On Tracking the Behaviour of an Output-Queued Switch using an Input-Queued Switch with Unity Speedup

Amir Gourgy\textsuperscript{1}, \textit{Student Member, IEEE}, Ted H. Szymanski\textsuperscript{1}, \textit{Member, IEEE}, and Douglas G. Down\textsuperscript{2}, \textit{Senior Member, IEEE}

\textsuperscript{1}Department of Electrical and Computer Engineering

\textsuperscript{2}Department of Computing and Software

McMaster University

Hamilton, Ontario, CANADA

\textbf{Abstract}

We address the problem of fair scheduling of packets in Internet routers with input-queued (IQ) switches and unity speedup. Scheduling in IQ switches is formulated as \textit{tracking} the behaviour of an output-queued (OQ) switch that provides optimal performance. We present the notion of \textit{“lag”} as a performance metric that measures the difference between a packet’s departure time in an IQ switch over that
provided by an OQ switch. We prove that per packet mean lag is bounded for a maximum weight matching scheduling policy that uses lag values for its weights and derive a bound on the mean lag value using a Lyapunov function technique. Furthermore, we propose a simple heuristic tracking scheduling policy and evaluate its performance by simulation.

1 INTRODUCTION

There is a tremendous demand for Internet core nodes to provide quality-of-service (QoS) guarantees for multimedia services, and to provide high switching capacity that makes use of the virtually unlimited bandwidth of optical fibers. The Internet’s success depends on the deployment of high-speed switches and routers that meet these two demands.

On the one hand, the demand of QoS guarantees can be met using output-queued (OQ) switches, which can provide optimal throughput. In addition, much research effort, considering algorithms such as the weighted fair queuing (WFQ) family (e.g., [1]) has been devoted to packet scheduling at output ports to support fair bandwidth sharing that provides delay bounds for regulated traffic. However, output queueing for an $N \times N$ switch requires the switching fabric and memory to run up to $N$ times faster than the line rate; unfortunately, for large or for high-speed data lines, memories with sufficient bandwidth are not available. On the other hand, the fabric and the memory
of an input-queued (IQ) switch need only to run as fast as the line rate. This property makes input queueing very appealing for switches with fast line rates or with a large number of ports. But IQ switching can suffer from head-of-line (HOL) blocking, which limits the throughput to just 58.6%, if each input maintains a single FIFO queue [2]. One method that has been proposed to reduce HOL blocking is to increase the speedup of a switch. A switch with a speedup of \( S \) can remove up to \( S \) packets from each input and deliver up to \( S \) packets to each output within a time slot, where a time slot is the time between packet arrivals at an input port.

A theoretical result [3] established that an \( N \times N \) combined input-and output-queued (CIOQ) switch with a speedup of two could exactly emulate an \( N \times N \) OQ switch for any traffic pattern of input cells. Emulation occurs at every time instance if, under identical inputs both systems produce identical departures. Unfortunately, the complexity of the scheduling algorithm presented in [3] renders OQ switch emulation infeasible (see [4], [5] for a discussion of the complexity). The speedup requirement translates to a smaller time available for the execution of the arbitration algorithm. In a hardware implementation, reduction of the available time by a factor of two poses a substantial problem, although the difference does not seem significant asymptotically; it translates to a requirement of doubling the operating frequency of the arbiter, which might not be practically achievable. The tradeoff between the delay and speedup in a CIOQ switch has been analyzed in [6]. Furthermore, Minkenberg [7] has shown that exact emulation of an
OQ using a CIOQ switch is possible only if the CIOQ switch has infinite output buffers.

Most commercial high-performance switches and routers (e.g., CISCO 1200 [8], BBN [9], Lucent Cajun [10] family, or Avici TSR45000 [11]) use IQ switches. Most of these high-speed switches are built around a crossbar switch that is configured using a centralized scheduler designed to provide high throughput and use a fixed-length cell as a transfer unit. Fixed-length switching technology is widely accepted for achieving high switching efficiency such that variable-length packets are segmented into fixed-length cells at the inputs and are reassembled at outputs. We assume fixed-length cell scheduling for the remainder of this paper.\(^1\)

We consider scheduling policies in an IQ-crossbar switch with a unity speedup. Given that an IQ switch requires at least a speedup of two to exactly emulate an OQ switch [3], an IQ scheduling policy with a unity speedup can not exactly emulate the behaviour of an OQ switch, under all possible traffic patterns. Consequently, we formulate scheduling in an IQ switch as the problem of tracking an OQ switch. We propose the “lag” as a performance metric that measures the difference between a packet’s departure time in an IQ switch over that provided by an OQ switch. We present an IQ scheduling policy with unity speedup for which the lag is bounded and derive a bound on the mean lag value per packet. Furthermore, we propose a simple heuristic tracking scheduling policy and evaluate its

\(^1\)The words packet and cell are used interchangeably for the remainder of this paper.
performance by simulation. Although in this paper we describe the case of tracking an OQ switch implementing only a FIFO scheduling policy, our results can be easily extended for other nonanticipative (decisions do not depend on future arrivals) scheduling policies.

This paper is organized as follows. Section 2 formulates scheduling in an IQ switch with unity speedup as tracking the behaviour of an OQ switch. Section 3 provides motivation for tracking the behaviour of an OQ switch and discusses related work. In Section 4, we present two scheduling policies for tracking the behaviour of an OQ switch. First, we present a scheduling policy called maximum weighted lag (MWL). We prove that the mean lag value is bounded for MWL and derive an upper bound on its value using a Lyapunov function technique. The MWL scheduling policy has a high implementation cost, but serves as a solid base for developing other practical scheduling policies that approximate its performance. Consequently, we present a simpler heuristic tracking policy that can be readily implemented in hardware. The performance of the proposed scheduling policies is evaluated by simulation in Section 6. Section 7 provides our conclusions.

2 PROBLEM FORMULATION

We consider an $N \times N$ OQ switch that uses scheduling policy $\Pi_{OQ}$ and an IQ switch with unity speedup that uses scheduling policy $\Pi_{IQ}$. For an $N \times N$ switch, we use the following notational conventions: $i$ an input, $1 \leq i \leq N$; $j$
an output, \( 1 \leq j \leq N \); \( Q_{i,j} \) is the VOQ at input \( i \) and buffers cells destined for output \( j \); \( HOL_{i,j} \) is the head-of-line cell at \( Q_{i,j} \).

Let the average cell arrival rate at input \( i \) for output \( j \) be \( \lambda_{ij} \). We assume that incoming traffic is admissible; that is, \( \sum_{i=1}^{N} \lambda_{ij} < 1 \), and \( \sum_{j=1}^{N} \lambda_{ij} < 1 \). The arrival process is identical to both switches. The goal is to find a scheduling policy \( \Pi_{IQ} \) that tracks the behaviour of the OQ switch as close as possible, where we define what tracking means more precisely after introducing some definitions. Given that an IQ switch requires at least a speedup of two to exactly emulate an OQ switch [3], a scheduling policy for an IQ switch with a unity speedup can not exactly emulate the behaviour of an OQ switch, under all possible traffic patterns. In general, cells arriving to the IQ switch implementing \( \Pi_{IQ} \) will depart at some later time than the OQ switch implementing \( \Pi_{OQ} \). Consequently, we say that an IQ switch implementing \( \Pi_{IQ} \) lags the behaviour of the OQ switch implementing \( \Pi_{OQ} \).

