Scheduling Input-Queued ATM Switches with QoS Features

Shizhao Li
New Jersey Institute of Technology
New Jersey Research Center for Digital Radio
Department of Electrical & Computer Engineering
University Heights, Newark, NJ 07102, U.S.A.
sxl3576@megahertz.njit.edu

Nirwan Ansari
Department of Information Engineering
The Chinese University of Hong Kong
Sha Tin, N.T., Hong Kong
nansari@ie.cuhk.edu.hk
(on leave from NJIT)

Abstract

The input-queued switching architecture is becoming the alternative architecture for high speed switches owing to its scalability. Tremendous amount of effort has been made to overcome the throughput problem caused by head of line blocking and the contentions occurred at input and output sides of a switch. Existing algorithms only aim at improving throughput but inadvertently ignore undesired effects on the traffic shape and quality of service features such as delay and fairness. In this paper, a new algorithm, referred to as longest normalized queue first, is introduced to improve upon existing algorithms in terms of delay, fairness and burstiness. The proposed algorithm is proven to be stable for all admissible traffic patterns. Simulation results confirm that the algorithm can smooth the traffic shape, and provide good delay property as well as fair service.

1. Introduction

The input-queued switching architecture has been adopted for high speed switch implementation owing to its scalability. A major problem with this architecture is the head-of-line blocking (HOL) [5], which limits the throughput of an input-queued switch to 58.6% under Bernoulli traffic when a single FIFO queue is used in each input.

Previous research has shown that the throughput of an input-queued switch can be improved by using well designed buffering schemes and scheduling algorithms. The HOL blocking can be completely eliminated by adopting virtual output queueing, in which multiple virtual output queues (VOQs) directed to different outputs are maintained at each input. However, the contentions occurred at the input and output sides of a switch still limit the throughput. Moreover, when more than one cell can be accessed by the scheduler in one input, selecting different cells for transmission could lead to different throughput, owing to the interdependence of the inputs.

Maximizing the throughput is similar to the matching problem in a bipartite graph [3]. An iterative algorithm called *i*SLIP, which is a maximum size matching based scheduler, can achieve 100% throughput for independent and uniform traffic [8]. Round robin scheduler, which has low implementation complexity, is adopted in *i*SLIP to resolve the contentions at both input and output sides of the switch. However, the priority of a round robin scheduler is not a function of the queue length. Thus, *i*SLIP performs poorly for non-uniform traffic, in which the average queue length of the VOQs could differ strikingly under loaded traffic. Maximum weight matching can achieve high throughput under both uniform and non-uniform traffic in which each session is assigned a weight and a match with the maximum aggregate weight is obtained. Longest queue first (LQF) [7] and oldest cell first (OCF) [9] are among the maximum weight matching approach, in which the queue length and the delay time of head of line cell are set as the weights, respectively.

Algorithms which only aim at maximizing throughput could generate adverse effects on traffic shape and quality of service (QoS) features such as delay and fairness. In LQF, the priority is set according to the queue length, i.e., the queue with the largest length has the highest priority to receive service. Since the queues of the VOQs with different arrival rates are built up at different speeds, using the queue length as the weight forces the scheduler to serve the VOQs with high arrival rates and starve the VOQs with low arrival rates. This is the main reason why LQF leads to unfair service and uncontrollable delay time for the VOQs with low arrival rates. On the other hand, OCF avoids starvation by setting delay time as the weight, in which the unserved cells get growing "older" until they eventually become "old" enough to be served. Using delay time as the weight forces the scheduler to serve the VOQs burst by burst, thus sacrificing the QoS requirements. By observing

0-8186-9014-3/98 $10.00

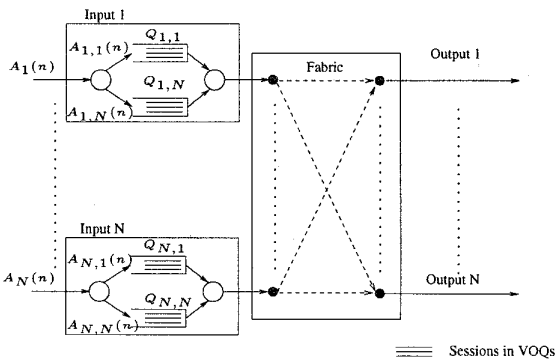

Figure 1. Input-queued switch model

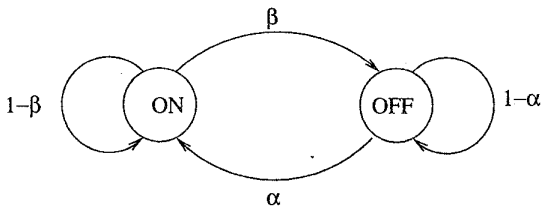

Figure 2. Simple ON-OFF traffic model

that session rates should be incorporated in the scheduler design in order to satisfy the QoS requirements, we propose a new algorithm, referred to as longest normalized queue first (LNQF), which performs better than LQF and OCF in terms of delay, fairness and burst reduction.

