Tradeoffs in Delay Guarantees and Computation Complexity for $N \times N$ Packet Switches

Michael J. Neely Eytan Modiano Charles E. Rohrs
MIT--LIDS: {mjneely@mit.edu, modiano@mit.edu, crohrs@mit.edu}

***Abstract* -- We consider the tradeoffs between packet delay guarantees and per-timeslot computation complexity in an $n \times n$ packet switch operating under the crossbar constraint. It is well known that scheduling packets every timeslot according to a Maximum Weight Matching (MWM) achieves 100% throughput. This algorithm ensures average packet delay is within $O(n)$ timeslots (where n is the number of input ports of the switch) but is quite complex to implement, requiring $O(n^3)$ computations every slot to compute the schedule. Here we develop a modified version of MWM which reduces computation complexity while still ensuring 100% throughput and offering polynomial delay bounds. Specifically, we develop a class of scheduling policies (parameterized by $\alpha \in (0, 3]$) which achieve a per-timeslot computation complexity of $O(n^{\alpha})$ and ensure $O(n^{4-\alpha})$ bounds on average delay. In particular, linear per-timeslot computation complexity is achievable with an $O(n^3)$ delay guarantee (case α=1). Furthermore, as $\alpha \to 0$, complexity can be made as low as desired while delay is held within $O(n^4)$. These results for the first time illustrate an explicit tradeoff between performance and scheduling complexity.**

I. Introduction

There is a great deal of interest in developing low complexity algorithms for scheduling packets in an $n \times n$ packet switch operating under the crossbar constraint [1-9]. In the landmark papers [1] and [2], it was independently shown that a Maximum Weight Matching algorithm (MWM) stabilizes the switch for any set of admissible (stabilizable) input rates, and hence MWM achieves 100% throughput. Such stability is guaranteed even if the input rates for each port are unknown. In [7] it is shown that this policy offers desirable delay performance by maintaining average packet delay within $O(n)$ timeslots[1] (where n is the number of input ports of the switch). However, the MWM algorithm is quite complex to implement, requiring $O(n^3)$ operations every timeslot to compute the switching matrix [11]. Such complexity makes MWM impractical for large switches operating at very high speeds.

Lower complexity algorithms which also offer 100% throughput are developed in [3, 4, 5, 6]. An algorithm developed by Tassiulas in [4] demonstrates that it is possible to achieve 100% throughput using a scheduling algorithm with *linear complexity* every timeslot (a tremendous improvement over the $O(n^3)$ complexity of MWM). However, the algorithm offers poor delay performance (as observed in simulations). It can be shown analytically that the Tassiulas algorithm keeps average packet delay within $O(n!)$ timeslots. The non-polynomial growth of this upper bound is consistent with simulation results, and hence delay may be unreasonably large as n increases.

[1] This is the best known asymptotic delay for switching algorithms operating on $n \times n$ switches with random inputs, and we conjecture it is the best performance possible.

Here, we develop a modified version of MWM which both reduces computation complexity and guarantees polynomial delay. Specifically, we describe a *class* of algorithms (parameterized by $\alpha \in (0, 3]$) which achieve a per-timeslot computation complexity of $O(n^{\alpha})$ and ensure $O(n^{4-\alpha})$ bounds on average delay. In particular, for the case α=1, linear per-timeslot computation complexity is achievable with an $O(n^3)$ delay guarantee. Furthermore, as $\alpha \to 0$ complexity can be made as low as desired while delay is held within $O(n^4)$. These results for the first time illustrate an explicit tradeoff between performance and scheduling complexity.

The idea is quite simple. Rather than compute a Maximum Weighted Match every timeslot, we allow k timeslots for the computation. The switching configuration determined by the outcome of the matching is then held fixed for another k slots while the next matching computation proceeds--using weights equal to the number of packets seen at the start of the k-slot interval. It is clear that this technique reduces the per-timeslot complexity of the $O(n^3)$ MWM algorithm at the expense of using out-of-date queue backlog information and imposing the restriction of a fixed switching configuration during the k-slot intervals. In the following analysis, we show that delay grows as $O(kn)$ because of such restrictions.

This technique is thus useful for reducing complexity at the expense of increasing delay. Furthermore, the algorithm naturally applies to systems whose physical constraints require switching configurations to be held fixed for more than one timeslot. Examples include:

-Optical switching devices which rely on slow mechanical reflectors for switching inputs to different output paths, and hence should be held in a fixed switching mode for longer durations of time (see [9]).

-Systems whose crossbar electronics operate at speeds slower than the input/output line rate.