2.1 Definition of Terms

Here we make precise some of the terminology used throughout this paper.

**Definition 1.** Arrival Rate Matrix (\( \lambda \)): \( \lambda \equiv [\lambda_{ij}] \), where the arrival process is assumed to be admissible and stationary; that is, \( \sum_{i=1}^{N} \lambda_{ij} < 1 \), \( \sum_{j=1}^{N} \lambda_{ij} < 1 \), \( \lambda_{ij} \geq 0 \) and associated arrival rate vector \( \lambda \equiv (\lambda_{1,1}, \ldots, \lambda_{1,N}, \ldots, \lambda_{N,1}, \ldots, \lambda_{N,N})^T \).

**Definition 2.** Ideal departure time (IDT): The ideal departure time for a cell \( c \), \( \text{IDT}(c) \), is the time slot at which \( c \) will depart from an OQ switch.
Definition 3. **Actual departure time (ADT):** The actual departure time (ADT) for a cell $c$, $ADT(c)$, is the time slot at which $c$ departs from the switch under consideration (i.e., IQ implementing $\Pi_{IQ}$).

Definition 4. **Cell Lag (CL):** The cell lag for a cell $c$, $CL(c)$, is the difference between the ideal departure time and the actual departure time. Precisely,

$$CL(c) \equiv \begin{cases} 
ADT(c) - IDT(c) & ADT(c) > IDT(c) \\
0 & otherwise
\end{cases}$$ (1)

In addition, we define the cell lag for a cell $c$ given the current time slot $n$, $CL(c, n)$, as the difference between the ideal departure time and the current time slot. Precisely,

$$CL(c, n) \equiv \begin{cases} 
n - IDT(c) & n > IDT(c) \\
0 & otherwise
\end{cases}$$

The goal of a scheduling policy can be characterized by any statistical metric that attempts to minimize the cell lag; for example, in Section 4.2 we present a scheduling policy that minimizes the mean lag value per packet.

Note that according to equation (1) the lag is nonnegative and generally a cell’s ADT is greater than its IDT, however, a cell may occasionally depart from an IQ switch earlier than an OQ switch; for example, consider a $2 \times 2$ switch at a specific time slot such that the two most lagging cells for its
outputs (e.g., outputs 1 and 2) reside at the same input port (e.g., input 1). Because the scheduling policy can transfer at most one cell from each input port (e.g., input 1), another cell with an IDT in the future can be selected from the other input port (e.g., input 2) to improve the throughput.

3 MOTIVATION AND RELATED WORK

In an OQ switch arriving packets are immediately available at the outgoing link. Consequently, the only shared resource in an OQ switch is the outgoing link for which packets contend for access (output contention). In an IQ switch packets are queued at the input port of the switch and they must first contend for access to the switch fabric (input contention), before contending for the outgoing link; that is, in an IQ switch, there are two shared resources: the switch fabric and the outgoing link.

All present IQ scheduling policies resolve input and output contention using heuristics such as using a round-robin scheme at both the input and output to solve the contention fairly [12], or using the packet’s age (i.e., time in the switch) to resolve contention [13]. All these schemes can be seen as an approximation to the ideal case of an OQ switch, where all of the outgoing links are independent and packets are served independently in each outgoing link; that is, by tracking the behaviour of an OQ switch and minimizing the lag, we automatically resolve input and output contention in a fair manner and eliminate any starvation problem of inputs that other scheduling policies
have to carefully handle.

We emphasize that significant research effort (e.g., [1], [14], [15]) has been done in developing scheduling policies for ideal servers that provide bounded latency, jitter, and end-to-end delay for traffic flows. Unfortunately, the Internet does not consist only of ideal servers, but rather of heterogeneous servers (i.e., non ideal IQ and CIOQ servers, and ideal OQ servers). By tracking the behaviour of an ideal server, we approximate its behaviour as close as possible and attempt to bound the performance difference between the ideal server and an IQ switch.

Tabatabae et al. [16] consider the related problem of packetizing arbitrary fluid policies in an \( N \times N \) crossbar switch using FIFO virtual output queues. They define trackable fluid policies such that for each pair of input and output ports, at each time step, the cumulative number of packets sent between these ports differs from the cumulative fluid scheduled between these ports by less than 1. They prove that a tracking policy always exists for the special case of a \( 2 \times 2 \) switch, provide an example for a \( 3 \times 3 \) switch where a nonanticipative tracking policy does not exist, and propose several heuristics for packetizing fluid policies on general \( N \times N \) switches. Rosenblum et al. [17] further extend the results in [16] by relaxing the tracking constraint such that the cumulative difference in the number of packets sent using the fluid and packetized policies can be more than one packet. Our work differs from [16] and [17] in that we track the precise packet departure sequence in an OQ switch rather than the aggregate rate provided by a fluid scheduling
policy in an IQ switch, which does not necessarily track an OQ switch; for two scheduling policies to provide the same service rate they need to serve only the same number of packets per link, rather than tracking the precise packet departure order, which can be different between the two scheduling policies. This issue is discussed in more detail in Section 4.2.

4 TRACKING SCHEDULING POLICIES

We consider the case of $\Pi_{OQ} = FIFO$. The architecture of our IQ switch is shown in Figure 1. We use virtual output queueing (VOQ) at each input port of the switch and a crossbar as the switching fabric.

4.1 Computing the Ideal Departure Time

For $\Pi_{OQ} = FIFO$, arriving cells at the IQ switch can be immediately assigned an IDT using a simple parallel prefix circuit [18] (i.e., a ranker circuit).
Let $N_j(n)$ be the number of cells in the OQ switch destined to output $j$ at time slot $n$. The IQ switch uses $N$ rankers such that each ranker calculates the number of cells present in the OQ switch being tracked. At the beginning of each time slot, $n$, the number of packets in the OQ switch is computed as follows:

$$N_j(n) \equiv \begin{cases} 
N_j(n-1) - 1 & N_j(n-1) > 0 \\
0 & N_j(n-1) = 0
\end{cases}$$

Note that the subtraction of one in the previous equation accounts for one (cell/time slot) departure in the OQ switch. For every new cell $c$ arriving at time slot $n$ destined to output $j$, ranker $j$ assigns a numeric rank (from $1 \ldots N$) in a linear order $^2$ to packets arriving for output port $j$. The IDT of each cell is equal to its numeric rank plus $N_j(n-1)$, and $N_j(n-1)$ is updated accordingly. The complexity of computing the IDT($c$) in hardware using a parallel prefix computation is $\Theta(\log N)$ depth and $\Theta(N)$ circuit size, expressed in terms of binary operators [18].

### 4.2 Maximum Weighted Lag Scheduling Policy

Maximum weighted lag (MWL) is based on the implementation of a maximum bipartite weight-matching algorithm (MWM)[19]. A maximum weight matching on a bipartite graph with weighted edges is defined as a set of edges

---

$^2$We investigated diverse ordering schemes (e.g., round-robin, linear, etc) for assigning IDT to simultaneous cell arrivals destined to the same output and found it to have an insignificant effect on the results.
between input and output nodes with the maximum total weight among all possible sets satisfying the constraint that any input node is matched to at most one output node. At every time slot \( n \), we associate a weight \( W_{i,j} \) to every \( Q_{i,j} \) such that \( W_{i,j} = CL(HOL_{i,j}, n) \); that is, \( W_{i,j} \) is the lag of an HOL packet in \( Q_{i,j} \). The maximum weighted lag scheduling policy finds a matching \( M \) that maximizes \( \sum_{(i,j) \in M} W_{i,j} \) and can be found by solving an equivalent network flow problem [19]. The sequential run time complexity of MWM is \( \Theta(N^3 \log N) \)[19].

