Adaptive Linear Prediction of Queues for Reduced Rate Scheduling in Optical Routers

Yang Jiao and Ritesh Madan
EE 384Y Final Project
Stanford University

***Abstract*—This paper describes a switching scheme that can be used to reduce delays in optical routers. Optical switches have a larger bandwidth and can potentially use less power than electronic switches. However optical switches take a significant amount of time (of the order of μsec) to change from one configuration to another. This introduces a time lag between the scheduling decision and actual change in configuration, leading to higher delays. We propose a scheduling scheme that decides the schedule for the future using maximum weight matching (MWM) on the predicted lengths of the virtual output queues (VOQs). The prediction is done using the adaptive Least Mean Square (LMS) algorithm. We show through simulations for correlated arrivals that this scheme reduces delay significantly in comparison to a scheme that makes a scheduling decision for the future using MWM on the current lengths of virtual output queues.**

I. Introduction

There has been a rapid increase in the bandwidth of high-speed access technologies in recent years. Dense Wavelength Division Multiplexing (DWDM) has increased the fiber transmission capacity at an unprecedented rate, making routers the main bandwidth bottleneck. Future routers need to accommodate hundreds of ports with line rates exceeding 80Gb/s. This has led to a belief that routers in the future will use optical components. If packet scheduling is used at such a high rate, with a packet size of 40B, the crossbar needs to switch to a new configuration every 4ns or less. (From this point onwards we will refer to this time corresponding to packet switch as a *single time slot*). This presents a great challenge in crossbar design since an optical switch takes a few microseconds to change from configuration to another. Hence in order to use an optical switch efficiently, we would like to switch at a rate less than once every time slot. A scheme that groups packets together in envelopes and switches the envelopes has been proposed by K. Kar et. al. [4]. However this results in unbounded delays at low loads. The delay can be bounded using flow management and multiple levels of scheduling, but this is at the cost of extra complexity.

Devavrat et. al. [1] have shown that a simple scheme that schedules packets in bursts gives 100% throughput and bounded delays. However the bound on the delay grows as a function of the burst length. This motivates the question whether we can reduce the delays by using past information of the VOQs to predict the state in the future. This paper explores the application of an adaptive linear filter to the prediction of VOQ lengths. Even though most flows are short lived, most of the packets belong to a small number of large flows (mice-elephant phenomenon). Hence we can expect the weights of the adaptive linear predictor to converge to optimal values for the large flows, with noise about these values caused due to the small flows. In the past linear predictors have been used to model VBR traces (eg. [6]). However these papers find the regression coefficients using least mean square error methods, with the assumption that the second order statistics of the trace are known before-hand. Hence this will give the best possible performance; however in real life it is difficult to estimate the second order statistics of incoming traffic at a router. Also the statistics change with time leading to nonstationarity. This is the reason why we propose an adaptive scheme as opposed to a scheme with fixed linear coefficients.

The rest of the paper is organized as follows: in Section II we consider different possible switch architectures. In section III we study the average delay for these architectures. Section IV describes the prediction scheme and discusses its convergence and stability. Section V simulation results and Section VI gives the conclusions and further investigation that is required.

II. Switch Architecture

We assume an input queued switching architecture using virtual output queues (VOQ). Time is slotted, and packet size is fixed. We assume that the switch has N inputs and N outputs. We will represent the VOQ sizes at time n as a vector X_n of length N^2, where $X_n(iN + j)$ is the VOQ at input i for output j. In each time slot, at most one packet arrives at each input. During each time slot the switching element can transfer at most one packet from each input, and at most one packet to each output. We will denote arrivals in time slot n as a N^2 dimensional vector A_n. The departure matrix chosen by the reduced rate scheduler for time slot n will be denoted as D_n. The VOQ size evolution can then be written as: $X_{n+1} = \max(X_n + A_{n+1} - D_n, 0)$.