The rest of the paper is organized as follows. In Section 2, we describe our switch and traffic models. Section 3 presents the proposed algorithm. The stability of the switch using the proposed algorithm is derived in Section 4. Section 5 shows the performance of the proposed algorithm. Concluding remarks are given in Section 6.

2. Switch and traffic models

Consider an $N \times N$ input-queued ATM switch consisting of N inputs, N outputs and a non-blocking switch fabric. To eliminate the HOL blocking, virtual output queueing is adopted, in which N virtual output queues (VOQs) directed to N different outputs are maintained at each input, as shown in Figure 1.

Let $Q_{i,j}$ denote the VOQ directed to output j at input i, and $A_{i,j}$ denote the arrival process to $Q_{i,j}$. To provide QoS features, the switch resources, i.e., the bandwidth and storage should be allocated on a per-session basis. There could be more than one session arrived at a certain input directed to the same output. Thus, multiple sessions could share the same VOQ, in which each session is maintained as an FIFO queue. Let $I_{i,j,k}$ be the kth session in $Q_{i,j}$ with arrival rate $\lambda_{i,j,k}$. Therefore, the arrival rate of $A_{i,j}$ can be expressed as $\lambda_{i,j} = \sum_k \lambda_{i,j,k}$. An arrival process A_i, which is the aggregate arrival process to input i, is said to be uniform if $\lambda_{i,m} = \lambda_{i,n}, \forall\, m \neq n,\ 1 \leq m,n \leq N$. Otherwise, the process is said to be non-uniform. The traffic pattern is admissible if and only if $\sum_{j=1}^{N} \lambda_{i,j} \leq 1$ and $\sum_{i=1}^{N} \lambda_{i,j} \leq 1$.

The traffic in a real network is highly correlated from cell to cell, and cells tend to arrive at the switch in "bursts." One way of modeling a bursty source is by using an ON-OFF model in the discrete-time domain. This model is equivalent to a two-state Markov Modulated Deterministic Process (MMDP)[2]. The two states, OFF state and ON state, are shown in the Figure 2. In the OFF state, the source does not send any cells. In the ON state, the source sends data cells at the peak cell rate (P). The source can independently shift from one state to another as shown in Figure 2. In a discrete-time domain, state changes may occur only at the end of a time-slot. At each time slot, the source in the OFF state changes to the ON state with a probability α. Similarly, the source in the ON state changes to the OFF state with a probability β. It must be remembered that there is no correlation between the two probabilities. The probabilities of the source being in the OFF state and ON state are given by $P_{off} = \frac{\beta}{(\alpha+\beta)}$ and $P_{on} = \frac{\alpha}{(\alpha+\beta)}$, respectively.

The bursty source is characterized by the peak cell rate (P), the average cell rate (A) and the average number of cells per burst (B). The burstiness of the traffic is defined as the ratio of the peak cell rate and average cell rate. Given these parameters, the state transition probabilities can be computed as $\alpha = \frac{A}{B(P-A)}$ and $\beta = \frac{1}{B}$.

3. The LNQF algorithm

The basic objective of scheduling an input-queued switch is to find a contention free match based on the connection requests. At the beginning of every time slot, each input sends requests to the scheduler. The scheduler selects a match between the inputs and outputs with the constraints of unique pairing, i.e., at most one input can be matched to each output and vice versa. At the end of the time slot, a cell is transmitted per matched input-output pair.