Previous work on MWM considered only the weight to be either some function of the occupancy of the VOQs (i.e., number of packets in each VOQ) or the waiting time of the cell at the head of line of each VOQ (e.g., [13], [20], [21], [22], and [23]). Consequently, these algorithms do not necessarily track the behaviour of an OQ switch and cells’ departure times may deviate from the ideal case under non-uniform traffic. In addition, using the occupancy of the VOQs as the edge weight can lead to starvation of certain inputs [13].

Because MWL computes the matching with the maximum possible total weight during every time slot, it aims at minimizing the mean lag (\( \mu_{lag} \)). Although this algorithm is too complex to implement in practice, it serves as a reference model for which other approximation algorithms are developed.

The stability of maximum weighted matching scheduling policies is a well studied problem in the literature. McKeown et al. [13] proved the stability of longest queue first (LQF) and oldest cell first (OCF) maximum weight matching for all admissible independent identically distributed (i.i.d.) ar-
arrival processes using a Lyapunov function technique; Dai and Prabhakar [21] extended the results to prove the stability of a maximum weight matching algorithm under any admissible arrival processes using fluid model techniques.

Although the results for the fluid model technique established in [21] could easily be used to prove the stability of MWL, it can not be further extended to derive a bound on the expected lag value. Consequently, we use a Lyapunov function technique that allows us to derive a bound on the expected lag value as described in Appendix B.

**Theorem 1.** A FIFO tracking policy that uses the maximum weighted lag as the scheduling policy is stable (achieves 100\% throughput) for all admissible i.i.d. arrival processes.

Proof: The proof is given in Appendix A. The proof of Theorem 1 for the stability of MWL is an adaptation of the proof for stability of Oldest Cell First scheduling presented in [13]. The proof uses substantially the same techniques to first develop a discrete time Markov chain reflecting the lag of a cell. The proof then identifies a quadratic Lyapunov function which establishes the existence of a negative drift in the Markov chain for sufficiently large states. The existence of the negative drift implies the stability of the Markov chain, using a result of Kumar and Meyn [24]. The stability of the lag implies the stability of the queue occupancy. The main differences in the proofs are as follows. The definition of a cell’s weight is changed from the cell’s age to the cell’s lag, which is equal to the cell age minus a positive term.
reflecting the cell’s ideal departure time. Lemma’s 7, 8, 9 and 10 in [13] are modified to reflect the new cell weights.

Different weight functions lead to different bounds on the average queue size (cell delay) with varying performance; for example, in [20] it is shown that all maximum weight matching scheduling policies with weight equal to the queue size raised to some positive $\alpha$, $\|Q_{i,j}\|^{\alpha}$, are stable. However, it is shown through simulation that under a specific arrival pattern the average cell delay is smaller when $\alpha = 0.5$ than for all higher values of $\alpha$. A methodology for deriving bounds on the cell delay and queue size is described in [23]. In [13] it was shown that Longest Queue First could potentially lead to starvation. Longest Port First (LPF) was proposed in [25] and was shown by simulation to provide better performance than LQF and OCF, but it is possible to construct a traffic pattern that leads to starvation for LPF [26]. All the previous results are applicable to stability in a single node (switch), the problem of scheduling a network of input-queued switches is considered in [27] and it is shown that both the LQF and LPF scheduling policies can be unstable for a fixed traffic pattern in a simple network of eight input-queued switches.

Before we proceed, we need the following definitions in addition to the definitions used for the Proof of Theorem 1 in Appendix A:

**Definition 5.** $L_1$ Norm: Given a vector $Z \in \mathbb{R}^{N^2}$, the norm $\|Z\|_1$ is defined
as:

\[ \|Z\|_1 = \sum_{k=1}^{N^2} |z_k| . \]

**Definition 6. Input-Output Norm:** Given a vector \( Z \in \mathbb{R}^{N^2}, Z = \{z_k, k = Ni + j, i, j = 1, \ldots, N\} \), the norm \( \|Z\|_{IO} \) is defined as:

\[ \|Z\|_{IO} = \max_{j=1,\ldots,N} \left\{ \sum_{k=1}^{N} |z_{Nk+j}|, \sum_{l=1}^{N} |z_{Nj+l}| \right\} \]

\( \|Z\|_{IO} \) takes the maximum of the sum of quantities related to all the queues referring either to the same input or to the same output; for example, the traffic arrival vector is admissible if and only if \( \|\Lambda\|_{IO} < 1 \).

**Definition 7.** Let \( L(n) \) be the lag vector at time slot \( n \) such that

\[ L(n) \equiv (L_{1,1}(n), \ldots, L_{1,N}(n), \ldots, L_{N,1}(n), \ldots, L_{N,N}(n))^T, \]

where \( L_{i,j}(n) \) is the lag of \( C_{i,j}(n) \) (cell at HOL of \( Q_{i,j} \) at time slot \( n \)).

**Theorem 2.** A bound on the mean lag, \( E[\|L(n)\|_1] \), using a maximum weighted Lag scheduling policy under any admissible i.i.d. arrival process is given by:

\[ E[\|L(n)\|_1] \leq \frac{N^3 + 3N^2\|\Lambda\|_1}{2(1 - \|\Lambda\|_{IO})}. \]

Proof: The proof is given in Appendix B.
We emphasize that the bound in Theorem 2 is a much stronger property than bounding the average packet delay in an IQ switch over that in an OQ switch. Not only does Theorem 2 provide a bound on the additional mean delay for all packets departing an IQ switch using MWL over an OQ switch, it also applies to any individual packet departing the IQ switch. Specifically, Theorem 2 provides a bound on the difference between the precise packet departure sequence from an IQ using MWL over that provided by an OQ switch; for example, consider an IQ scheduling policy that periodically serves the same number of packets per output port as an OQ switch over a time interval larger than the corresponding time interval in an OQ switch. For all admissible traffic, this behaviour would imply a bounded per packet average delay compared to an OQ switch, but it does not imply the property of Theorem 2. This behaviour occurs because each packet’s departure order could be different from the IQ scheduling policy compared to the OQ scheduling policy; the key difference lies in the lag definition such that a packet departing ahead of its time would have a zero lag. Observe that if a negative lag was allowed then the mean lag value becomes the additional mean delay in an IQ switch over that in an OQ switch as packets departing ahead of their IDT (negative lag) would offset packets departing after their IDT (positive lag). Furthermore, bounding the mean delay in an IQ switch over that in an OQ switch requires only knowledge about the average service rate per output port in both switches rather than the precise packet departure sequence from each switch.
5 Iterative Lag Scheduling Policy

Iterative lag (iLag) is a simple heuristic based on maximal matching. A maximal matching algorithm is one that adds connections incrementally, without removing connections made earlier. iLag can be implemented using an arbiter at each input and output port using a request-grant-accept paradigm. Initially all input and output arbiters are unmatched, then in each iteration:

1. Request: Each unmatched input sends a request to every unmatched output for which it has a queued cell.

2. Grant: If an unmatched output receives any requests, it chooses the request with the most lagging cell and sends a grant to this input.

