sqrt{N}} \quad (27)$$

Using equations (22), (25), and (27).

$$E\left[\tilde{\underline{W}}^T(n+1)Q\tilde{\underline{W}}(n+1) - \underline{W}^T(n)Q\underline{W}(n)\middle|\underline{W}(n)\right]$$
$$\le -2\beta\frac{\lambda_{min}}{N\sqrt{N}}\|\underline{W}(n)\| + K \quad (28)$$

where $\varepsilon = 2\beta\frac{\lambda_{min}}{N\sqrt{N}}$. ❐

Lemma 4: $E\left[\underline{W}^T(n+1)Q\underline{W}(n+1) - \underline{W}^T(n)Q\underline{W}(n)\middle|\underline{W}(n)\right] \le -\varepsilon\|\underline{W}(n)\| + K$

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

Proof: We can draw the following relationship between the two waiting times:

$$W_{i,j}(n+1) = \begin{cases}\tilde{W}_{i,j}(n+1), & \tilde{W}_{i,j}(n+1) \ge 0\\ 0, & \tilde{W}_{i,j}(n+1) < 0\end{cases}. \quad (29)$$

Since Q is a positive definite matrix, (29) implies:

$$\underline{W}^T(n+1)Q\underline{W}(n+1) \le \tilde{\underline{W}}^T(n+1)Q\tilde{\underline{W}}(n+1), \forall n. \quad (30)$$

Hence,

$$E\left[\underline{W}^T(n+1)Q\underline{W}(n+1) - \underline{W}^T(n)Q\underline{W}(n)\middle|\underline{W}(n)\right]$$
$$\le E\left[\tilde{\underline{W}}^T(n+1)Q\tilde{\underline{W}}(n+1) - \underline{W}^T(n)Q\underline{W}(n)\middle|\underline{W}(n)\right]. \quad (31)$$

This proves Lemma 4. ❐

Lemma 5: *There exists a quadratic Lyapunov function,* $V(\underline{W}(n))$ *such that:*

$$E\left[V(\underline{W}(n+1)) - V(\underline{W}(n))\middle|\underline{W}(n)\right] \le -\varepsilon\|\underline{W}(n)\| + K. \quad (32)$$

where K is a constant and $\varepsilon > 0$.

Proof: From Lemma 4, $V(\underline{W}(n)) = \underline{W}^T(n)Q\underline{W}(n)$ $\varepsilon = 2\beta\frac{\lambda_{min}}{N\sqrt{N}} > 0$, and $K = L + N > 0$. ❐

Theorem 2: *Under the OCF algorithm, the waiting times are stable for all admissible and independent arrival processes, i.e.,* $E[\|\underline{W}(n)\|] \le C < \infty$.

Proof: Since there exists a quadratic Lyapunov function $V(\underline{W}(n))$, such that (32) is satisfied, the switch is stable. ❐

Theorem 3: *Under the OCF algorithm, the queue occupancies are stable for all admissible and independent arrival processes, i.e.,* $E[\|\underline{L}(n)\|] \le C < \infty$.

Proof: From Fact 3: $W_{i,j}(n) \ge L_{i,j}(n), \forall i, j, n$. Thus,

$$E[\|\underline{L}(n)\|] \le E[\|\underline{W}(n)\|] \le C < \infty. \quad (33)$$

Hence, $E[\|\underline{L}(n)\|]$ is also bounded above by C. ❐