Logarithmic Delay for $N \times N$ Packet Switches Under the Crossbar Constraint

Michael J. Neely , Eytan Modiano , Yuan-Sheng Cheng

***Abstract*— We consider the fundamental delay bounds for scheduling packets in an $N \times N$ packet switch operating under the crossbar constraint. Algorithms that make scheduling decisions without considering queue backlog are shown to incur an average delay of at least $O(N)$. We then prove that $O(\log(N))$ delay is achievable with a simple frame based algorithm that uses queue backlog information. This is the best known delay bound for packet switches, and is the first analytical proof that sublinear delay is achievable in a packet switch with random inputs.**

***Index Terms*— stochastic queueing analysis, scheduling, optimal control**

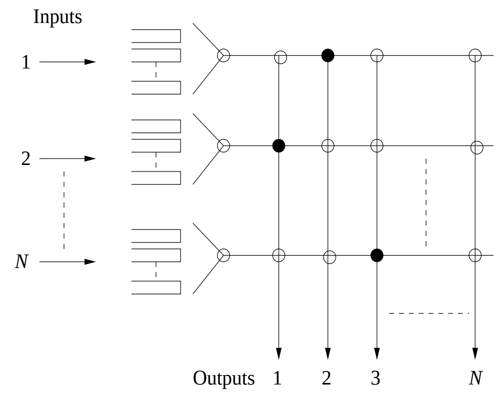

Fig. 1. An $N \times N$ packet switch under the crossbar constraint.

I. Introduction

We consider an $N \times N$ packet switch with N input ports an N output ports, shown in Fig. 1. The system operates in slotted time, and every timeslot packets randomly arrive at the inputs to be switched to their destinations. Scheduling is constrained so that each input can transfer at most one packet per timeslot, and outputs can receive at most one packet per timeslot. This constraint arises from the physical limitations of the *crossbar switch fabric* that is commonly used to transfer packets from inputs to outputs, and gives rise to a very rich problem in combinatorics and scheduling theory. This problem has been extensively studied over the past decade [1]-[21], and remains an important topic of current research. This is due to both its technological relevance to high speed switching systems and its pedagogical example as a network complex enough to inspire interesting research yet simple enough for an extensive network theory to be developed.

In this paper, we show that if the matrix of input rates to the switch has a sufficient number of non-negligible entries (to be made precise in Section III), then any scheduling strategy that does not consider queue backlog information necessarily incurs an average delay of at least $O(N)$. Strategies that do not consider backlog have been proposed in a variety of contexts, including work by Chang et al. [2][3], Leonardi et al. [4], Koksal [5], and Andrews and Vojnović [6]. The basic idea is to construct a randomized or periodic scheduling rule precisely matched for known input rates. If these rates are indeed known a-priori and do not change with time, then such scheduling offers arbitrarily low per-timeslot computation complexity, as any startup complexity associated with computing the scheduling rule is mitigated as the same rule is repeatedly used for all time.

The $O(N)$ delay result introduces an intuitive tradeoff between delay and implementation complexity, as algorithms which do not consider backlog information may have lower complexity yet necessarily incur delay that grows linearly in the size of the switch. To improve delay, we construct a simple algorithm called Fair-Frame that uses queue backlog information when making scheduling decisions. For independent Poisson inputs, we show that the Fair-Frame algorithm stabilizes the system and provides $O(\log(N))$ delay whenever the input rates are within the switch capacity region. This work for the first time establishes that sub-linear delay is possible in an $N \times N$ switch. Furthermore, the proof is simple and provides the intuition that logarithmic delay is achievable in any single-hop network with a

Michael J. Neely is with the Department of Electrical Engineering, University of Southern California, Los Angeles, CA 90089 USA (http://www-rcf.usc.edu/∼mjneely).

Eytan Modiano is with the Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, Cambridge, MA 02139 USA. Yuan-Sheng Cheng is with the Department of Electrical Engineering, University of Southern California.