Finding a contention free match between inputs and outputs is equivalent to solving a bipartite graph matching problem, as shown in Figure 3(a). Each vertex on the left side represents an input, and that on the right side represents an output. An edge connects input vertex i and output vertex j if there are cells stored in $Q_{i,j}$. Associated with each edge is a weight, which is defined differently by different algorithms. For example, setting weight as the queue length

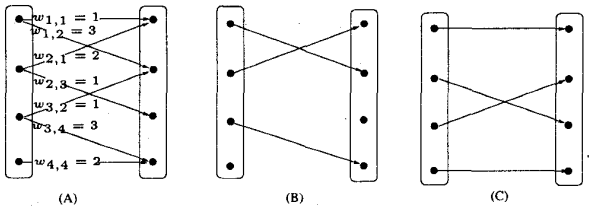

Figure 3. A Bipartite graph matching example:(a) the request graph, (b) a maximum weight match, and (c) a maximum size match.

of the VOQ leads to LQF, and setting weight as the delay time of the head cell in the VOQ leads to OCF. A maximum weight matching algorithm computes a match which can maximize the aggregate weight. Note that maximum size matching in which the number of connections between the inputs and outputs is maximized is a special case of maximum weight matching with the weights of the non-empty VOQs set to 1 and those of empty VOQs set to 0, respectively. A maximum weight match and a maximum size match for the same request graph can be different as shown in Figure 3(b) and 3(c), respectively. It was shown that maximum size matching is not stable for non-uniform traffic [7]. Therefore, maximum weight matching is adopted in LNQF.

Denote $l_{i,j,k}(n)$ as the length of the FIFO queue corresponding to session $I_{i,j,k}$ in $Q_{i,j}$ and $l_{i,j}(n) = \sum_k l_{i,j,k}(n)$ denote the length of $Q_{i,j}$ at time slot n. In LNQF, the weight of a VOQ is set to the normalized queue length which is the total queue length of the VOQ divided by its rate, i.e., $w_{i,j}(n) = \frac{l_{i,j}(n)}{\lambda_{i,j}(n)}$. Let $\underline{W}_i(n) = (w_{i,1}(n), w_{i,2}(n), \ldots, w_{i,N}(n))^T$ be the weight vector of input i and $S = [S_{i,j}(n)]$ be the service matrix which indicates the match between inputs and outputs. $S_{i,j}(n)$ is set to 1 if input i is scheduled to transmit a cell to output j. Otherwise, $S_{i,j}(n)$ is set to 0. Let $\underline{S}_i(n) = (S_{i,1}(n), S_{i,2}(n), \ldots, S_{i,N}(n))^T$ be the service vector associated with input i. The LNQF scheduler, as shown in Figure 4, performs the following for each time slot n:

1. Each input i computes the normalized queue length of each VOQ, sets it as the weight of the VOQ, and sends the weight vector $\underline{W}_i(n)$ to the scheduler;

2. The scheduler searches for a match that achieves the maximum aggregate weight under the constraint of unique pairing, i.e.,

$$\arg\max_{S}[\sum_{i,j} S_{i,j}(n) w_{i,j}(n)]$$

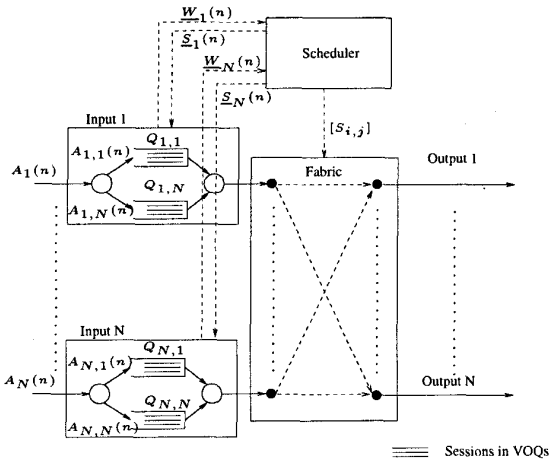

Figure 4. LNQF scheduler

such that $\sum_i S_{i,j}(n) = \sum_j S_{i,j}(n) = 1$, sends the service vector $\underline{S}_i(n)$ to the corresponding input, and uses the service matrix $[S_{i,j}(n)]$ to configure the fabric;

3. Each input i computes the normalized queue length of each session in the matched VOQ indicated by $\underline{S}_i(n)$, and selects the session with the longest normalized queue length for transmission.

The LNQF algorithm gives preference to the VOQs with large normalized queue lengths for transmission. Note that the average queue length of a VOQ in a fair server should be proportional to its arrival rate. Using the normalized queue length as the weight forces the scheduler to serve VOQs more fairly, thus preventing VOQs with slow arrival rate from starvation. In addition, using normalized queue length as the weight allows cells arrived later to have higher weights than cells which come earlier, therefore performing burst reduction.