We will assume that a single switching element takes m time slots to change from one configuration to another, during which it remains in an invalid state. We will refer to m as the *decision time*. Let's consider two architectures using such optical switches. In architecture AI, we consider time multiplexing between $(m + 1)$ switching elements. At every time slot, exactly one switching element is used to transfer packets. Each switching element transfers packets for one slot and then remains in a transient state for next m time slots before it is used to switch packets again. Thus switching occurs every time slot. From a scheduling algorithm's point of view, the difference between such an architecture and a packet switch is that the switch schedule needs to be decided m time slots before it is implemented. This is similar to the pipeline MWM considered in [1]. The second switch architecture, AII, uses two switching elements. When one switching element is in transient, the other is used to transfer packets for m time slots in one fixed configuration. We will refer to this time interval as the *burst time*. The two switching elements change roles every m time slots. This is a more general case of burst scheduling in [1]. From the schedulers point of view, the switch schedule needs to be decided m time slots ahead, and only one scheduling decision can be made every m time slots. It should be noted that AI and AII represent two extreme points in a generalized switching architecture using slow switching elements. A generalized architecture would use $(m + k)/k$ switching elements with each elements being scheduled for k time slots at a stretch. The next section discusses the effect of varying k, and

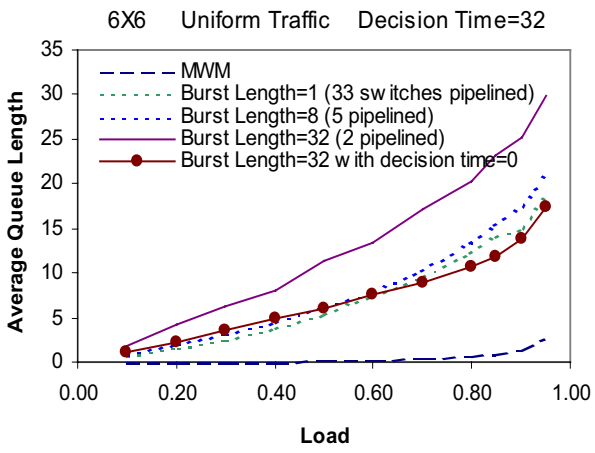

Fig. 1. Delay in reduced rate switches

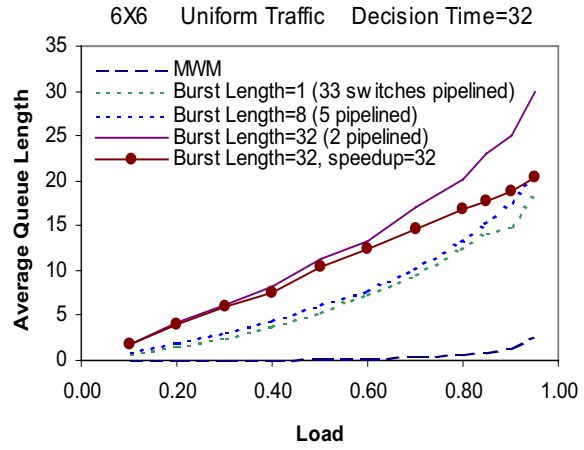

Fig. 2. Effect of speedup in reduced rate switches

the study will provide insight into ways to reduce the delay caused by reduced rate switching.

III. Delay Characterization

Using a larger number of switching elements, each with a shorter burst length k, will decrease the delay in the switch at the cost of complexity and power. We would like to know what value of the burst length we should use. We simulated a 6X6 switch with uniform arrivals. Figure 1 shows the average queue length for several values of the burst length k. Simulations for other traffic patterns show similar behavior. By decreasing k, the average queue length decreases, but asymptotically approaches a lower bound. We hypothesize that the delay can be broken into two dominating components, the burst time delay, and the decision time delay. We derived an analytical bound on the average delay in the Appendix using the method in [2]. However since it is only a bound it does not give an actual variation of the delay with the burst and decision times. We provide some intuition using simulations.