This work was presented in part at the IEEE Workshop on High Performance Switching and Routing (HPSR), April 2004.

capacity region that is described by a polynomial number of constraints. Such delay improvement is achieved by taking advantage of statistical multiplexing gains, which is not possible for backlog-unaware algorithms.

Previous work in scheduling is found in [1]-[21]. In [2] it is shown that stable scheduling can be achieved with a queue length-oblivious strategy by using a Birkhoff-Von Neumann decomposition on the known arrival rate matrix. In [7] and [8], it was shown that scheduling according to a *Maximum Weighted Match* (MWM) every timeslot stabilizes the switch whenever possible and does not require prior knowledge of the input rates. Computation of maximum weight matches requires $O(N^3)$ operations per slot. In [9] the average delay under the MWM algorithm was shown to be no more than $O(N)$. We note that MWM scheduling is queue length-aware, and hence it may be possible to tighten the delay bound to less than $O(N)$, as is suggested in the simulations of [9]. However, $O(N)$ delay is the tightest known analytical bound for MWM scheduling, and was previously the best known delay bound for *any* algorithm for a switch with random (Poisson) inputs. Queue length-aware scheduling has also been considered for multi-cast switches, for example, in [10] [11].

In [12] it is shown that if a switch has an internal *speedup of* 2 (allowing for two packet transfers from input to output every timeslot), then exact output queue emulation can be achieved via stable marriage matchings, yielding optimal $O(1)$ delay. To date, there are no known delay optimal scheduling strategies for packet switches without speedup. However, work in [13] considers a loss-rate optimal scheduling algorithm for a 2×2 switch with finite buffers.

Frame based approaches for stabilizing switches and networks with deterministically constrained traffic are considered in [14] [15] [16], and in [17] it was shown that a frame based algorithm using 'greedy' maximum size matches can be used to stabilize an $N \times N$ packet switch with Poisson inputs. Related early work in this area, developed for the context of satellite switched time division multiple access (SS/TDMA) systems, considers the problem of minimizing the average waiting time in a batch of packets that arrive to a switching system at time 0 [18] [19]. The results of [18] [19] can perhaps be used to yield a constant factor improvement in average packet delay for frame based switching mechanisms, at the cost of increased scheduling complexity (discussed in more detail in Section IV-B).

Complexity and delay tradeoffs are explored in [20], where an achievable complexity-delay curve is established allowing for stable scheduling at any arbitrarily low computation complexity with a corresponding tradeoff in average delay. Similar complexity reductions were developed in [21]. In this paper, we show that the $(complexity, delay)$ operating point of the Fair-Frame algorithm sits below the curve achieved by the class of algorithms given in [20]. Indeed, Fair-Frame offers logarithmic delay and can be implemented with $O(N^{1.5}\log(N))$ total operations per timeslot. The combination of low complexity and low delay makes Fair-Frame competitive even with output queue emulation strategies in switches with a speedup of 2.

In the next section, we describe the capacity region of the $N \times N$ packet switch and present a simple stabilizing algorithm designed for known arrival rates that achieves $O(N)$ delay without considering queue backlog. In Section III it is shown that $O(N)$ delay is necessary for any such backlog-independent algorithm. In Section IV, the Fair-Frame algorithm is developed and shown to enable $O(\log(N))$ delay. Sections V and VI discuss implementation issues and provide simulation results for uniform, non-uniform, and bursty traffic.

II. The Crossbar Constraint

Consider the $N \times N$ packet switch in Fig. 1. At each input, memory is sectioned into distinct storage buffers to form N *virtual input queues*, one for each destination. Packets randomly arrive to each input every timeslot and are placed in the virtual input queues according to their destinations. These input queues are *virtual* because it is actually only the *pointers* to the local memory location of each packet that is buffered in the appropriate queue upon packet arrival. Note that there are a total of N^2 virtual input queues, indexed by (i, j) for $i, j \in \{1, \ldots, N\}$, where queue (i, j) holds data at input i destined for output j.