The switching algorithms developed here suggest a host of potential improvements. For example, a Max Weight Matching algorithm distributed over k timeslots might use dynamic link weights determined by the new packet arrivals every slot. Furthermore, switching configurations may be altered during the k-slot interval if a better match is found by other means. Example heuristics of this type are considered.

In the next section, we describe the set of admissible input rates $\vec{\lambda}$ which form the stability region of the system. In Section III the modified MWM policy is developed and the delay/complexity tradeoff is illustrated for the case when arrivals occur *iid* every timeslot. In Section IV we consider non-iid arrivals and show that the policy is robust to arbitrary changes in the input rates from timeslot to timeslot. Finally, in Section V we consider some simple heuristic improvements to the given algorithms.

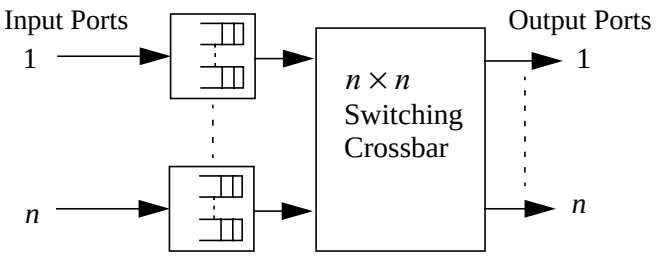

Figure 1: An $n \times n$ packet switch with n input ports and n^2 input queues.

II. The Stability Region

Consider the packet switch with n input ports and n output ports shown in Fig. 1. The switch operates in discrete time, where timeslots are normalized to 1 unit. Every timeslot, a random number of packets arrive at each input and are queued according to their intended destinations, so that queue (i, j) contains packets from input i destined for output j. If a connection is established between input i and output j at the beginning of a timeslot, exactly one packet waiting in queue (i,j) can be transferred during the slot. All connections are established on timeslot boundaries ($t = 0, 1, 2, \ldots$), and the system operates according to the *crossbar constraint*: during a timeslot, each input can transfer at most one packet to an output, and each output can receive at most one packet from an input. Hence, at most n connections can be established in any timeslot, and the set of valid connections consists of the $n!$ possible permutations between inputs and outputs. Define:

$N_{ij}(t)$ = Number of packets in queue *(i,j)* at time *t.*

$a_{ij}(t)$ = Number of new arrivals to queue *(i,j)* during $[t, t+1)$.

$c_{ij}(t)$ = connection decision for queue *(i,j)* at time t

(where $c_{ij}(t) = 1$ if a connection is established, and 0 else).

$$\lambda_{ij} = \lim_{t \to \infty} \frac{1}{t} \sum_{\tau = 1}^{t} a_{ij}(\tau).$$

The dynamics of queue *(i,j)* proceed according to the equation:

$$N_{ij}(t+1) = max(N_{ij}(t) - c_{ij}(t), 0) + a_{ij}(t) \quad (1)$$

where every timeslot the crossbar constraint limits the matrix of control decisions $(c_{ij}(t))$ to the set of $n \times n$ permutation matrices (i.e., the set of 0-1 matrices $M = \{M_1, M_2, \ldots, M_{n!}\}$ with exactly one "1" in each row and column).

The stability region is the set Ω such that the system cannot be stabilized if the rate matrix $(\lambda_{ij}) \notin \Omega$, and the system *can* be stabilized using some scheduling algorithm whenever the rate matrix is *strictly interior* to Ω (the system may or may not be stabilizable if (λ_{ij}) is on the boundary of the stability region). It is well known that the stability region is the set of all rate matrices (λ_{ij}) such that the sum of the entries in any row or column is less than or equal to 1:

$$\Omega = \left\{ (\lambda_{ij}) \mid \sum_i \lambda_{ij} \le 1, \sum_j \lambda_{ij} \le 1 \right\} \quad (2)$$

It is clear that $(\lambda_{ij}) \in \Omega$ is a necessary condition for stability. Indeed, if the rate matrix is not within Ω, then at least one of the inequality constraints in (2) must be violated, and hence some input port or output port is oversubscribed--leading to an infinite buildup of packets in the system with probability 1.

To show sufficiency, it is useful to consider the subset of Ω consisting of points where the inequalities in (2) are met with equality. The Birkhoff-Von Neumann Theorem [10] states that this subset can be expressed as the convex combination of permutation matrices:

Theorem (Birkhoff-Von Neumann):

Convex Hull$\{M_1, M_2, \ldots, M_{n!}\} =$

$$\left\{ (\lambda_{ij}) \mid \sum_i \lambda_{ij} = 1, \sum_j \lambda_{ij} = 1 \right\}. \square$$

The Birkhoff-Von Neumann Theorem thus implies that the convex hull of all permutation matrices forms the *dominant subset* of Ω: The subset of Ω such that every rate matrix in Ω is entrywise dominated by a point in the subset. Thus, for all $(\lambda_{ij}) \in \Omega$, there exists a matrix (μ_{ij}) in *Convex Hull*$\{M_1, M_2, \ldots, M_{n!}\}$ such that $\lambda_{ij} \le \mu_{ij}$ for all i and j.