3. Accept: If an unmatched input receives any grants, it chooses the grant for its most lagging cell and sends an accept signal to this output.

The input and output arbiter are considered matched. The algorithm executes until either no more matches can be made or a fixed number of iterations are performed. The hardware implementation of iLag comprises the hardware to compute the IDTs in an OQ switch, the hardware to select the maximum lagging cells at each output arbiter to send the grant signal, and the hardware at each input arbiter to select the maximum lagging cell to send the accept signal.
6 SIMULATION RESULTS

The average cell delay and $E[\|L\|_1]$ of MWL and iLag are evaluated by simulation for a $16 \times 16$ switch and compared to LPF [25], LQF [13], islip [12] and PIM [28]. All simulations were performed with 99% confidence and 1% accuracy. iLag, islip, and PIM were executed with 4 iterations. Bernoulli and bursty traffic distributions are used for performance evaluation.

6.1 Bernoulli Traffic Distribution

For Bernoulli i.i.d. distribution, we use three traffic models: uniform, log diagonal, and diagonal arrival pattern.

1. Uniform: $\lambda_{i,j} = \frac{\rho}{N}$ $\forall i, j$, where $N = 16$ is the size of the switch.

2. LogDiagonal: $\lambda_{i,j} = 2\lambda_{i,|j+1|}$, and $\sum_i \lambda_{i,j} = \rho$; for example, the distribution of the load at input 1 across all outputs is $\lambda_{i,j} = \frac{2N-j}{2N} \rho$. This arrival pattern is more skewed than uniform loading.

3. Diagonal: $\lambda_{i,j} = 2\rho/3$, $\lambda_{i,|i+1|} = \rho/3$ $\forall i$, and $\lambda_{i,j} = 0$ for all other $i$ and $j$. This is very skewed loading and is more difficult to schedule than uniform loading.

As shown in Figure 2, MWL provides the lowest $E[\|L\|_1]$ compared to other maximum weight matching schemes under uniform Bernoulli arrivals, although all maximum weight matching schemes have almost the same aver-
Figure 2: $E[\|L\|_1]$ versus offered load for uniform Bernoulli i.i.d traffic

Figure 3: Average cell delay versus offered load for uniform Bernoulli i.i.d. traffic
age cell delay as shown in Figure 3. The same trend occurs for iLag compared to islip and PIM.

![Graph showing comparison of mean L1 norm across different scheduling policies]

Figure 4: $E[\|L\|_1]$ versus offered load for log diagonal traffic

Similarly, under log diagonal traffic, MWL provides the lowest $E[\|L\|_1]$ as shown in Figure 4, whereas the delay of all maximum weighted matching scheduling policies is almost identical as shown in Figure 5.

The same trend occurs for diagonal traffic as shown in Figures 6 and 7.

### 6.2 Bursty Traffic Distribution

Internet traffic is bursty in nature [29]. We considered an ON/OFF Markov Modulated Process with geometric burst size of 16. This traffic model is described in detail in [30].

The value of $E[\|L\|_1]$ is generally higher under bursty traffic than under a Bernoulli traffic distribution. As shown in Figure 8, MWL achieves the
Figure 5: Average cell delay versus offered load for log diagonal traffic

Figure 6: $E[\|L\|_1]$ versus offered load for diagonal traffic
Figure 7: Average cell delay versus offered load for diagonal traffic

Figure 8: $E[\|L\|_1]$ versus offered load for bursty traffic
lowest lag compared to other maximum weighted matching policies, whereas their delays is almost identical as shown in Figure 9. Similarly, iLag achieves the smallest lag compared to islip and PIM.

7 CONCLUSION

IQ switches are commercially used in most Internet routers due to their capability of operating at high line speeds with a lower memory bandwidth requirement than OQ switches. In this paper, we addressed the issue of fair scheduling in Internet routers with IQ switches. We formulated switch scheduling in an IQ switch with unity speedup as tracking the behaviour of an OQ switch. By tracking the behaviour of an OQ switch, an IQ switch resolves input and output contention fairly, eliminates any starvation of inputs, and approximates the behaviour of an OQ switch as close as possible.
We introduced the lag as a performance metric that measures the difference between a packet’s departure time in an IQ switch compared to an OQ switch. We proved that per packet lag is bounded for a maximum weighted matching scheduling policy that uses lag values for its weights and derived a bound on the mean lag value using a Lyapunov function technique. Finally, we proposed a simple heuristic tracking scheduling policy and evaluated its performance by simulation.

References


APPENDIX A: MWL STABILITY PROOF

Model

The arrival process at each input port $i$ is assumed to be a discrete-time stationary ergodic process of fixed size cells. At the beginning of each slot, either zero or one cell arrives at each input port. Virtual output queueing is used such that when a cell arrives at time slot $n$ for output $j$ at input $i$, it is placed in queue $Q_{i,j}$.

Definition 8. Let $Q(n)$ be the occupancy vector at time slot $n$ such that

$$Q(n) \equiv (Q_{1,1}(n), \ldots, Q_{1,N}(n), \ldots, Q_{N,1}(n), \ldots, Q_{N,N}(n))^T.$$

Definition 9. Let $\lambda_{\min} \equiv \min(\lambda_{i,j}, 1 \leq i, j \leq N)$.

Definition 10. Let $C_{i,j}(n)$ denote the HOL cell of $Q_{i,j}$ at time slot $n$.

Definition 11. Let $\tau(n)$ be the interarrival time vector such that

$$\tau(n) \equiv (\tau_{1,1}(n), \ldots, \tau_{1,N}(n), \ldots, \tau_{N,1}(n), \ldots, \tau_{N,N}(n))^T.$$

where $\tau_{i,j}(n)$ is the interarrival time between $C_{i,j}(n)$ and the cell behind it in $Q_{i,j}$ ([13], appendix B, definition 2).

Definition 12. Let $A(n)$ be the arrival matrix representing the arrivals into
each queue at time slot \( n \), \( A(n) \equiv [A_{i,j}(n)] \) where

\[
A_{i,j}(n) \equiv \begin{cases} 
1 & \text{if an arrival occurs at } Q_{i,j} \text{ at time slot } n \\
0 & \text{otherwise}
\end{cases}
\]

and the associated arrival vector is

\[
A(n) \equiv (A_{1,1}(n), \ldots, A_{1,N}(n), \ldots, A_{N,1}(n), \ldots, A_{N,N}(n))^T.
\]

([13], appendix A, definition 2)

**Definition 13.** Let \( S(n) \) be the service matrix indicating which queues are served during slot \( n \), \( S(n) \equiv [S_{i,j}(n)] \) where

\[
S_{i,j}(n) \equiv \begin{cases} 
1 & \text{if } Q_{i,j} \text{ is served at time slot } n \\
0 & \text{otherwise}
\end{cases}
\]

and \( S(n) \in S \), the set of service matrices. Note that \( S(n) \) is a permutation matrix; that is, \( \sum_{i=1}^{N} S_{ij} = \sum_{j=1}^{N} S_{ij} = 1 \). We define the associated service vector \( S(n) \equiv (S_{1,1}(n), \ldots, S_{1,N}(n), \ldots, S_{N,N}(n))^T \).