In architecture AII (e.g. when $k = 1$), the increase in the average delay is purely due to the decision time m. At time slot t, a set of N VOQ's are scheduled to be served at $t + m$. It should be noted that the delay in this architecture is not a true pipeline delay, since the set of VOQ's sitting in the buffer to be served at $t + m$ may get service between time slots t to $t + m - 1$. Nevertheless, there will be a large number of times when set of VOQ's are not served between time slots t to $t + m - 1$, then the set will be effectively sitting in a pipeline buffer of length m. Therefore, we expect a extra delay on the order of m time slots for burst length $k = 1$.

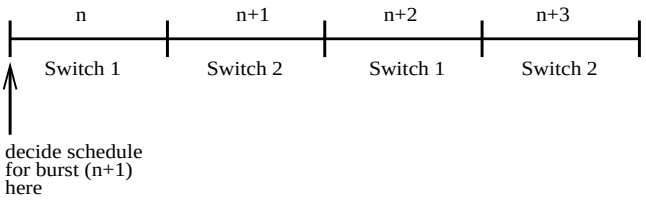

Fig. 3. Two Switch Architecture

In architecture AI (e.g. when $k = m$), the decision time pseudo-pipeline delay is still present. In addition, decision is only made once every k time slots. Arrivals between decisions will not effect the scheduling decision for $\frac{k}{2}$ time slots on average. Therefore, we expect a extra delay on the order of $\frac{k}{2}$ time slots. This argument is supported by the the simulation curve for $m = 0, k = 32$, also shown on Figure 1.

Figure 2 provides some more justification for our intuitive reasoning. We let $m = 32$ and $k = 32$, but we let the switch transfer packets 32 times faster than before. Most of the times, the high speedup will just let the switching element empty the N VOQ's it is serving in one time slot. But, Figure 2 shows that this does not significantly reduce the delay. This confirms our intuitive reasoning, since speedup does not change the fact that it takes m time slots for the right set of VOQ's to be served, and the fact that arrivals do not affect scheduling decisions for $\frac{k}{2}$ time slots.

By increasing the number of switching elements, the burst length can be decreased, and the delay improves. But the system complexity increases. Since our intuition suggests that the burst time delay and decision time delay are comparable, we will avoid using a large number of switching elements. Instead we aim to reduce the decision time delay rather than the burst time delay. Indeed, if we can predict what the VOQ sizes will be at time $t + m$, then the delay due to decision time will be eliminated. The next section discusses how this could be achieved using prediction of VOQs.

IV. Prediction Scheme

In this section we describe a scheme for adaptive prediction of VOQ lengths for Architecture II (which uses 2 switches operating in parallel). As shown in Figure 3, the schedule for burst (n+1) needs to be decided at the beginning of burst n. One possible way to schedule for burst (n+1) is to do MWM on VOQs at the beginning of burst n. However as we saw in the previous section if we could estimate the VOQs at the beginning of burst (n+1), we can reduce the delay significantly (MWM curve in Figure 2). The only missing information that we need to do this is the number of arrivals during burst n. If we can predict the number of arrivals during burst n, we can estimate the VOQs at the beginning of burst (n+1) as follows:

$$X_{pred}(n+1) = \{X(n) + A_{pred}(n+1) - D(n)\}^{+} \quad (1)$$

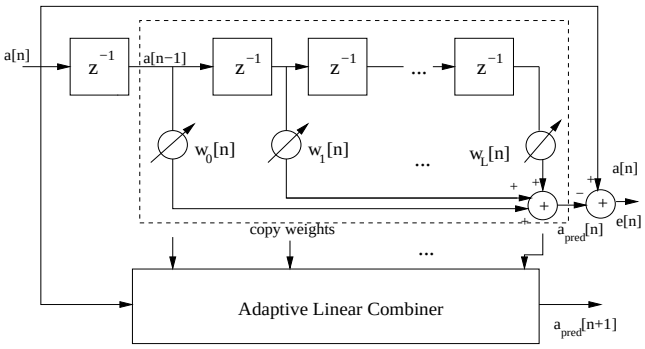

Fig. 4. Prediction Method

We study the application of an adaptive linear combiner to predict the arrivals during burst (n+1) using the history of arrivals till burst (n). Consider a linear combiner with l coefficients. The prediction is done using the following equation:

$$\hat{a}[n+1] = \sum_{i=0}^{l} w_i[n]a[n-i] \tag{2}$$

where $w_0[n], w_1[n], \ldots, w_l[n]$ represent the filter coefficients at time n. $a[n]$ represents the number of arrivals to a given VOQ during burst n (we drop the subscript i, j for brevity), and $\hat{a}[n+1]$ is the predicted number of arrivals during burst (n+1). The reason we use time varying weights is that we cannot know the statistics of the arrival process before-hand; and also the statistics are likely to change over time.

The adaptation of weights is shown in Figure 4. Let $W[n] = [w_0[n], w_1[n], \ldots, w_l[n]]$ be the weight vector during burst n and $A[n] = [a[n-1], a[n-1], \ldots, a[n-l]]$ be the input vector to the top filter in Figure 4 at the beginning of burst $(n+1)$. For adaptation we predict $a[n]$ using arrivals till burst $(n-1)$, i.e.

$$\hat{a}[n] = W^T[n]A[n]$$

Since we know $a[n]$, the error signal is given by

$$e[n] = a[n] - \hat{a}[n]$$

Using this error signal we adapt the weights according to the LMS algorithm [3]

$$W[n+1] = W[n] + 2\mu e[n]A[n] \tag{3}$$

A. Rate of Adaptation and Convergence

The time constant of adaptation of the LMS algorithm is inversely proportional to μ (Chapter 6 of [3]). However the algorithm does not converge for an arbitrarily large value of μ. Convergence is guaranteed if [3]

$$\mu < \frac{1}{(l+1)E\{a^2[n]\}} \tag{4}$$

An upper bound for $E\{a^2[n]\}$ can be obtained by using the actual maximum packet rate for the particular VOQ.

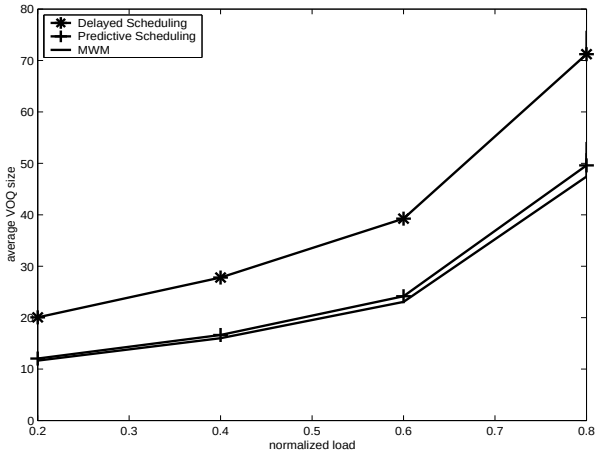

Fig. 5. Average Queue length as a function of load for an AR(1) process

B. 100% Throughput for Bernoulli IID traffic

For a stationary arrival process the LMS algorithm converges to the optimal Weiner solution if μ satisfies condition(4). Let W^P be the weight obtained by the prediction scheme and W^* be the weight obtained by MWM. Since the weight vector converges, we can write $W^* - W^P < C$ for a constant C. Then as shown in [1] the algorithm will give 100% throughput for Bernoulli IID traffic.

C. Complexity of Prediction

For each of the N^2 VOQs we need to do $(l+2)$ multiplications and $(l+1)$ additions. However since we can do these operations in parallel for each weight, we can do the computation in time needed for 2 multiplications and 1 addition, assuming fully parallelized hardware.

V. Simulation Results

Simulations were done for a 6X6 switch, for a uniform load. The simulations were done to study the difference in the performance of three schemes

1) **MWM:** Schedule for burst n by MWM on VOQs at beginning of burst n.
2) **Delayed Scheduling:** Schedule for burst n by MWM on VOQs at beginning of burst $(n-1)$.
3) **Predictive Scheduling:** Scheduling for burst n by MWM on VOQs predicted at beginning of burst $(n-1)$.