4. Stability of LNQF

To prove that LNQF is stable, the stability of a switch is first defined.

Definition 1 *A switch is stable if and only if the expected queue length in the switch does not increase without bound, i.e.,*

$$E[\sum_{i,j} l_{i,j}(n)] < \infty, \quad \forall n.$$

Theorem 1 *The switch using LNQF is stable for all admissible traffic patterns.*

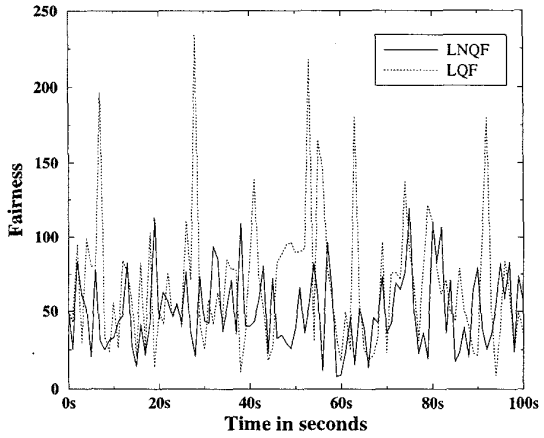

Figure 5. Fairness of LNQF versus LQF

Proof: We have adopted an approach similar to [7] to prove the stability of LNQF. The complete proof is given in the appendix, and the main idea is to show that the quadratic function of the queue length vector has a negative expected single-step drift when the total queue length is large.

5. Performance

A 4×4 input-queued switch was considered for simulations in which the bursty traffic was generated based on the on-off traffic model. The average burst length was chosen to be 20 cells and the burstiness was 2. The traffic was non-uniform, i.e., the arrival rates of the VOQs in the same input were different, and were 0.5, 1, 2, and 5Mbps. Two sessions in each VOQ, a fast session with a rate four times that of a slow session, were generated. A traffic load of 0.95 was assumed, and each simulation lasted through 100 seconds.

The fairness of LNQF and LQF is compared in Figure 5, where the fairness is defined as [4]:

$$F = \max_{\forall i,j,\ j \neq i} \left| \frac{W_i(t_1,t_2)}{\lambda_i} - \frac{W_j(t_1,t_2)}{\lambda_j} \right|,$$

where $W_i(t_1,t_2)$ is the number of cells delivered for session i during the time interval $[t_1,t_2]$, and λ_i is the rate of session i. The fairness is the maximum difference of the normalized service time, which is the service a session received normalized by its rate, among all sessions. It provides a metric on how fair a server is. The smaller the amount of fairness, the fairer the server is. The fairness of a perfectly fair server is 0. As shown in the figure, the average fairness of LNQF is smaller than that of LQF, and so is the variation of the fairness.

Table 1 summarizes the performance comparison among LNQF, LQF and OCF. Note that LQF tends to starve slow sessions in which the average delays are much higher than the faster sessions, while LNQF provides comparable delay for each session. The average time for OCF to complete transmitting a burst is much smaller than that for LNQF, implying that LNQF is a better "traffic shaper" in reducing burstiness.

schedulers	LNQF	LQF	OCF
$d_{1,1}$ (time slot) λ=0.1Mbps	91.5	1049.6	147.8
$d_{1,2}$ (time slot) λ=0.4Mbps	137.5	463.0	144.6
$d_{2,1}$ (time slot) λ=0.2Mbps	110.1	644.1	129.3
$d_{2,1}$ (time slot) λ=0.8Mbps	130.0	213.4	133.7
$d_{3,1}$ (time slot) λ=0.4Mbps	112.8	328.6	129.0
$d_{3,2}$ (time slot) λ=1.6Mbps	129.2	103.7	132.2
$d_{4,1}$ (time slot) λ=1Mbps	128.2	149.7	133.6
$d_{4,2}$ (time slot) λ=4Mbps	139.5	44.2	137.0
fairness	50.3	68.1	50.5
transmission time(time slot)/burst:	135.6	138.1	92.4

$d_{i,j}$: average delay of the jth session in VOQ i.

Table 1. Statistics of the simulation results

6. Conclusions

We have proposed a new algorithm, LNQF, to improve upon existing algorithms in terms of delay, fairness, and burstiness. We have also proved that LNQF is stable under all admissible traffic patterns and demonstrated through simulations that LNQF is a starvation-free algorithm which can provide better fairness than LQF, and better effectiveness in smoothing traffic than OCF.