**Definition 14.** Let \( L(n) \) be the lag vector at time slot \( n \) such that

\[
L(n) \equiv (L_{1,1}(n), \ldots, L_{1,N}(n), \ldots, L_{N,1}(n), \ldots, L_{N,N}(n))^T,
\]

\[\text{3This definition of the “service” matrix is a permutation matrix, which includes the case where an empty queue is served.}\]
where $L_{i,j}(n)$ is the lag of $C_{i,j}(n)$ (cell at HOL of $Q_{i,j}$ at time slot $n$). (Recall that the lag is the difference between the ideal departure time and the current time, also note that all elements in the lag vector are nonnegative.)

Definition 15. Let $L_{\text{max}} \equiv \max(L_{i,j}, 1 \leq i, j \leq N)$.

Definition 16. Let $T$ be a positive-semidefinite diagonal matrix whose diagonal elements are $\lambda_{1,1}, \ldots, \lambda_{1,N}, \ldots, \lambda_{N,1}, \ldots, \lambda_{N,N}$.

Definition 17. $[a \odot b \odot c]$ denotes a vector in which each element is a product of the corresponding elements of the vectors: $a$, $b$, and $c$, i.e., $a_{i,j}b_{i,j}c_{i,j}$.

Definition 18. Let $\mathbf{1}$ denote a column vector of dimension $N^2$ whose elements are all ones.

Definition 19. Let $D(n, n + \Delta n)$ be the aggregate arrival vector for each output port during the time interval $[n, n + \Delta n]$

$$D(n, n + \Delta n) = (D_1(n, n + \Delta n), \ldots, D_N(n, n + \Delta n))^T,$$

where $D_j(n, n + \Delta n)$ represents the aggregate number of cells that arrived to the switch during the time interval $[n, n + \Delta n]$ destined to output $j$. Note that the dimension of the vector $D(n, n + \Delta n)$ is $N$, whereas most previously defined vectors have dimension $N^2$; consequently, we define the following
vector:

\[
Z(n, n + \Delta n) \equiv \left( D_1(n, n + \Delta n), \ldots, D_N(n, n + \Delta n), \ldots, \\
D_1(n, n + \Delta n), \ldots, D_N(n, n + \Delta n), \ldots, \\
D_1(n, n + \Delta n), \ldots, D_N(n, n + \Delta n) \right)^T
\]

i.e., the vector \(Z(n, n + \Delta n)\) is the vector \(D(n, n + \Delta n)\) written out \(N\) times.

**Definition 20.** The approximate Lag next-state vector, which does not consider the case of an empty queue is given by:

\[
\tilde{L}(n+1) \equiv L(n) + 1 - \left[ S(n) \odot [\tau(n) + Z(\tau(n))] \right]
\]

Explanation: The above equation describes the evolution of the lag vector. In the above equation, if \(Q_{i,j}\) is not serviced at slot \(n\) then its corresponding \(S_{i,j}\) element in \(S(n)\) is zero and the corresponding term in \(S(n) \odot [\tau(n) + Z(\tau(n))]\) cancels out. In this case the lag increases by 1. Alternatively, if the HOL cell at \(Q_{i,j}\) is serviced at time slot \(n\), then we need to calculate the lag of the cell following it in the queue. We consider two subcases:

CASE A: There were no packet arrivals to the switch destined to output \(j\) during the interarrival period between the HOL cell at \(Q_{i,j}\) and the cell following it (i.e., \(Z_{i,j}(\tau_{ij})\) is zero). In this case, the corresponding element for \(Q_{i,j}\) in \(S(n)\) is 1 and \(Z_{i,j}(\tau_{ij})\) is zero. Therefore,

\[
L_{i,j}(n + 1) = L_{i,j}(n) + 1 - \tau_{i,j}(n),
\]
i.e., the new lag is the old lag minus the interarrival time between the two cells.

CASE B: There were arrivals during the interarrival period between the HOL cell in $Q_{i,j}$ and the cell following it. In this case, all cells that arrived during this interarrival period should depart from the switch (or be selected to be transferred across the switch by the scheduler) before the new HOL cell at $Q_{i,j}$, so the new lag is given by:

$$L_{i,j}(n+1) \equiv L_{i,j}(n) + 1 - \tau_{i,j}(n) - Z_{i,j}(\tau_{i,j}(n)),$$

The following facts are used in the proof of the stability of the lag vector.

**Fact 1.** For all $i, j, n$ an interarrival time $\tau_{i,j}(n)$ is independent of the lag $L_{i,j}(n)$. This is true because we are assuming an i.i.d. traffic model.

**Fact 2.** $\tau_{i,j}(n) \geq 1$ because there is at most one arrival per time slot, so the arrival times of any two consecutive cells must be at least one slot apart.

**Fact 3.** For all $i, j, n$ ($\lambda_{i,j} = 0$) $\Rightarrow (\|Q_{i,j}\| = 0) \Rightarrow (L_{i,j}(n) = 0)$; that is, any queue whose arrival rate is zero is empty and consequently has a zero lag.

Proof of Theorem 1. We prove the stability of the lag vector, which implies the stability of the queue occupancy. Recall that the lag is defined in terms of the total occupancy of packets in the switch destined to an output port.

The following Lemma is adapted from [13], Lemma 7.
Lemma 1. $L^T(n)\lambda - L^T(n)S^*(n) \leq 0$, $\forall L(n), \lambda$ where $S^*(n)$ is such that $L^T(n)S^*(n) = \max(L^T(n)S(n))$ (Note that $S^*(n)$ is the service vector selected by the maximum weighted lag scheduling policy at time slot $n$.)

Proof. Identical to the proof of [13], Lemma 2.

The following Lemma is adapted from [13], Lemma 8 and is simplified for an $N \times N$ switch rather than an $N \times M$ switch.

Lemma 2. For all $\lambda \leq (1 - \beta)\lambda_m$ (the inequality is interpreted component-wise), $0 < \beta < 1$, where $\lambda_m$ is any rate vector such that $\|\lambda_m\|^2 = N$, there exists $0 < \varepsilon < 1$ such that

$$
E[\tilde{L}^T(n + 1)T\tilde{L}(n + 1) - L^T(n)TL(n)|L(n)] \leq \varepsilon\|L(n)\| + K.
$$

Proof. By expansion

$$
\begin{align*}
\tilde{L}^T(n + 1)T\tilde{L}(n + 1) &= L^T(n)TL(n) + 2L^T(n)\lambda \\
&\quad - 2L^T(n)[S^*(n) \odot \tau(n) \odot \lambda] - 2L^T(n)[S^*(n) \odot Z(\tau(n)) \odot \lambda] \\
&\quad + \sum_{i,j} \lambda_{i,j} - 2 \sum_{i,j} S^*_{i,j}(n)\tau_{i,j}(n)\lambda_{i,j} \\
&\quad - 2 \sum_{i,j} S^*_{i,j}(n)Z_{i,j}(\tau_{i,j}(n))\lambda_{i,j} + \sum_{i,j} S^2_{i,j}(n)\tau^2_{i,j}(n)\lambda_{i,j} \\
&\quad + 2 \sum_{i,j} S^*_{i,j}(n)Z_{i,j}(\tau_{i,j}(n)) + \sum_{i,j} S^2_{i,j}(n)Z^2_{i,j}(\tau_{i,j}(n))\lambda_{i,j}.
\end{align*}
$$