Every timeslot, each input selects a packet from one of its queues and sends the packet over its transmission line. The transmission lines for the N inputs are shown in Fig. 1. These lines are drawn horizontally and intersect the vertical lines leading into the output queues. The crosspoints of these wires form a matrix, and the switch fabric allows individual crosspoints to be activated or deactivated—physically establishing a connection or disconnection between inputs and outputs. If two or more crosspoints are simultaneously activated in the same column (corresponding to the same output line) then two different packets may collide at the same output port, resulting in a corrupted signal. Likewise, if two or more crosspoints in the same row are connected, a duplicate packet is sent to the wrong destination.[1]

[1] Such a property can be considered a *feature* in multicast situations, see [5].

Crosspoint connections are hence limited to connections corresponding to the set of $N!$ *permutation matrices*: $N \times N$ matrices composed of all 0's and 1's, with exactly one "1" in each row and column.

Let $A_{ij}(t)$ represent the number of packets arriving to queue (i, j) in slot t, and let $L_{ij}(t)$ represent the current number of packets in queue (i, j). Define the control decision variables $S_{ij}(t)$ as follows:

$$S_{ij}(t) = \begin{cases} 1 & \text{if crosspoint } (i,j) \text{ is activated at slot } t \\ 0 & \text{else} \end{cases}$$

The crossbar constraint limits $(S_{ij}(t))$ to the set of permutation matrices $M = \{M_1, M_2, \ldots, M_{N!}\}$. The system evolves according to the following dynamics:

$$L_{ij}(t+1) = \max\{L_{ij}(t) - S_{ij}(t), 0\} + A_{ij}(t)$$

The goal is to choose the $(S_{ij}(t))$ matrices every timeslot in order to maintain low backlog and ensure bounded average delay.

A. Stability and Delay

Here we describe the switch capacity region and give an example of a stabilizing algorithm that does not use queue backlog information (but does use a-priori knowledge of the input rates). The algorithm is a simple variation of the well known Birkhoff-Von Neumann decomposition technique [2], and is presented to provide a representative example of a queue length-independent policy which offers $O(N)$ delay.

Assume inputs are rate ergodic and define the input rate to queue (i, j) as $\lambda_{ij} \triangleq \lim_{t\to\infty} \frac{1}{t}\sum_{\tau=0}^{t} A_{ij}(\tau)$. The *capacity region* of the switch is defined as the closure of the set of all rate matrices (λ_{ij}) which can be stabilized by the switch by using some scheduling algorithm. It is well known that the capacity region of the switch is the set of rate matrices satisfying the following $2N$ inequalities:

$$\sum_{j=1}^{N} \lambda_{ij} \leq 1 \text{ for all inputs } i \tag{1}$$

$$\sum_{i=1}^{N} \lambda_{ij} \leq 1 \text{ for all outputs } j \tag{2}$$

It is clear that the above inequalities are necessary for stability. Indeed, note that the maximum rate out of any input port is one packet per slot, and the maximum rate into any output is one packet per slot. Hence, if any of the above inequalities is violated, then some input port or output port must be overloaded—leading to an infinite buildup of packets in the system with probability 1.

Sufficiency is classically shown using a combination of results due to both Birkhoff [22] and Von-Neumann [23]. Specifically, consider a subset of the capacity region consisting of rate matrices (μ_{ij}) satisfying all inequalities (1) and (2) with equality. A theorem of Birkhoff states that this subset can be expressed as the convex combination of permutation matrices:

Fact 1. *(Birkhoff Decomposition [22])*

$$\text{Convex Hull}\{M_1, M_2, \ldots, M_{N!}\} = \Big\{(\mu_{ij}) \mid \textstyle\sum_i \mu_{ij} = 1, \sum_j \mu_{ij} = 1\Big\} \; \Box$$