This fact can be used to show that (λ_{ij}) being strictly interior to Ω is a sufficient condition for stability. To see this, suppose (λ_{ij}) is strictly interior to Ω and choose a matrix (μ_{ij}) within *Convex Hull*$\{M_1, M_2, \ldots, M_{n!}\}$ such that $(\lambda_{ij}) < (\mu_{ij})$ (where the inequality is considered entrywise). From the Birkhoff-Von Neumann Theorem, we can find non-negative values $\{p_1, p_2, \ldots, p_{n!}\}$ such that $\sum p_m = 1$, where:

$$(\mu_{ij}) = p_1 M_1 + p_2 M_2 + \ldots + p_{n!} M_{n!} \quad (3)$$

To stabilize the system, consider the following policy: Every timeslot, randomly choose a control matrix $(c_{ij}(t))$ from among the set of all permutation matrices, such that matrix M_i is chosen with probability p_i. From (3) it follows that every timeslot a server connection is established for queue *(i,j)* with probability μ_{ij}. This effectively creates a geometric "service time" for each packet, and hence each queue *(i,j)* is transformed into a slotted G/G/1 queue with arrival rate λ_{ij} and service rate μ_{ij}. Because $\lambda_{ij} < \mu_{ij}$, each queue is stable.

Example: Suppose all inputs are Poisson with uniform rates $\lambda < 1/n$, and let $\mu_{ij} = 1/n$ for all i and j. Each queue is then equivalent to a slotted M/G/1 queue with geometric service time and loading $\rho = \lambda n$. The average delay can be easily calculated (see [12]):

$$\overline{W} = \frac{n - 1/2}{1 - \rho} + 1. \quad (4)$$

The above delay is clearly $O(n)$, and hence delay scales linearly as the number of input ports n is increased. One might suspect that delay can be reduced if the randomness of the service algorithm is replaced by a periodic schedule which services each queue (i,j) a fraction of time μ_{ij}. Indeed, for uniform traffic, it is easy to see that switching periodically amongst the n

cyclic permutation matrices provides a server to each queue every n timeslots. The resulting system is similar to an M/D/1 queue, and delay can be calculated using the techniques in [12]:

$$\overline{W}_{\text{periodic schedule}} = \frac{n}{2(1-\rho)} + 1 . \qquad (5)$$

Although the above delay is reduced from the delay of the random control algorithm, it still scales linearly with n. Intuitively, this is because each input port can service at most one of its n queues per timeslot, and hence it takes an average of $n/2$ timeslots for any queue to receive a server. We conjecture that $O(n)$ delay is the best asymptotic delay performance that any algorithm can provide when inputs are random.[2]

III. Delay / Complexity Tradeoffs

Here we consider switch scheduling when the input rates (λ_{ij}) are interior to the stability region Ω but are unknown to the scheduler. In [1], [2] it was shown that a Maximum Weight Matching algorithm which chooses a permutation matrix $(c_{ij}(t))$ to maximize $\sum_{i,j} N_{ij}(t)c_{ij}(t)$ every timeslot stabilizes the system whenever the system is stabilizable, and hence allows for 100% throughput. In [7] a packet delay analysis for the MWM algorithm was presented, which demonstrated that packet delay is $O(n)$--the best known asymptotic delay for any algorithm. However, the MWM algorithm is complex to implement, requiring $O(n^3)$ operations to be performed within a single timeslot [11]. Here we present a class of algorithms which allow per-timeslot implementation complexity to be reduced at the expense of increasing packet delay. The idea is to reduce complexity by holding switching configurations fixed for k timeslots--allowing more time for completion of the matching algorithm.

Let t represent the starting time for such a k-slot interval. The number of packets k timeslots into the future satisfies:

$$N_{ij}(t+k) \le max(N_{ij}(t) - kc_{ij}(t), 0) + \sum_{r=0}^{k-1} a_{ij}(t+r). \qquad (6)$$

Note that the inequality is due to the fact that new packets arriving in the interval $[t, t+k-1)$ can potentially be served during this k-slot interval, and hence will not be in the system at time $t+k$. In the case $k=1$, the inequality (6) can be written as an equality, and reduces to (1).