References

[1] G. Birkhoff. Three observations on linear algebra. *Rev. Univ. Nac. Tucumán, Ser. A.*, 5:147–151, 1946.

[2] R. Bolla, F. Davoli, and M. Marchese. Evaluation of a cell loss rate computation method in ATM multiplexers with multiple bursty sources and different traffic classes. In *Proceedings of IEEE GLOBECOM*, pages 437–441, Nov. 1996.

[3] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. New York: McGraw Hill, 1989.

[4] S. Golestani. A self-clocked fair queueing scheme for broadband applications. In *Proceedings of IEEE INFOCOM*, pages 636–646, 1994.

[5] M. J. Karol, M. G. Hluchyj, and S. P. Morgan. Input versus output queueing on a space-division packet switch. *IEEE Transactions on Communications*, COM-35:1347–1356, Dec. 1987.

[6] P. R. Kumar and S. P. Meyn. Stability of queueing networks and scheduling policies. *IEEE Transactions on Automatic Control*, 40(2):251–260, Feb. 1995.

[7] N. McKeown, V. Anantharam, and J. Walrand. Achieving 100% throughput in an input-queued switch. In *Proceedings of IEEE INFOCOM*, pages 296–302, 1996.

[8] N. McKeown, J. Walrand, and P. Varaiya. Scheduling cells in an input-queued switch. *IEE Electronics Letters*, pages 2174–75, Dec. 9th 1993.

[9] A. Mekkittikul and N. McKeown. A starvation-free algorithm for achieving 100% throughput in an input-queued switch. In *Proceedings of the ICCCN*, Oct. 1996.

7. Appendix

Several definitions and lemmas, similar to [7], will first be defined and proven in order to facilitate the proof of stability of LNQF.

Definition 2 *The rate matrix is defined as:*

$$\Lambda = [\lambda_{i,j}]$$

where

$$\lambda_{i,j} \geq 0, \quad \sum_{i=1}^{N} \lambda_{i,j} \leq 1, \quad \sum_{j=1}^{N} \lambda_{i,j} \leq 1.$$

Definition 3 *The rate vector associated with the rate matrix Λ is defined as:*

$$\underline{\lambda} = (\lambda_{1,1}, \ldots \lambda_{1,N}, \ldots \lambda_{N,1}, \ldots \lambda_{N,N})^T$$

Definition 4 *The arrival vector representing the arrivals to the VOQs is defined as:*

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

where $A_{i,j}$ represents the number of cells arrived at the $Q_{i,j}$ at time n.

Definition 5 *The service matrix indicating the match between inputs and outputs is defined as:*

$$S(n) = [S_{i,j}(n)],$$

where

$$S_{i,j}(n) = \begin{cases} 1 & \text{if } Q_{i,j} \text{ is selected for service at time } n, \\ 0 & \text{if } Q_{i,j} \text{ is not selected for service at time } n. \end{cases}$$

Since $\sum_{i=1}^{N} S_{i,j}(n) = \sum_{j=1}^{N} S_{i,j}(n) = 1$, the service matrix is a permutation matrix.

Definition 6 *The service vector corresponding to the service matrix is defined as:*

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

Definition 7 *The queue length vector representing the queue length of the VOQs at time n is defined as:*

$$\underline{L}(n) = (L_{1,1}(n), \ldots L_{1,N}(n), \ldots L_{N,1}(n), \ldots L_{N,N}(n))^T$$

where $L_{i,j}(n)$ represents the queue length of the $Q_{i,j}$ at time n.

Definition 8 *The normalization matrix R is defined as:*

$$R = diag[\lambda_{1,1}^{-1}, \ldots \lambda_{1,N}^{-1}, \ldots \lambda_{N,1}^{-1}, \ldots \lambda_{N,N}^{-1}]$$

Definition 9 *The approximate next state vector of queue length is defined as:*

$$\underline{\hat{L}}_{i,j} = (L_{1,1}(n+1), \ldots L_{1,N}(n+1), \ldots L_{N,1}(n+1), \ldots L_{N,N}(n+1))^T$$

where

$$\hat{L}_{i,j}(n+1) = L_{i,j}(n) - S_{i,j}(n) + A_{i,j}(n),$$

$\hat{L}_{i,j}(n+1)$ approximates the exact next state queue length of $Q_{i,j}$,

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

where $[x]^+ = \max\{0, x\}$.