A related result of Von Neumann [23] shows that any rate matrix (λ_{ij}) which satisfies the stability constraints (1) and (2) with strict inequality in all entries can be term-by-term dominated by a matrix (μ_{ij}) which satisfies all constraints with equality. By Fact 1, this dominating matrix (μ_{ij}) is within the convex hull of the permutation matrices $\{M_1, M_2, \ldots, M_{N!}\}$. This fact can immediately be used to show that (λ_{ij}) being strictly interior to the capacity region is sufficient for stability. For a simple way to see this, suppose (λ_{ij}) is strictly interior to the capacity region, and choose a matrix (μ_{ij}) within Convex Hull$\{M_1, M_2, \ldots, M_{N!}\}$ such that $\lambda_{ij} < \mu_{ij}$ for all (i, j). By the definition of a convex hull, we can find probabilities $\{p_1, p_2, \ldots, p_{N!}\}$ such that:

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

This naturally leads to the scheduling policy of randomly choosing a control matrix $(S_{ij}(t))$ from among the set of all permutation matrices, such that permutation M_k is independently chosen with probability p_k every timeslot. From (3) it follows that every timeslot a server connection is established for input queue (i, j) with probability μ_{ij}. This effectively creates a geometric "service time" for each packet, where each queue (i, j) has arrival rate λ_{ij} and service rate μ_{ij}. Because $\lambda_{ij} < \mu_{ij}$, each queue is stable. Note that this stabilizing policy chooses permutation matrices independent of the current queue backlog. Below we calculate the resulting average packet delay under this algorithm for a simple example of Poisson inputs.

Example: Consider any rate matrix (μ_{ij}) satisfying all inequalities (1) and (2) with equality, and suppose packet arrivals are Poisson with rates $\lambda_{ij} = \rho\mu_{ij}$ for some loading value $\rho < 1$. (Note that this includes the case of uniform traffic where $\mu_{ij} = 1/N$ and $\lambda_{ij} = \rho/N$ for all (i, j).) Each input queue (i, j) is then equivalent to a slotted M/G/1 queue with geometric service time and loading ρ. The exact average delay $\overline{W}_{ij}$ of such a queue can be easily calculated (see [24]):

$$\overline{W}_{ij} = \frac{1/\mu_{ij} - 1/2}{1-\rho} + 1$$

$$\overline{W}_{\text{randomized}} = \frac{1}{\lambda_{tot}} \sum_{ij} \lambda_{ij} \overline{W}_{ij} = \frac{N - 1/2}{1-\rho} + 1 \quad (4)$$

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 that services each queue (i, j) a fraction of time μ_{ij}, as in [2] [6]. Indeed, for uniform traffic, consider the periodic schedule that on timeslot t connects each input port $i \in \{1, \ldots, N\}$ to output port $[(i+t) \mod N]+1$. This scheduling algorithm 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 [24]:

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

The above delay indeed is reduced from the delay of the random control algorithm, although 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 an arriving packet to see a server. In the next section we elaborate on this intuition to show that $O(N)$ delay is incurred by any scheduling algorithm that operates independently of the input streams and current levels of queue backlog. We note that the example algorithms described above are similar to the general random and pseudo-random algorithms described in [6] [2] [5] [3], all of which do not consider queue backlog. Thus, while these algorithms have many desirable properties, the result of the next section implies that they cannot improve upon $O(N)$ delay.[2]

Output Queueing: It is useful to compare an $N \times N$ packet switch to a corresponding output queued system with the same inputs. An output queued system is equivalent to a switch with an "infinite speedup," where all input queues are bypassed and packets are immediately transfered to their appropriate destination queues upon arrival, where they are served with unit service time. It is not difficult to show that, under any scheduling algorithm, the average delay in an $N \times N$ switch is greater than or equal to the corresponding delay in an output queued system. To compare with the above example of Poisson inputs, suppose inputs are uniform so that $\lambda_{ij} = \rho/N$ for all (i, j). Then the average delay in the output queued system is the same as that of a slotted M/D/1 queue with loading ρ, and is given by:

$$\overline{W}_{\text{output queue}} = \frac{1}{2(1-\rho)} + 1 \quad (6)$$

The above delay is significantly smaller than the corresponding delay for both the randomized and periodic switch scheduling algorithms in equations (4) and (5), and demonstrates $O(1)$ delay independent of the size of the switch. Such performance improvement is due to the statistical multiplexing gains achieved by the output queue configuration. This gap between $O(1)$ delay and $O(N)$ delay suggests that dramatic improvements are possible through queue length-aware scheduling, and motivates our search for sublinear delay algorithms.