If a policy computes a Max Weight Match during a k-slot interval $[t, t+k)$, the resultant match will be implemented as a fixed server configuration on the next interval $[t+k, t+2k)$. Hence, if the computation uses queue occupancy data $(N_{ij}(t))$ from timeslot t, this information will be more than k slots out of date (and as much as $2k$ slots out of date) when the server implementation is finally applied. It is thus better to use for the matching the *estimated* packet occupancy at time $t+k$ (when the implementation starts), where this estimate is based on the known packet occupancy at time t and the known server configuration during $[t, t+k)$. We thus define:

$\tilde{N}_{ij}(t)$ = Number of packets currently in queue (i,j) that did not arrive during the past k-slot interval $(t-k, t]$.

Notice that $N_{ij}(t)$ and $\tilde{N}_{ij}(t)$ remain close together for all time t:

$$N_{ij}(t) \le \tilde{N}_{ij}(t) + \sum_{r=0}^{k-1} a_{ij}(t-k+r) \le N_{ij}(t) + \sum_{r=0}^{k-1} a_{ij}(t-k+r) \qquad (7)$$

and hence $\tilde{N}_{ij}(t)$ serves as a good estimate of $N_{ij}(t)$. Furthermore, the $\tilde{N}_{ij}(t+k)$ values can be obtained exactly from knowledge that is available at time t:

$$\tilde{N}_{ij}(t+k) = max(N_{ij}(t) - kc_{ij}(t), 0). \qquad (8)$$

We now define a class of scheduling policies π_k, parameterized by k, which use the $\tilde{N}_{ij}(t)$ information as follows:

Scheduling Policy π_k: At times $t=\{0, k, 2k, 3k, ...\}$ perform the following:

- Configure the switch matrix $(c_{ij}(t))$ to the permutation calculated by the MWM algorithm during the past k-slot interval, and maintain this configuration for the new interval $[t, t+k)$. (If $t=0$, simply use the identity permutation matrix.)
- Calculate $\tilde{N}_{ij}(t+k)$ values using $N_{ij}(t)$ and $(c_{ij}(t))$ according to (8).
- Use $\tilde{N}_{ij}(t+k)$ values as weights in an MWM computation during the interval $[t, t+k)$.

To analyze the performance of this strategy, we use a Lyapunov drift technique for bounding average occupancy.

A. Lyapunov Drift. Consider a general system of J queues, and let $\vec{N}(t) = (N_1(t), N_2(t), \ldots, N_J(t))$ represent the vector of queue occupancy which changes state at times $t=\{0, 1, 2,...\}$. Assume the resulting process is a Markov chain. We develop a criterion using Lyapunov drift which ensures a steady state occupancy exists and provides a bound on average delay. The technique uses a well established theory of Lyapunov functions applied to queueing systems [1,2,7,8], and the bounding technique is similar to the method for establishing delay bounds developed in [7].

The vector notation here is for convenience. Note that the matrix of packet occupancies $(N_{ij}(t))$ in a switch can be written as a stacked vector, and proceeds according to a Markov chain if the system is sampled every k timeslots and if the control decisions $(c_{ij}(t))$ are appended to the system state.

Consider a Lyapunov function defined on the state of queue occupancies: $L(\vec{N}) = \sum_{j=1}^{J} N_j^2$. In [2] the following theorem is given, which establishes the existence of a steady state distribution for the vector process $\vec{N}(t)$:

[2] This conjecture does not hold when inputs are not random. Examples of periodic inputs and scheduling algorithms can be constructed which ensure constant delay independent of n.

Theorem 1: Suppose there exists a positive number γ and a bounded region Λ of the state space of the Markov process $\vec{N}(t)$ such that:

(i) $E[L(\vec{N}(t+1))|\vec{N}(t)] < \infty$ for all $\vec{N}(t)$

(ii) $E[L(\vec{N}(t+1)) - L(\vec{N}(t))|\vec{N}(t)] \le -\gamma$

whenever $\vec{N}(t) \notin \Lambda$

Then a steady state distribution for $\vec{N}(t)$ exists. ❑

Below we present a condition for the Lyapunov drift which, together with Theorem 1, establishes a finite upper bound on the average values of $\vec{N}(t)$. The proof uses a simple telescoping series argument similar to the technique presented in [7].

Theorem 2: Suppose there exist positive values $\{\gamma_j\}$ and B such that for all $\vec{N}(t)$:

$$E[L(\vec{N}(t+1)) - L(\vec{N}(t))|\vec{N}(t)] \le B - \sum_{j=1}^{J} \gamma_j N_j(t). \quad (9)$$

then: (a) A steady state distribution exists for $\vec{N}(t)$.