Note that the MWM scheme above is a hypothetical scheme for an optical switch as the scheduling decision needs to be made in advance. We use this a benchmark for our prediction scheme.

We give the simulations plots for two different correlated arrival processes. A burst lenghts of 64 time slots was used. We did not get much improvement for Bernoulli IID traffic (as expected because prediction will not do well if there is no correlation in the arrivals).

Figure 5 shows the performance of the three schemes for an arrival process $a(n)$ generated using an autoregressive process of order 1 (AR(1)). The predictive scheme does much better than delayed scheduling, and almost as well as MWM. Figure 6 shows a similar plot for an AR(3) process. Notice the improvement with respect to the delayed scheduling scheme

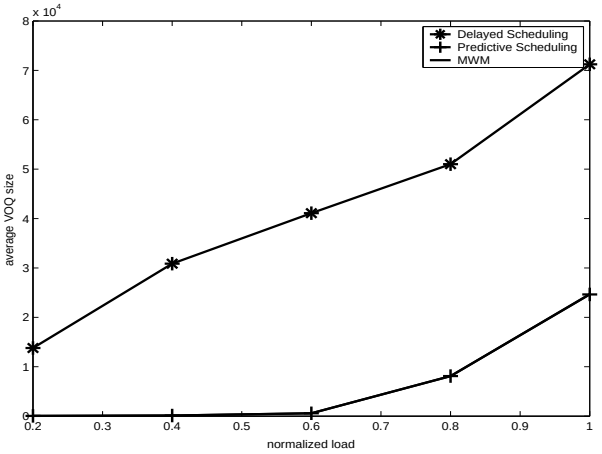

Fig. 6. Average Queue length as a function of load for an AR(3) process

is much more in the case of an AR(3) process (the plots for MWM and predictive scheduling are indistinguishable at this scale). This is because as the correlation between the arrivals increases, the prediction becomes more accurate. Also note that the queue sizes are higher in an AR(3) process than an AR(1) process. This is because if there are a large number of arrivals in a burst, it will be followed by a greater number of more such consecutive bursts with a large number of arrivals in the case of an AR(3) process than in an AR(2) process.

VI. Conclusions

We motivated the choice of Architecture II (which uses two switches in parallel) through a simulation study of delay in different architectures. We then studied a scheduling scheme that uses prediction to drastically reduce the delay caused by the decision time in optical switches. We showed that if the arrival process is stationary and correlated, prediction gives a performance comparable to MWM.

However we did not study the performance for actual internet traces due to the lack of availability of traces that give the arrivals to different VOQs at a switch. More study with such traces would be helpful, because the traffic at a router is expected to be non-stationary. For a non stationary arrival process the optimal weight solution will vary with time, and hence the adaptation needs to be quick enough to capture this variation. At the same time the adaptation rate is limited by convergence criteria.

Appendix

Intuitively, reduced rate scheduling will increase the queueing delay compared to MWM scheduling. To quantify the delay of reduced rate scheduling, we derive analytical bounds on the average delay under i.i.d. arrivals. The derivation uses drift analysis method first applied to IQ switches by Leonardi et. al. [2] Let $V(X_n)$ be a polynomial function of X_n. It was shown that for large values of X_n, if $V(X_n)$ has an average downward drift as a function of n, then the average queue size is bounded. We formalize this fact in the following theorem.

Theorem 1: Let $V(X_n)$ be a polynomial function of X_n. If

$$\mathrm{E}[V(X_{n+1})|X_n] < \infty \ \forall X_n \tag{5}$$

and $\exists \epsilon \geq 0$ and $B \geq 0$ such that:

$$\begin{aligned}\mathrm{E}[V(X_n+1) - V(X_n)|X_n] < -\epsilon f(\|X_n\|) \\ \forall X_n : \|X_n\| > B\end{aligned} \tag{6}$$

where $f(x)$ is a continuous function, then

$$\begin{aligned}&\lim_{n\to\infty} \mathrm{E}[f(\|X_n\|)] \\ &\leq \lim_{n\to\infty} \mathrm{E}[f(\|X_n\|) + \\ &\frac{V(X_n+1) - V(X_n)}{\epsilon} | \|X_n\| < B] P(\|X_n\| < B)\end{aligned} \tag{7}$$

Proof: This is a special case of the result given in [2]. Readers are refered to that paper for the proof.