[2]We note that the Maximum Weight Matching algorithm (which does not require a-priori rate information) was shown in [9] to also offer an average delay of no more than $O(N)$, which is tight in the case when input streams are perfectly correlated. However, as the Maximum Weight Match policy is queue length-dependent, the actual delay performance for independent data streams could be sublinear (as suggested by simulations) and a tight delay bound for MWM scheduling in this scenario remains an open question.

III. An O(N) Delay Bound for Backlog-Independent Scheduling

Consider an $N \times N$ packet switch with general stochastic inputs arriving to each of the N^2 input queues. All inputs are assumed to be stationary and ergodic. Assume the system is initially empty and let $X_{ij}(t)$ represent the arrivals to input queue (i, j) during the interval $[0, t]$ (i.e., $X_{ij}(t) = \sum_{\tau=0}^{t} A_{ij}(\tau)$). Recall that $L_{ij}(t)$ represents the current number of packets queued at input (i, j), and $S_{ij}(t)$ represents the server control decision at slot t (where $(S_{ij}(t))$ is a permutation matrix). Here we show that if the control decisions $(S_{ij}(t))$ are stationary and independent of the arrival streams, and if the rate matrix has a sufficiently large set of positive rate entries, then average delay in the switch is necessarily $O(N)$. Because backlog is directly related to the arrival streams, it follows that stationary switching schemes that operate independently of queue backlog incur at least $O(N)$ delay.

Specifically, we assume that for all inputs (i, j) and all slots $\tau \in \{0, 1, 2, \ldots\}$ the scheduling decisions satisfy:

$$\mathbb{E}\{S_{ij}(\tau) \mid \boldsymbol{X}\} = \mathbb{E}\{S_{ij}(0)\} \triangleq \mu_{ij} \quad (7)$$

where $\boldsymbol{X}$ represents the arrival processes $(X_{ij}(t))$ over the entire infinite timeline $t \in \{0, 1, 2, \ldots\}$. The condition (7) allows for a slightly larger class of scheduling processes $(S_{ij}(t))$ than the class of stationary processes.

As a caveat, we note that periodic scheduling schemes such as round-robin are by definition not stationary. Furthermore, the deterministic schedules developed to achieve guaranteed rate services in [2] are both non-stationary and potentially non-periodic. These schemes are designed to deterministically achieve a given target scheduling rate matrix (μ_{ij}), in the sense that over any time interval of size P, the empirical average scheduling rate for each port (i, j) satisfies:

$$\mu_{ij} - \phi(P) \leq \frac{1}{P} \sum_{\tau=0}^{P-1} S_{ij}(t+\tau) \leq \mu_{ij} + \phi(P)$$

where $\phi(P)$ is a function such that $\lim_{P\to\infty} \phi(P) = 0$.

However, for such a process $(S_{ij}(t))$, we can define a new process $(\tilde{S}_{ij}(t))$ *with a randomized phase* as follows: Fix an integer P^*, and choose a random integer Z independently and uniformly over the set $\{0, 1, \ldots, P^* - 1\}$. Define $(\tilde{S}_{ij}(t)) \triangleq (S_{ij}(t + Z))$. It follows from the above inequality that for any input (i, j) and any timeslot τ, we have $\mathbb{E}\left\{\tilde{S}_{ij}(\tau) \mid \boldsymbol{X}\right\} = \frac{1}{P^*}\sum_{v=0}^{P^*-1} \mathbb{E}\left\{S_{ij}(t+v) \mid \boldsymbol{X}\right\}$, and hence:

$$\mu_{ij} - \phi(P^*) \leq \mathbb{E}\left\{\tilde{S}_{ij}(\tau) \mid \boldsymbol{X}\right\} \leq \mu_{ij} + \phi(P^*)$$