(b) The steady state values of queue occupancy are finite and satisfy:

$$\sum_j \gamma_j \overline{N}_j \le B. \quad (10)$$

Proof of (a): From (9), it is clear that condition (i) of Theorem 1 is satisfied. To show that (ii) is also satisfied, fix a constant $\alpha > 0$ and define the set $\Lambda = \left\{\vec{N} \middle| \sum_j \gamma_j N_j \le \alpha + B\right\}$. From (9), it follows that the one step Lyapunov drift for $\vec{N}(t)$ is less than or equal to $-\alpha$ whenever $\vec{N}(t) \notin \Lambda$. ❑

Proof of (b): In a manner similar to the method demonstrated in [7], we take expectations of both sides of (9) and sum from $t = 0, \ldots, \tau - 1$ to obtain:

$$E[L(\vec{N}(\tau))] - E[L(\vec{N}(0))] \le \tau B - \sum_{t=0}^{\tau-1} \sum_{j=1}^{J} \gamma_j E[N_j(t)]. \quad (11)$$

Dividing (11) by τ and shifting terms, we find:

$$\sum_j \gamma_j \left[\frac{1}{\tau}\sum_{t=0}^{\tau-1} E[N_j(t)]\right] \le B + \frac{1}{\tau} E[L(\vec{N}(0))]. \quad (12)$$

Now, because a steady state distribution exists for $\vec{N}(t)$ and because time averages $\frac{1}{\tau}\sum_{t=0}^{\tau-1} E[N_j(t)]$ are bounded, it follows that first moments exist and are reached as limits of the time average. Taking limits of (12) as $\tau \to \infty$ thus establishes (10) and concludes the proof. ❑

B. Performance Analysis of strategy π_k: We use the Lyapunov drift Theorem 2 to obtain a bound on the steady state behavior of the $(\tilde{N}_{ij}(t))$ process, which allows a delay bound for the π_k scheduling strategy to be developed. Let t be the beginning of a k-slot interval for the π_k strategy. The matrix $(\tilde{N}_{ij}(t+k))$ can be bounded in terms of $(\tilde{N}_{ij}(t))$ using the $(c_{ij}(t))$ control decisions according to the inequality:

$$\tilde{N}_{ij}(t+k) \le max(\tilde{N}_{ij}(t) - kc_{ij}(t), 0) + \sum_{r=0}^{k-1} a_{ij}(t-k+r) \quad (13)$$

i.e., the value of $(\tilde{N}_{ij}(t+k))$ is less than or equal to its value at time t, minus the departures during the past k-slot interval, plus the new arrivals that occurred in the second to last k-slot interval. Because some of these new arrivals may be served before time $t+k$, (13) is an inequality instead of an equality.

Consider now the inputs $a_{ij}(t)$ to the switch. Assume that on each timeslot, a new matrix of arrivals occurs *iid* with some distribution $f(\underline{a})$ (where $\underline{a}$ represents a matrix of packet arrivals). Note that although arrivals in different timeslots are independent, arrivals to different queues in the same timeslot may be correlated. For simplicity, here we assume that no more than one new packet can arrive to an input port in any given timeslot:

$$\sum_{j=1}^{n} a_{ij} \in \{0, 1\} \text{ for all input ports } i \in \{1, \ldots, n\}.$$

This assumption is not necessary, but it simplifies the resulting expression for average packet delay, and it is consistent with the physical constraints of $n \times n$ packet switches.

Let the input rate matrix $(\lambda_{ij}) = (E[a_{ij}(t)])$ for the switch be strictly interior to the stability region Ω. Let $\varepsilon > 0$ be the largest value that can be added to each entry of the rate matrix such that the resulting matrix remains within the stability region, i.e., $(\lambda_{ij} + \varepsilon)$ is on the boundary of Ω. Viewing $\vec{\varepsilon} = (\varepsilon, \ldots, \varepsilon)$ as a stacked vector with n^2 entries, we can thus define the *Euclidean distance* d of the rate matrix to the boundary of the capacity region to be the norm of $\vec{\varepsilon}$. Clearly $\|\vec{\varepsilon}\|^2 = n^2\varepsilon^2$, and hence:

$$d = n\varepsilon. \quad (14)$$

To analyze delay performance of an $n \times n$ switch, it is appropriate to keep d constant as n scales. This effectively preserves the same loading for each input port $i \in \{1, \ldots, n\}$. For example, if input rates to each queue (i,j) are uniformly ρ/n (where $\rho < 1$ represents the total loading on an input), then $\varepsilon = (1-\rho)/n$, and $d = (1-\rho)$ is held constant.

Let $\lambda_{av} = \frac{1}{n}\sum_{i=1}^{n}\left(\sum_{j=1}^{n}\lambda_{ij}\right)$ denote the average rate of packets entering an input port i (averaged over all $i \in \{1, \ldots, n\}$).