McKeown et. al. [5] have shown that, for a MWM scheduler,

$$D_n^{MWM} X_n^T \geq \lambda X_n^T, \tag{8}$$

where D_n^{MWM} is the departure matrix given by a MWM scheduler, λ is any admissible arrival matrix ($\|\lambda\|_\alpha \leq 1$). Since the difference in the weight of the reduced rate scheduler and the MWM scheduler is C, we have

$$D_n X_n^T \geq \lambda X_n^T - C \tag{9}$$

We now derive a bound on the average queue size. Let $f(\|X_n\|) = \|X_n\|_1$, $V(X_n) = X_n X_n^T$ We will first find the value of ϵ in (5)

$$\begin{aligned}&\mathrm{E}\left[X_{n+1}X_{n+1}^T - X_n X_n^T | X_n\right] \\ =&\mathrm{E}\left[(X_n + A_{n+1} - D_n)(X_n + A_{n+1} - D_n)^T - X_n X_n^T | X_n\right] \\ =&\mathrm{E}\left[2(A_{n+1} - D_n)X_n^T + (A_{n+1} - D_n)(A_{n+1} - D_n)^T | X_n\right] \\ =&2\left(\mathrm{E}[A_{n+1}] - D_n\right) X_n^T + \mathrm{E}\left[(A_{n+1} - D_n)(A_{n+1} - D_n)^T | X_n\right] \\ \leq&2\left(\mathrm{E}[A_{n+1}] - D_n\right) X_n^T + N\end{aligned} \tag{10}$$

where we have used the fact that there are at most N arrivals per time slot. We make use of (9) to bound the first term in the above expression. Let $\hat{\lambda} = \mathrm{E}[A_{n+1}] + (1 - \|\mathrm{E}[A_{n+1}]\|_\alpha) D_n$. Thus $\hat{\lambda}$ is an admissible arrival matrix and hence,

$$\|\hat{\lambda}\|_\alpha = \|\mathrm{E}[A_{n+1}]\|_\alpha + (1 - \|\mathrm{E}[A_{n+1}]\|_\alpha) = 1 \tag{11}$$

so we have:

$$\begin{aligned}&2\left(\mathrm{E}[A_{n+1}]X_n^T - D_n X_n^T\right) + N \\ \leq&2\left(\mathrm{E}[A_{n+1}]X_n^T - (\hat{\lambda} X_n^T - C)\right) + N \\ \leq&-2\left((1 - \|\mathrm{E}[A_{n+1}]\|_\alpha) D_n X_n^T - C\right) + N\end{aligned} \tag{12}$$

Now let $\hat{\lambda} = \frac{1}{N} e$, where e is a vector of ones of length N^2. Clearly, $\hat{\lambda}$ is an admissible arrival matrix. Applying (9) again:

$$\begin{aligned}&-2\left((1 - \|\mathrm{E}[A_{n+1}]\|_\alpha) D_n X_n^T - C\right) + N \\ \leq&-2\left((1 - \|\mathrm{E}[A_{n+1}]\|_\alpha)(\frac{1}{N}\|X_n\|_1 - C) - C\right) + N \\ =&\frac{-2(1 - \|\mathrm{E}[A_{n+1}]\|_\alpha)}{N}\|X_n\|_1 + K\end{aligned} \tag{13}$$

where K is a constant. So now $\forall \epsilon < \frac{2(1-\|\mathrm{E}[A_{n+1}]\|_\alpha)}{N}$, $\exists B$ s.t.