If the original scheduling process $(S_{ij}(t))$ is periodic with period P^*, then the random phase process $(\tilde{S}_{ij}(t))$ will satisfy (7) exactly. Otherwise, for any value γ such that $0.5 < \gamma < 1$, a sufficiently large value of P^* can be chosen to ensure that for each (i, j) with $\mu_{ij} > 0$ and for all τ, we have:

$$\mathbb{E}\left\{\tilde{S}_{ij}(\tau) \mid \boldsymbol{X}\right\} \leq \mu_{ij}/\gamma \tag{8}$$

which can also be used to prove our $O(N)$ result [see eqs. (30)(29) in Appendix A]. However, if the arrival process is stationary and independent of the server process, then such phase randomization does not change the average delay. Hence, the following $O(N)$ delay result also holds for any scheduling algorithm that is independent of backlog and that can be made to satisfy either (7) or (8) by phase randomization.

The following lemma is useful for obtaining lower bounds on delay. The proof uses a technique similar to that used in [25] to show that fixed length packets minimize delay over all packet length distributions with the same mean, and is given in Appendix A.

Lemma 1: For a switch with general arrival processes, any scheduling algorithm that satisfies (7) yields a time average queue occupancy $\overline{L}_{ij}$ for each queue (i, j) satisfying:

$$\overline{L}_{ij} \geq \overline{U}_{ij}$$

where U_{ij} represents the *unfinished work* (or "fractional packets") in a system with the same inputs but with constant server rates of μ_{ij} packets/slot, for at least one existing set of rates (μ_{ij}) such that $\sum_j \mu_{ij} \leq 1$ for all i, and $\sum_i \mu_{ij} \leq 1$ for all j.

Proof: The proof is given in Appendix A. Intuitively, the result holds because the congestion in a queue with a time varying server is greater than or equal to the corresponding congestion in a queue with a constant server with service rate equal to the time average rate of the original process. □

The lemma above produces a lower bound on delay in terms of a system of queues with the same inputs but with constant server rates, and leads to the following theorem.

Theorem 1: If inputs X_{ij} are Poisson with uniform rates $\lambda_{ij} = \rho/N$ (for $\rho < 1$ representing the loading on each input), then any scheduling algorithm that satisfies (7) incurs an average delay of at least $\frac{N}{2(1-\rho)}$.

Proof: The unfinished work in an M/D/1 queue with arrival rate λ_{ij} and service time $1/\mu_{ij}$ is equal to $\overline{U}_{ij}(\mu_{ij}) = \frac{\lambda_{ij}}{2(\mu_{ij}-\lambda_{ij})}$, which can be computed by adding $\rho_{ij}/2$, the average portion of a packet remaining in the server, to the expresion for the average number of packets in the buffer of an M/D/1 queue [24]. From Lemma 1, there exists a rate matrix (μ_{ij}) with row and column sums bounded by 1, so that $\overline{L}_{ij} \geq \overline{U}_{ij}$ for all (i, j). Define Λ as the set of all rate matrices (μ_{ij}) satisfying $\sum_i \mu_{ij} \leq 1$, $\sum_j \mu_{ij} \leq 1$. Using Little's Theorem, and the fact that $\sum_{ij} \lambda_{ij} = \rho N$, we have the following average delay $\overline{W}$:

$$\overline{W} = \frac{1}{\rho N} \sum_{ij} \overline{L}_{ij} \geq \inf_{(\mu_{ij})\in\Lambda} \left\{ \frac{1}{\rho N} \sum_{ij} \overline{U}_{ij}(\mu_{ij}) \right\}$$

However, because the $\overline{U}_{ij}(\mu_{ij})$ functions are identical and convex, the expression inside the infimum is a convex symmetric function and attains its minimum at $\mu_{ij} = 1/N$ for all (i, j), and the result follows. □