Theorem 3: For any integer $k>0$, the policy π_k has per-timeslot complexity $O(n^3/k)$ and ensures an average delay of $O(nk)$. Specifically, average delay $\overline{W}$ satisfies: $\overline{W} \le \frac{kn}{d\lambda_{av}} + k$.

Proof: That the per-timeslot complexity of π_k is $O(n^3/k)$ follows immediately from the fact that the $O(n^3)$ MWM computation is distributed over k slots. To prove the delay bound, we

define the quadratic Lyapunov function $L(\underline{\tilde{N}}) = \sum_{i,j} \tilde{N}_{i,j}^2$, where $\underline{\tilde{N}} = (\tilde{N}_{ij})$. Define:

$$A_{ij}(t) = \sum_{r=0}^{k-1} a_{ij}(t-k+r). \qquad (15)$$

For convenience, we neglect the time subscripts and use $\tilde{N}_{ij}$, A_{ij}, and c_{ij} to respectively represent $\tilde{N}_{ij}(t)$, $A_{ij}(t)$, and $c_{ij}(t)$. Using (15) in (13), we have:

$$\tilde{N}_{ij}^2(t+k) \le \tilde{N}_{ij}^2 + k^2 c_{ij}^2 - 2\tilde{N}_{ij}kc_{ij} + A_{ij}^2 + 2A_{ij}max(\tilde{N}_{ij} - kc_{ij}, 0)$$
$$\le \tilde{N}_{ij}^2 + k^2 c_{ij}^2 + A_{ij}^2 - 2k\tilde{N}_{ij}(c_{ij} - A_{ij}/k) \qquad (16)$$

Taking conditional expectations of (16) given $\tilde{N}_{ij}(t)$ and summing over all *i* and *j*, we have:

$$E[L(\underline{\tilde{N}}(t+k)) - L(\underline{\tilde{N}}(t))\big|\underline{\tilde{N}}(t)] \le$$
$$\sum_{i,j} k^2 E[c_{ij}^2\big|\underline{\tilde{N}}] + \sum_{i,j} E[A_{ij}^2] - 2k\sum_{i,j}\tilde{N}_{ij}(E[c_{ij}\big|\underline{\tilde{N}}] - \lambda_{ij}) \qquad (17)$$

where the *iid* nature of packet arrivals was used to substitute $E[A_{ij}\big|\underline{\tilde{N}}] = k\lambda_{ij}$ above.

Note that the control matrix $(c_{ij}(t))$ is a deterministic function of the $\underline{\tilde{N}}$ matrix, determined by the policy π_k. Hence, we can write $E[c_{ij}\big|\underline{\tilde{N}}]$ as $c_{ij}^{\pi}(t)$ --the control decision made by policy π_k at time *t*. The policy π_k chooses a permutation matrix $c_{ij}^{\pi}(t)$ from the set of all possible permutation matrices (c_{ij}) to maximize $\sum_{i,j}\tilde{N}_{ij}(t)c_{ij}$. However, by the Birkhoff-Von Neumann Theorem, all points in the stability region Ω can be dominated by a convex combination of permutation matrices. Hence, the resulting value of $\sum_{i,j}\tilde{N}_{ij}(t)c_{ij}^{\pi}(t)$ also maximizes the linear function $\sum_{i,j}\tilde{N}_{ij}(t)\gamma_{ij}$ over all rates $(\gamma_{ij}) \in \Omega$ (see [14]). Because $(\lambda_{ij} + \varepsilon)$ is in Ω, it follows that:

$$\sum_{i,j}\tilde{N}_{ij}(t)c_{ij}^{\pi}(t) \ge \sum_{i,j}\tilde{N}_{ij}(t)(\lambda_{ij} + \varepsilon). \qquad (18)$$

Using (18) in (17), we have:

$$E[L(\underline{\tilde{N}}(t+k)) - L(\underline{\tilde{N}}(t))\big|\underline{\tilde{N}}(t)]$$
$$\le \sum_{i,j} k^2 E[c_{ij}^2\big|\underline{\tilde{N}}] + \sum_{i,j} E[A_{ij}^2] - 2k\sum_{i,j}\tilde{N}_{ij}\varepsilon$$
$$\le k^2\sum_{i,j} c_{ij}^{\pi} + \sum_{i,j} E[kA_{ij}] - 2k\sum_{i,j}\tilde{N}_{ij}\varepsilon \qquad (19)$$
$$= k^2 n + k^2 n\lambda_{av} - 2k\sum_{i,j}\tilde{N}_{ij}\varepsilon \qquad (20)$$

where (19) follows because both c_{ij} and a_{ij} are in {0, 1}, so $c_{ij}^2 = c_{ij}$ and $A_{ij} \le k$. The condition (20) is the negative drift condition we are after. Applying the result of Theorem 2, we find that a steady state value for $\tilde{N}_{ij}$ exists and satisfies:

$$\sum_{i,j} E[\tilde{N}_{ij}] \le \frac{k^2 n(1+\lambda_{av})}{2k\varepsilon}. \qquad (21)$$