Subtracting $L^T(n)TL(n)$ from both sides and taking the expected value and
observing that the expected value of \( \tau \) is \( \frac{1}{\lambda} \),

\[
E[\tilde{L}^T(n + 1)T \tilde{L}(n + 1) - \tilde{L}^T(n)TL(n)|L(n)] = 2\tilde{L}^T(n)\tilde{\lambda} - 2\tilde{L}^T(n)S^*(n) - 2\tilde{L}^T(n)\left(S^*(n) \odot Z(\tau) \odot \tilde{\lambda}\right) + \sum_{i,j} \lambda_{i,j} - 2 \sum_{i,j} S^*_{i,j}(n) - 2 \sum_{i,j} S^*_{i,j}(n)E[Z_{i,j}(\tau_{i,j})\lambda_{i,j}] + \sum_{i,j} S^*_{i,j}(n)E[Z_{i,j}^2(\tau_{i,j})\lambda_{i,j}].\tag{2}
\]

We make use of the following properties to simplify equation (2) and establish Lemma 2:

(a) \( \sum_{i,j} \lambda_{i,j} < N \); (from the admissibility constraints)

(b) \( \sum_{i,j} S^*_{i,j}(n) \geq 0 \); (from the scheduling algorithm properties) so, this term can be ignored in equation (2) because it has a negative sign.

(c) \( \tilde{L}^T(n)\left(S^*(n) \odot Z(\tau) \odot \tilde{\lambda}\right) \geq 0 \); (because each element in this term is non-negative; observe that this term has a negative sign in equation (2) so it can be ignored)

(d) \( \sum_{i,j} S^*_{i,j}(n)E[Z_{i,j}(\tau_{i,j})\lambda_{i,j}] \geq 0 \); (because each element in this term is non-negative; observe that this term has a negative sign in equation (2) so it can be ignored) Also, note that the following positive terms in equation (2) are bounded:
\[
\sum_{i,j} S^*_{i,j}(n)/\lambda_{i,j} \leq \psi < \infty,
\]
\[
\sum_{i,j} S^*_{i,j}(n)E[Z^2_{i,j}(\tau_{i,j})\lambda_{i,j}] \leq \alpha < \infty
\]
\[
\sum_{i,j} S^*_{i,j}(n)E[Z_{i,j}(\tau_{i,j})] \leq \gamma < \infty
\]

From equation (2), properties (a) through (d), and equation (3), we obtain

\[
E[\tilde{L}^T(n+1)T\tilde{L}(n+1) - L^T(n)T\tilde{L}(n)\tilde{L}(n)]
\]
\[
\leq 2L^T(n)\Delta - 2L^T(n)S^*(n) + N + \psi + 2\alpha + \gamma.
\]

Using Lemma 2, we obtain:

\[
L^T(n)T\tilde{L}(n) \leq -\beta L^T(n)\Delta_m
\]
\[
L^T(n)\Delta - L^T(n)S^*(n) \leq -\beta\|L^T(n)\|\|\Delta_m\| \cos(\theta)
\]

where \(\theta\) is the angle between \(L^T(n)\) and \(\Delta_m\).

We now show that \(\cos(\theta) > \delta\) for some \(\delta > 0\) whenever \(L^T(n) \neq 0\) using the same approach as in [13], equations (16)-(18). This is included here for completeness and is simplified for an \(N \times N\) switch rather than an \(N \times M\) switch.

We do this by contradiction: suppose that \(\cos(\theta) = 0\), i.e., \(L^T(n)\) and \(\Delta_m\) are orthogonal. This can only occur if \(L^T(n) = 0\), or if for some \(i, j\), both \(\lambda_{i,j} = 0\) and \(L_{i,j}(n) > 0\), which is not possible: for \(Q_{i,j}\) to have a lag
greater than zero, \( \lambda_{i,j} \) must be greater than zero. Therefore, \( \cos(\theta) > 0 \) unless \( \mathbf{L}^T(n) = 0 \). Now we show that \( \cos(\theta) > \delta \) for some \( \delta > 0 \). Because \( \lambda_{i,j} > 0 \) wherever \( \mathbf{L}_{i,j}(n) > 0 \), and because \( \|\mathbf{\Delta}\|^2 < N \)

\[
\cos(\theta) = \frac{\mathbf{L}^T(n)\mathbf{\lambda}}{\|\mathbf{L}(n)\|\|\mathbf{\lambda}\|} \geq \frac{L_{\text{max}}(n)\lambda_{\text{min}}}{\|\mathbf{L}(n)\|\sqrt{N}}.
\]

Also, \( \|\mathbf{L}(n)\| \leq N\mathbf{L}_{\text{max}}(n) \), and so \( \cos(\theta) \) is bounded below by

\[
\cos(\theta) \geq \frac{\lambda_{\text{min}}}{N^{\frac{3}{2}}}.
\]

(6)

Substituting equation (6) in equation (5) we get

\[
\cos(\theta) = \frac{\mathbf{L}^T(n)\mathbf{\lambda}}{\|\mathbf{L}(n)\|\|\mathbf{\lambda}\|} \geq \frac{L_{\text{max}}(n)\lambda_{\text{min}}}{\|\mathbf{L}(n)\|\sqrt{N}}
\]

\[
E[\mathbf{\tilde{L}}^T(n + 1)\mathbf{\tilde{L}}(n + 1) - \mathbf{L}^T(n)\mathbf{L}(n)] | \mathbf{L}(n)] \leq -2\varepsilon\|\mathbf{L}(n)\| + K
\]

(7)

where \( \varepsilon = 2\beta\frac{\lambda_{\text{min}}}{N} \) and \( K = \psi + N + 2\alpha + \gamma \).

The following Lemma is adapted from [13] Lemma 9 and is simplified for an \( N \times N \) switch rather than an \( N \times M \) switch.

**Lemma 3.** For all \( \mathbf{\lambda} \leq (1 - \beta)\mathbf{\lambda}_m \) (the equation is interpreted component-wise), \( 0 < \beta < 1 \), where \( \mathbf{\lambda}_m \) is any rate vector such that \( \|\mathbf{\lambda}_m\| = N \), there exists \( 0 < \varepsilon < 1 \) such that

\[
E[\mathbf{L}^T(n + 1)\mathbf{L}(n + 1) - \mathbf{L}^T(n)\mathbf{L}(n)] | \mathbf{L}(n)] \leq \varepsilon\|\mathbf{L}(n)\| + K.
\]
Observe that the difference between Lemmas 2 and 3 is that Lemma 2 uses the approximate next state vector, whereas Lemma 3 uses the exact next state vector. The approximate next state vector assumes that each VOQ always has a packet. The exact next state vector takes the empty queue case into account.

The proof of this Lemma is similar to the proof of Lemma 9 in [1], and is included here for completeness.