$$\begin{aligned} \mathrm{E}\left[X_{n+1}X_{n+1}^T - X_n X_n^T | X_n\right] &< \epsilon \|X_n\|_1 \\ \forall X_n : \|X_n\|_1 &< B \end{aligned} \tag{14}$$

Now Theorem 1 can be applied.

$$\begin{aligned} &\lim_{n\to\infty} \mathrm{E}[\|X_n\|_1] \\ &\leq \lim_{n\to\infty} \mathrm{E}[\|X_n\|_1] + \frac{X_{n+1}X_{n+1}^T - X_n X_n^T}{\epsilon} | \|X_n\| < B] \\ &\qquad * P(\|X_n\| < B) \\ &\leq \lim_{n\to\infty} \mathrm{E}[\|X_n\|_1] + \\ &\qquad \frac{-2\left((1-\|\mathrm{E}[A_{n+1}]\|_\alpha)(\frac{1}{N}\|X_n\|_1 - C) - C\right)}{\epsilon} + \\ &\qquad \frac{(A_{n+1}-D_n)(A_{n+1}-D_n)^T}{\epsilon} | \|X_n\| < B] \\ &\qquad * P(\|X_n\| < B) \\ &\leq \lim_{n\to\infty} \frac{2((1-\|\mathrm{E}[A_{n+1}]\|_\alpha)C + C}{\epsilon} + \\ &\qquad \mathrm{E}[\frac{(A_{n+1}-D_n)(A_{n+1}-D_n)^T}{\epsilon} | \|X_n\| < B] \\ &\qquad * P(\|X_n\| < B) \\ &\leq \lim_{n\to\infty} \frac{2((1-\|\mathrm{E}[A_{n+1}]\|_\alpha)C + C)}{\epsilon} + \\ &\qquad \mathrm{E}[\frac{(A_{n+1}-D_n)(A_{n+1}-D_n)^T}{\epsilon}] \end{aligned} \tag{15}$$

where we have used the fact that $\mathrm{E}\left[\mathrm{E}[Y|X]]X < B\right] = \mathrm{E}\left[Y|X < B\right]$ and $\epsilon < \frac{2(1-\|\mathrm{E}[A_{n+1}]\|_\alpha)}{N}$. Now let ϵ approach its maximum value, and we get

$$\begin{aligned} &\lim_{n\to\infty} \mathrm{E}[\|X_n\|_1] \\ &\leq \lim_{n\to\infty} (N + \frac{N}{2(1-\|\mathrm{E}[A_{n+1}]\|_\alpha)})C + \\ &\qquad \frac{\mathrm{E}[(A_{n+1}-D_n)(A_{n+1}-D_n)^T]}{\epsilon} \end{aligned} \tag{16}$$

All the terms can be easily evaluated given the traffic matrix. Note the linear dependency on C.

REFERENCES

[1] Shah D., Kopikare M., "Delay Bounds for the approximate Maximum weight matching algorithm for input queued switches",*To appear in IEEE INFOCOM '2002*.

[2] Leonardi E., Mellia M., Neri F., Ajmone Marsan M., "Bounds on Average Delays and Queue Size Averages and Variances in Input-Queued Cell-Based Switches", *IEEE INFOCOM 2001*, Alaska, April 2001, pp. 1095-1103.

[3] Widrow B., Stearns S.D., *Adaptive Signal Processing*, Addison Wesley Longman, 1985.

[4] K. Kar et. al, "Reduced complexity input buffered switches," *Proc. Hot Interconnects VIII*, 2000, pp. 13–20.

[5] Mc Keown N., Anantharan V., Walrand J., "Achieving 100% throughput in an input-queued switch", *IEEE INFOCOM '96*, vol. 1, San Francisco, Mar. 1996, pp. 296-302.

[6] D.Morat, J.Aracil, L.A.Dez, M.Izal y E.Magaa, "On linear prediction of Internet traffic for packet and burst switching networks", *Proceedings of International Conference on Computer Communications and Networks (ICCCN 2001)*, Scottsdale, Arizona, USA, October 2001.