Simplifying (21) and using the fact that $\lambda_{av} < 1$ as well as the fact that $d = n\varepsilon$, we have:

$$\sum_{i,j} E[\tilde{N}_{ij}] \le \frac{kn^2}{d} \qquad (22)$$

Using (7) to express a bound for the actual number of packets in the switch (N_{ij}) in terms of the bound for $(\tilde{N}_{ij})$ yields:

$$\sum_{i,j} E[N_{ij}] \le \frac{kn^2}{d} + kn\lambda_{av}. \qquad (23)$$

Dividing both sides of (23) by the total input rate $n\lambda_{av}$ and using Little's Theorem yields:

$$\overline{W} \le \frac{kn}{d\lambda_{av}} + k \qquad (24)$$

and proves the theorem. ❑

The 1/*d* behavior of the waiting time bound is worth noting. Recall that *d* can be viewed as the Euclidean distance of the rate matrix (λ_{ij}) to the boundary of the stability region (in the sense that ε is the maximum value that can be added to all n^2 entries of the rate matrix such that the matrix $(\lambda_{ij}) + (\varepsilon)$ remains within the stability region, and the norm of the stacked vector $\grave{\varepsilon}$ is *d*). Thus, the bound grows asymptotically like 1/*d* as the rate matrix is pushed towards the boundary. Such behavior is characteristic of queueing systems, as illustrated by the standard P-K formula for average occupancy in an M/G/1 queue [12].

The relationship between complexity and delay is now apparent. Indeed, choose a value $\alpha \in [0, 3]$. From Theorem 3, we can choose an integer $k = \lceil n^{3-\alpha} \rceil$ such that policy π_k has per-timeslot complexity $O(n^{\alpha})$ and a delay bound of $O(n^{4-\alpha})$.

IV. Robustness To Input Rate Changes

The analysis in the previous section assumes *iid* arrivals. Here we demonstrate that the π_k algorithm is robust to arbitrary changes in the input rate matrix (λ_{ij}) as long as the matrix remains within the stability region Ω at each timeslot. Specifically, suppose that the input rate matrix to the system is $\underline{\lambda}^{(1)}$ for a certain duration of time, then changes to $\underline{\lambda}^{(2)}$ --perhaps due to changing user demands. This change will be reflected in the backlog that builds up in the queues of the system. Because the policy π_k bases decisions on the size of the queues, it reacts smoothly to such changes in the input statistics.

Formally, this situation is modeled by defining an input distribution $f_t(\underline{a})$ on the arrival matrix $(a_{ij}(t))$ at each timestep t. The $f_t(\underline{a})$ distributions are arbitrary and unknown to the scheduler, although we assume they yield rate matrices $(\lambda_{ij}(t)) = (E[a_{ij}(t)])$, all of which are within the stability region. System dynamics thus proceed according to a time-varying Markov chain. Although there is no notion of steady state behavior for a time-varying chain, below we show that time averages are well behaved.

Assume that there is some distance d such that all instantaneous arrival rate matrices $(\lambda_{ij}(t))$ are at least a distance d from the boundary of the stability region, i.e., for $\varepsilon = d/n$, $(\lambda_{ij}(t) + \varepsilon) \in \Omega$ for all t. The one-step Lyapunov drift can be analyzed for this system as before. Indeed, notice that inequality (17) remains valid, where expectations are taken over the t^{th} arrival distribution $f_t(\underline{a})$. Following through with the same derivation as in Theorem 3, it can be shown:

$$E[L(\tilde{\underline{N}}(t+k)) - L(\tilde{\underline{N}}(t)) \,|\, \tilde{\underline{N}}(t)] \le k^2 n(1 + \lambda_{av}(t)) - \frac{2k}{n}\sum_{i,j}\tilde{N}_{ij}d \quad (25)$$

where $\lambda_{av}(t)$ is the average rate to an input port taken over the t^{th} arrival distribution. From a telescoping series argument similar to Theorem 2, it can be shown that:

$$\lim_{\tau\to\infty}\sup\frac{1}{\tau}\sum_{t=0}^{\tau-1}\left(\sum_{i,j}\overline{N}_{ij}(t)\right) \le \frac{kn^2}{d} + kn. \quad (26)$$

V. Heuristic Improvements

The policy π_k suggests several heuristic improvements. For example, rather than use fixed values of queue occupancy while the MWM computation is running, the matching scheme might use dynamic weight updates based on the new data entering the system. Another improvement is to enable the switching matrix to switch to an alternate configuration in the middle of a k-slot interval when the new configuration is clearly preferable. For example, if the switch has cleared all packets in the n queues it is serving under a given connectivity matrix, it is better to change to another matrix which can serve packets in other queues rather than remain idle for the rest of the k-slot interval.