Proof.

\[ L_{i,j}(n+1) = \begin{cases} \tilde{L}_{i,j}(n+1) & \tilde{L}_{i,j}(n+1) \geq 0 \\ 0 & \tilde{L}_{i,j}(n+1) < 0 \end{cases} \]  

(8)

The fact that \( T \) is a positive-semidefinite matrix together with equation (8) imply that for all \( n \)

\[ L^T(n+1)TLL(n+1) \leq \tilde{L}^T(n+1)T\tilde{L}(n+1). \]

Therefore,

\[ E \left[ L^T(n+1)TLL(n+1) - L^T(n)TLL(n) | L(n) \right] \leq E \left[ \tilde{L}^T(n+1)T\tilde{L}(n+1) | L(n) \right]. \]

This proves the Lemma. \( \square \)

**Lemma 4.** There exists a quadratic Lyapunov function \( V(L(n)) \) such that

\[ E \left[ V(L(n+1)) - V(L(n)) | L(n) \right] \leq -\varepsilon \| L(n) \| + K \]
where $K, \varepsilon > 0$.

Proof. From Lemma 3, $V(L(n)) = L^T(n)TLL(n)$, $\varepsilon = 2/3\lambda_{\text{min}} N$, and $K = \psi + N + 2\alpha + \gamma$. \hfill \square

**Theorem 3.** Under Maximum Weighted Lag, the expectation of the lag values are bounded for all $n$ under all admissible and independent arrival processes, i.e., $\forall n, E[\|L(n)\|] < \infty$.

Proof. $V(L(n)) = L^T(n)TLL(n)$ is a quadratic Lyapunov function and according to the arguments in [24], it follows that the expectation of the lag values is bounded for all $n$ under the maximum weighted lag scheduling policy. \hfill \square

**Theorem 4.** Under the MWL scheduling policy, the expectation of the queue occupancy is bounded for all $n$ under all admissible and independent arrival process, i.e., $\forall n, E[\|Q(n)\|] < \infty$.

Proof. That stability of the lag values implies the stability of the per packet additional waiting in the IQ switch using the MWL scheduling policy over that provided by the OQ switch being tracked. Given the traffic admissibility constraints, each packet’s delay in the OQ switch being tracked is finite. Consequently, the total delay provided by the IQ switch using MWL is bounded. Therefore, all the queue occupancies in the IQ switch under MWL are bounded for all $n$. \hfill \square
APPENDIX B: LAG BOUND FOR MWL SCHEDULING POLICY

In addition to the definitions given in Appendix A, the following definitions are necessary in this part.

**Definition 21.** Given a vector \( \mathbf{Z} \in \mathbb{R}^{N^2} \), the second order norm \( \| \mathbf{Z} \|_2 \) is defined as:

\[
\| \mathbf{Z} \|_2 = \sqrt{\sum_{k=1}^{N^2} (Z_k)^2}
\]

**Definition 22.** The unit vector parallel to \( \mathbf{Z} \) is denoted by \( \hat{\mathbf{Z}} \), and is defined as:

\[
\hat{\mathbf{Z}} = \frac{\mathbf{Z}}{\| \mathbf{Z} \|_1}
\]

To proceed we need the following theorem due to Leonardi et al. [23], Theorem 3.6, which is presented here in a form appropriate for the problem under consideration.

**Theorem 5** ([23], Theorem 3.6). Given a system of queues whose evolution is described by a Discrete Time Markov Chain (DTMC) with state vector \( Y_n \in \mathbb{N}^M \), whose state space \( H \) is a subset of the Cartesian product of a denumerable state space \( H_L \) and a finite state space \( H_K \), and for which all the polynomial moments of lag distributions are finite, if a lower bounded polynomial function \( V(L(n)), V : \mathbb{N}^N \rightarrow \mathbb{R} \), can be found, such that \( E[V(L(n)) \mid Y_n] < \infty \) and there exist two positive real numbers \( \epsilon \in \mathbb{R}^+ \) and
\( B \in \mathbb{R}^+, \text{ such that} \)

\[
E[V(L(n+1)) - V(L(n)) \mid Y_n] \leq -\epsilon f(\|L(n)\|) \quad \forall Y_n : \|L(n)\| > B, \quad (9)
\]

where \( f(x) \) is a continuous function in \( \mathbb{R}^+ \), then

\[
\lim_{n \to \infty} E\left[f\left(\|L(n)\|\right)\right] \leq \lim_{n \to \infty} E\left[f\left(\|L(n)\|\right)\right] + \frac{V(L(n+1)) - V(L(n))}{\epsilon} \mathbb{P}(Y_n \in H_B) \times P(Y_n \in H_B)
\]


(10)

Note that for MWL \( Y(n) = (A(n), L(n), \tau(n)) \) is an appropriate DTMC and all the polynomial moments of the lag distribution are finite by Theorem 3.5 of [23].

The proof of Theorem 2 consists of two steps. First, we find a lower bound on \( \epsilon \) in equation (9). The second step is to use equation (10) to derive the bound on \( E[\|L(n)\|_1] \).

Using equation (10) with \( f(\|L(n)\|) = \|L(n)\|_1 \) and \( V(L(n)) = L^T(n)TL(n) \)

\[
- \frac{E[L^T(n+1)TL(n+1) - L^T(n)TL(n)L(n)]}{\|L(n)\|_1} \geq \epsilon \quad \forall L(n) : \|L(n)\|_1 > B
\]

(11)
for some $B > 0$. The function at the left hand side of equation (11) admits
a limit for $\|L(n)\|_1 \to \infty$ which depends on the direction of the vector $L(n)$.
Let $\epsilon_{\text{max}}$ be the smallest value for this limit, i.e.

$$
\epsilon_{\text{max}} = \lim_{\|L(n)\|_1 \to \infty} \frac{E[L^T(n+1)TL(n+1) - L^T(n)TL(n)|L(n)]}{\|L(n)\|_1}.
$$

Substituting equation (2) in the above equation and observing that all
the terms in the numerator that do not contain $L(n)$ will go to zero upon
dividing by $\|L(n)\|_1 \to \infty$, we get

$$
\epsilon_{\text{max}} = \lim_{\|L(n)\|_1 \to \infty} \frac{2L^T(n)\lambda - 2L^T(n)S^*(n) - 2L^T(n)(S^*(n) \odot Z(\tau) \odot \lambda)}{\|L(n)\|_1}.
$$

Rearranging the terms we get:

$$
\epsilon_{\text{max}} = \lim_{\|L(n)\|_1 \to \infty} \frac{2L^T(n)S^*(n) + 2L^T(n)(S^*(n) \odot Z(\tau) \odot \lambda) - 2L^T(n)\lambda}{\|L(n)\|_1}.
$$

Taking 2 as a common factor and rearranging the terms we get:

$$
\epsilon_{\text{max}} = 2 \lim_{\|L(n)\|_1 \to \infty} \left( \frac{L^T(n)S^*(n) - L^T(n)\lambda}{\|L(n)\|_1} + \frac{L^T(n)(S^*(n) \odot Z(\tau) \odot \lambda)}{\|L(n)\|_1} \right).
$$

(12)

We make use of the following proposition, which was proved in [23]
(Proposition A.1) and is included here for completeness.
**Proposition 1.** For any nonnull normalized vector \( \hat{Z}(n) \in \mathbb{R}^{+N^2} \):

\[
S^*(n)\hat{Z}^T(n) \geq \frac{1}{N}
\]

Applying Proposition 1 to the second term in equation (12) we get:

\[
\frac{L^T(n)(S^*(n) \odot Z(\tau) \odot \Lambda)}{\|L(n)\|_1} \geq \frac{Z(\tau) \cdot \Lambda}{N} \geq 0.
\]

Now, we use a technique from [23](pg. 542 and 543) to bound the following term:

\[
\frac{L^T(n)S^*(n) - L^T(n)\Lambda}{\|L(n)\|_1}.
\]