Fact 1 *(Birkhoff's Theorem)[1] The doubly sub-stochastic N×N square matrices form a convex set, C, with the set of extreme points equal to permutation matrices.*

Lemma 1 $\underline{L}^T(n)(\underline{1} - R\underline{S}^*(n)) \leq 0, \forall(\underline{L}(n), R)$, *where $\underline{S}^*(n)$ is the match (solution) with the maximum weight.*

Proof: Consider a linear programming problem as follows:

$$\max_{\underline{\lambda}}(\underline{L}^T(n)R\underline{\lambda}) \;\; s.t. \;\; \lambda_{i,j} \geq 0, \sum_{i=1}^{N} \lambda_{i,j} \leq 1, \sum_{j=1}^{N} \lambda_{i,j} \leq 1$$

From **Fact 1** we know that a doubly sub-stochastic matrix Λ forms a convex set, which has extreme points indicated by permutation matrices. The above linear programming problem has a solution at the extreme points of the convex set. Therefore,

$$\begin{aligned} \max(\underline{L}^T(n)R\underline{\lambda}) &\leq \max(\underline{L}^T(n)R\underline{S}(n)) \\ &= \underline{L}^T(n)R\underline{S}^*(n) \end{aligned} \quad (1)$$

Thus,

$$\underline{L}^T(n)(\underline{1} - R\underline{S}^*(n)) \leq 0 \quad (2)$$

Lemma 2

$$E[\underline{\hat{L}}^T(n+1)R\underline{\hat{L}}(n+1) - \underline{L}^T(n)R\underline{L}(n) \mid \underline{L}(n)] \leq M + L,$$

where M and L are positive constants.

Proof:

$$\begin{aligned}
& \hat{\underline{L}}^T(n+1)R\hat{\underline{L}}(n+1) - \underline{L}^T(n)R\underline{L}(n) \\
= \; & [\underline{L}(n) + \underline{A}(n) - \underline{S}(n)]^T R[\underline{L}(n) + \underline{A}(n) - \underline{S}(n)] \\
& -\underline{L}^T(n)R\underline{L}(n) \\
= \; & 2\underline{L}^T(n)R[\underline{A}(n) - \underline{S}(n)] + \underline{A}^T(n)R\underline{A}(n) \\
& -2\underline{A}^T(n)R\underline{S}(n) + \underline{S}^T(n)R\underline{S}(n) \qquad (3)
\end{aligned}$$

After taking expectation of Equation (3),

$$\begin{aligned}
& E[\hat{\underline{L}}^T(n+1)R\hat{\underline{L}}(n+1) - \underline{L}^T(n)R\underline{L}(n) \mid \underline{L}(n)] \\
= \; & 2\underline{L}^T(n)(\underline{1} - R\underline{S}^*(n)) + \underline{\lambda}^T R\underline{\lambda} + \underline{S}^{*T}(n)R\underline{S}^*(n) \\
& -2\underline{\lambda}^T R\underline{S}^*(n) \\
= \; & 2\underline{L}^T(n)(\underline{1} - R\underline{S}^*(n)) + \sum_{i,j}\lambda_{i,j} + \sum_{i,j}\frac{s_{i,j}(n)}{\lambda_{i,j}} \\
& -2\sum_{i,j} S_{i,j}(n) \\
\leq \; & 2\underline{L}^T(n)(\underline{1} - R\underline{S}^*(n)) + M + L \qquad (4) \\
\leq \; & M + L
\end{aligned}$$

Thus,

$$E[\hat{\underline{L}}^T(n+1)R\hat{\underline{L}}(n+1) - \underline{L}^T(n)R\underline{L}(n) \mid \underline{L}(n)] \leq M + L$$

where $M = \sum_{i,j}\lambda_{i,j} \geq 0$ and $L = \sum_{i,j}\frac{s_{i,j}(n)}{\lambda_{i,j}} \geq 0$.

Lemma 3

$$\begin{aligned}
& E[\hat{\underline{L}}^T(n+1)R\hat{\underline{L}}(n+1) - \underline{L}^T(n)R\underline{L}(n) \mid \underline{L}(n)] \\
\leq \; & -\varepsilon \parallel \underline{L}(n) \parallel + M + L, \qquad (5)
\end{aligned}$$

where ε, M and L are positive constants.