Finally, the MWM computation often requires less that $O(n^3)$ complexity. For example, one set of weights may require the computation to run for 50 timesteps, while in the next k-slot interval, only 15 timesteps are needed to find the max weight match. A modified π_k algorithm which allows for these variable computation times could thus initiate the next MWM computation immediately if the previous one finishes early. Such techniques intuitively improve the operation of policy π_k, and hence potentially improve upon the delay/complexity tradeoff profile.

VI. Conclusions

We have demonstrated a class of switch scheduling policies π_k for an $n \times n$ packet switch which offer 100% throughput and enable delay performance to be traded off for reduced per-timeslot computation complexity. It was shown that for any parameter $\alpha \in [0, 3]$, a policy can be designed that has a per-timeslot computation complexity of $O(n^\alpha)$ and provides an average delay within $O(n^{4-\alpha})$. While much of the previous literature on $n \times n$ packet switches has concentrated on stability results, this work for the first time enables an explicit characterization of the tradeoffs between performance guarantees and per-timeslot computation complexity.

The policies π_k were shown to be robust to arbitrary changes in the input rates from one timeslot to the next provided that the instantaneous arrival rates remain within the stability region Ω. Furthermore, the described policies naturally apply to systems whose physical constraints preclude rapid switching changes. It is remarkable that 100% throughput can still be achieved when switching rates are decreased well below the linespeed. We believe these results are fruitful and provide direction for future research into designing powerful scheduling algorithms which offer small packet delay with low implementation complexity.

References:

[1] N. McKeown, V. Anantharam, and J. Walrand, "Achieving 100% Throughput in an Input-Queued Switch," in *Proc. IEEE INFOCOM,* San Francisco, CA, Mar. 1996, pp. 296-302.

[2] L. Tassiulas and A. Ephremides, "Stability Properties of Constrained Queueing Systems and Scheduling Policies for Maximum Throughput in Multihop Radio Networks," *IEEE Transactions on Automatic Control*, Vol. 37, no. 12, Dec. 1992.

[3] A.Mekkittikul, N.McKeown, "A Practical Scheduling Algorithm to achieve 100% Throughput in Input-Queued Switches," *IEEE INFOCOM Proceedings* 1998, pp. 792-799.

[4] L. Tassiulas, "Linear Complexity Algorithms for Maximum Throughput in Radio Networks and Input Queued Switches," *IEEE Proc. of INFOCOM*, 1998.

[5] D. Shah, "Stable Algorithms for Input Queued Switches," *Proceedings of the 39th Annual Allerton Conf. on Communication, Control, and Computing*, Oct. 2001.

[6] I. Keslassy and N. McKeown, "Analysis of Scheduling Algorithms that Provide 100% Throughput in Input-Queued Switches," *Proceedings of the 39th Annual Allerton Conf. on Communication, Control, and Computing*, Oct. 2001.

[7] E. Leonardi, M. Mellia, F. Neri, and M. Ajmone Marson, "Bounds on Average Delays and Queue Size Averages and Variances in Input-Queued Cell-Based Switches," *IEEE INFOCOM Proceedings* 2001, vol 2.

[8] P.R.Kumar, S.P.Meyn, "Stability of Queueing Networks and Scheduling Policies," *IEEE Transactions on Automatic Control*, vol.40,.n.2, Feb. 1995, pp.251-260.

[9] R.L. Cruz and Saleh Al-Harthi, "Packet Scheduling with Switch Configuration Delays," *Proceedings of the 39th Annual Allerton Conf. on Communication, Control, and Computing*, Oct. 2001.

[10] G. Birkhoff, "Tres Observaciones Sobre el Algebra Lineal," Univ. Nac. Tucuman Rv. Ser. A5, pp. 147-150, 1946.

[11] H.A.B.Saip, C.L.Lucchesi, "Matching Algorithms for Bipartite Graphs," Relatorio Tecnico DCC-03/93.

[12] D.P.Bertsekas and R. Gallager. *Data Networks*. Englewood Cliffs, NJ: Prentice-Hall, 1992.

[13] R.G. Gallager. *Discrete Stochastic Processes*. Boston: Kluwer Academic Publishers, 1996.

[14] D.P. Bertsekas. *Convex Analysis & Optimization*: Ch.1 prob. 23. http://www.mit.edu:8001//people/dimitrib/home.html.