Consider the vector \( \overline{U}(n) = E[A(n)] + (1 - \|\Lambda\|_{IO})S^*(n) \). It is straightforward to prove that \( \|\overline{U}(n)\|_{IO} \leq 1 \). Also, the fact that the system is stable implies \( E[A(n)] = E[S^*(n)] = \overline{\Lambda} \). Thus,

\[
\frac{S^*(n)L^T(n) - \overline{U}(n)L^T(n)}{\|L(n)\|_1} = \frac{S^*(n)L^T(n) - \overline{\Lambda} \odot L^T(n) - (1 - \|\overline{\Lambda}\|_{IO})S^*(n)L^T(n)}{\|L(n)\|_1} \geq 0
\]

and from Lemma 1 we have

\[
\frac{S^*(n)L^T(n) - \overline{\Lambda}L^T(n)}{\|L(n)\|_1} \geq (1 - \|\overline{\Lambda}\|_{IO})S^*(n)L^T(n).
\]
Applying Proposition 1 we get:

\[
\frac{S^*(n)L^T(n) - \Lambda L^T(n)}{\|L(n)\|_1} \geq \frac{(1 - \|\Lambda\|_{IO})}{N}.
\]  

(13)

Substituting equations (7) and (13) in equation (10), we get:

\[
\epsilon_{\text{max}} \geq \frac{2}{N}(1 - \|\Lambda\|_{IO}).
\]  

(14)

The next step is to evaluate equation (10):

\[
\lim_{n \to \infty} E \left[ f\left(\|L(n)\|\right) \right] \leq \lim_{n \to \infty} E \left[ f\left(\|L(n)\|\right) \right] + \frac{V(L(n+1)) - V(L(n))}{\epsilon} |Y_n \in H_B| \times P[Y_n \in H_B]
\]

Evaluating the term \(E[V(L(n+1)) - V(L(n))|Y_n \in H_B]\) appearing in equation (9) and using the result from equation (2)

\[
E[V(L(n+1)) - V(L(n))|Y_n \in H_B]
\]

\[
= E[L^T(n)\Lambda - L^T(n)S^*(n) - L^T(n)(S^*(n) \odot Z(\tau) \odot \Lambda)]
\]

\[
+ \sum_{i,j} \lambda_{i,j} - 2E\left[\sum_{i,j} S^*_{i,j}(n)\right] - 2E\left[\sum_{i,j} S^*_{i,j}(n)Z_{i,j}(\tau_{i,j})\lambda_{i,j}\right]
\]

\[
+ E\left[\sum_{i,j} S^*_{i,j}(n)\frac{\lambda_{i,j}}{\lambda_{i,j}}\right] + 2E\left[\sum_{i,j} S^*_{i,j}(n)Z_{i,j}(\tau_{i,j})\right]
\]

\[
+ E\left[\sum_{i,j} S^*_{i,j}(n)Z^2_{i,j}(\tau_{i,j})\lambda_{i,j}\right].
\]
and using the result of equation (7) and equation (13) we get

\[
E[L^T(n + 1)T L(n + 1) - L^T(n)T L(n)|L(n)] \leq \\
\epsilon_{\text{max}}\|L(n)\|_1 + \sum_{i,j} \lambda_{i,j} - 2E\left[\sum_{i,j} S_{i,j}^*(n)\right] - 2E\left[\sum_{i,j} S_{i,j}^*(n)Z_{i,j}(\tau_{i,j})\lambda_{i,j}\right] \\
+ E\left[\sum_{i,j} \frac{S_{i,j}^*(n)}{\lambda_{i,j}}\right] + 2E\left[\sum_{i,j} S_{i,j}^*(n)Z_{i,j}(\tau_{i,j})\right] \\
+ E\left[\sum_{i,j} S_{i,j}^*(n)Z_{i,j}^2(\tau_{i,j})\lambda_{i,j}\right].
\]

From stability we have \(E[S(n)] = E[\Delta], \ E[S^T(n)S(n)] = \|\Delta\|_1, \) and \(E[\Delta^T S(n)] = E[\Delta^T]E[S(n)] = \|\Delta\|_2^2.\) Also, \(E[S_{i,j}] = \lambda_{i,j};\) so, \(E\left[\sum_{i,j} \frac{S_{i,j}^*(n)}{\lambda_{i,j}}\right] = N^2\) because we are summing over \(N^2\) elements and each element is 1. Similarly, \(E\left[\sum_{i,j} S_{i,j}^*(n)Z_{i,j}(\tau_{i,j})\right] \leq N\|\Delta\|_1
\]

\[
E[L^T(n + 1)T L(n + 1) - L^T(n)T L(n)|L(n)] \leq \\
\epsilon_{\text{max}}\|L(n)\|_1 - \|\Delta\|_1 - 2E\left[\sum_{i,j} S_{i,j}^*(n)Z_{i,j}(\tau_{i,j})\lambda_{i,j}\right] \\
+ N^2 + 2N\|\Delta\|_1 + E\left[\sum_{i,j} S_{i,j}^*(n)Z_{i,j}^2(\tau_{i,j})\lambda_{i,j}\right].
\]

(15)
Substituting equation (15) in equation (10) we get:

\[
E \left[ \| L(n) \|_1 \right] \leq E \left[ \| L(n) \|_1 + \frac{V(L(n+1)) - V(L(n))}{\epsilon} \right] \\
E \left[ \| L(n) \|_1 \right] \leq E \left[ \| L(n) \|_1 \left( 1 - \frac{\epsilon_{\text{max}}}{\epsilon} \right) \right] \\
+ \frac{N^2 + 2N \| \Delta \|_1 + E \left[ \sum_{i,j} Z_{i,j}^2 (\tau_{i,j}) \lambda_{i,j}^2 \right] - \| \Delta \|_1 - 2E \left[ \sum_{i,j} Z_{i,j} (\tau_{i,j}) \lambda_{i,j}^2 \right]}{\epsilon}.
\]

If we set \( \epsilon = \epsilon_{\text{max}} \) we get:

\[
E \left[ \| L(n) \|_1 \right] \leq \frac{N^2 + 2N \| \Delta \|_1 + E \left[ \sum_{i,j} Z_{i,j}^2 (\tau_{i,j}) \lambda_{i,j}^2 \right] - \| \Delta \|_1 - 2E \left[ \sum_{i,j} Z_{i,j} (\tau_{i,j}) \lambda_{i,j}^2 \right]}{\epsilon_{\text{max}}} \\
\leq \frac{N^2 + 2N \| \Delta \|_1 + E \left[ \sum_{i,j} Z_{i,j}^2 (\tau_{i,j}) \lambda_{i,j}^2 \right] - \| \Delta \|_1}{\epsilon_{\text{max}}} \\
\leq \frac{N^2 + 2N \| \Delta \|_1 + N \| \Delta \|_1}{\frac{1}{\epsilon_{\text{max}}} (1 - \| \Delta \|_{\text{IO}})} \\
\leq \frac{N^2 + 3N \| \Delta \|_1}{\frac{1}{\epsilon_{\text{max}}} (1 - \| \Delta \|_{\text{IO}})} \\
\leq \frac{N^3 + 3N^2 \| \Delta \|_1}{2(1 - \| \Delta \|_{\text{IO}})}.
\]