Proof: For any rate vector $\underline{\lambda}$, a vector $\underline{\lambda}^u$ can be found to satisfy the following conditions:

$$(1-\beta)\underline{\lambda}^u = \underline{\lambda}$$

$$\sum_{i=1}^{N}\lambda_{i,j}^u \leq 1, \quad \sum_{j=1}^{N}\lambda_{i,j}^u \leq 1, \quad \lambda_{i,j}^u \geq 0, \quad \forall i,j$$

where $0 \leq \beta \leq 1$

Thus, the first term of equation (4) can be expressed as:

$$\begin{aligned}
& \underline{L}^T(n)R(\underline{\lambda} - \underline{S}^*(n)) \\
= \; & \underline{L}^T(n)R((1-\beta)\underline{\lambda}^u - \underline{S}^*(n)) \\
= \; & \underline{L}^T(n)R(\underline{\lambda}^u - \underline{S}^*(n)) - \underline{L}^T(n)R(\beta\underline{\lambda}^u) \\
\leq \; & -\beta\underline{L}^T(n)R\underline{\lambda}^u \\
= \; & -\frac{\beta}{1-\beta}\underline{L}^T(n)R\underline{\lambda} \\
= \; & -\frac{\beta}{1-\beta} \parallel \underline{L}(n) \parallel\parallel \underline{1} \parallel \cos\theta \\
= \; & -\frac{N\beta}{1-\beta} \parallel \underline{L}(n) \parallel \cos\theta
\end{aligned}$$

Since $L_{i,j}(n) \geq 0, \forall i,j$

$$\cos\theta = \frac{\underline{L}(n)\underline{1}}{\parallel \underline{L}(n) \parallel\parallel \underline{1} \parallel} \geq 0$$

From equation (4),

$$\begin{aligned}
& E[\hat{\underline{L}}^T(n+1)R\hat{\underline{L}}(n+1) - \underline{L}^T(n)R\underline{L}(n) \mid \underline{L}(n)] \\
\leq \; & -\frac{2N\beta}{1-\beta}\cos\theta \parallel \underline{L}(n) \parallel + M + L \qquad (6)
\end{aligned}$$

Let $\varepsilon = \frac{2N\beta}{1-\beta}\cos\theta$, Lemma 3 is proved.

Lemma 4

$$\begin{aligned}
& E[\underline{L}^T(n+1)R\underline{L}(n+1) - \underline{L}^T(n)R\underline{L}(n) \mid \underline{L}(n)] \\
\leq \; & -\varepsilon \parallel \underline{L}(n) \parallel + M + L + \frac{N}{\lambda_{min}}, \qquad (7)
\end{aligned}$$

where ε, M, L and N are positive constants.

Proof: The exact next state queue length is:

$$L_{i,j}(n+1) = [L_{i,j}(n) - S_{i,j}(n)]^+ + A_{i,j}(n) \quad (8)$$

From Definition (9), the approximation of (8) becomes

$$\hat{L}_{i,j}(n+1) = L_{i,j}(n) - S_{i,j}(n) + A_{i,j}(n) \qquad (9)$$

Since $S_{i,j}(n)$ is either 0 or 1, the approximated next state queue length has the following relation with the exact next state queue length:

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

Thus,

$$\underline{L}^T(n+1)R\underline{L}(n+1) - \hat{\underline{L}}^T(n+1)R\hat{\underline{L}}(n+1) \leq \frac{N}{\lambda_{min}}$$

where $\lambda_{min} = \min\{\lambda_{i,j}\}$.

From Lemma 3,

$$\begin{aligned}
& E[\underline{L}^T(n+1)R\underline{L}(n+1) - \underline{L}^T(n)R\underline{L}(n) \mid \underline{L}(n)] \\
\leq \; & -\varepsilon \parallel \underline{L}(n) \parallel + \frac{N}{\lambda_{min}} + M + L \qquad (10)
\end{aligned}$$

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

$$E[V(\underline{L}(n+1)) - V(\underline{L}(n)) \mid \underline{L}(n)] \leq -\varepsilon \parallel \underline{L}(n) \parallel + K$$

where ε and K are positive constant.

Proof: Lemma 5 follows from Lemma 4 by letting $V(\underline{L}(n)) = \underline{L}^T(n)R\underline{L}(n)$ and $K = M + L + \frac{N}{\lambda_{min}}$.

From Lemma 5, the quadratic Lyapunov function of the queue length vector has a negative drift when $\parallel \underline{L}(n) \parallel$ is large. According to [6], Theorem 1 is proved.