Note that this lower bound differs by one timeslot from the delay expression in (5) for the periodic scheduling algorithm given in Section II. Because of the $1/(1-\rho)$ factor, delay in the $N \times N$ packet switch with Poisson inputs necessarily grows to infinity as the loading ρ approaches 1. For any fixed loading value, delay grows linearly in the size of the switch. This $O(N)$ result holds more generally. Indeed, consider general stationary and ergodic arrival streams X_{ij} with data rates λ_{ij}, and define the average rate into input ports of the switch to be $\lambda_{av} \triangleq \frac{1}{N}\sum_{ij} \lambda_{ij}$. (Note that in the uniform loading case, $\lambda_{av} = \rho$, and $\lambda_{ij} = \rho/N$.) We assume that there are at least $O(N^2)$ entries of the rate matrix which have rates greater than or equal to $O(\lambda_{av}/N)$.

Theorem 2: For general stationary and ergodic inputs with data rates λ_{ij}, if $O(N^2)$ of the rates are greater than $O(\lambda_{av}/N)$, then average delay under any algorithm that satisfies (7) is at least $O(N)$.

Proof: As in the proof of Theorem 1, we have from

Lemma 1:

$$\overline{W} \geq \inf_{(\mu_{ij})\in\Lambda} \left\{ \frac{1}{\lambda_{av}N} \sum_{ij} \overline{U}_{ij}(\mu_{ij}) \right\} \quad (9)$$

where $\overline{U}_{ij}(\mu_{ij})$ is the time average unfinished work in a queue with a constant service rate μ_{ij}. This unfinished work is at least as much as the average unfinished work in the *server* of queue (i, j), which by Little's Theorem is equal to $\lambda_{ij}/(2\mu_{ij})$. Furthermore, the infimum in (9) is greater than or equal to the (less restricted) infimum taken over all μ_{ij} such that $\sum_{ij} \mu_{ij} \leq N$. By a simple Lagrange Multiplier argument, it can be shown that the infimum of $\sum_{ij} \lambda_{ij}/(2\mu_{ij})$ over this larger set of rates is achieved when $\mu_{ij} = N\sqrt{\lambda_{ij}} \Big/ \sum_{ij} \sqrt{\lambda_{ij}}$. It follows that $Delay \geq \frac{(\sum_{ij}\sqrt{\lambda_{ij}})^2}{2\lambda_{av}N^2}$. Because $O(N^2)$ of the rates are greater than $O(\lambda_{av}/N)$, the numerator is greater than or equal to $\left(O(N^2)\sqrt{\lambda_{av}/N}\right)^2$, and the result follows. □

A simple counter-example shows that delay can be $O(1)$ if the rate matrix does not have a sufficient number of entries with large enough rate: Consider a rate matrix equal to the identity matrix multiplied by the scalar $\lambda < 1$. Then, the switch can be configured to always transfer input 1 to output 1, input 2 to output 2, etc., and average delay is the same as the $O(1)$ delay of an output queue.

Similar results can be obtained for $N \times N$ packet switches with a *speedup of* R, where R is an integer greater than or equal to 1. That is, the input rate matrix (λ_{ij}) is assumed to still satisfy the original constraints (1) and (2), but the internal crossbar switching can be done at a rate that is R times faster. In this situation, the statement of Theorems 1 and 2 can be repeated to prove that delay in the input queues is greater than or equal to the delay in a system with constant input rates μ_{ij} such that the sum of any row or column of the the (μ_{ij}) rate matrix is less than or equal to R. Using reasoning similar to the arguments in Theorems 1 and 2, it follows that average delay is at least $O(N/R)$ for a switch with speedup R that makes scheduling decisions independent of queue backlog. Thus, constant speedups (typically on the order of $2, 4$, or 8), cannot change the $O(N)$ characteristic for backlog-independent scheduling.

IV. An $O(\log(N))$ Delay Bound for Backlog-Aware Scheduling

Here we show that $O(\log(N))$ delay is possible by using a backlog-aware scheduling strategy. This result for the first time establishes that sublinear delay is possible in an $N \times N$ packet switch without speedup. The algorithm is similar to the *frame based* schemes considered in [14] [17], and is based on the principle of iteratively clearing backlog in minimum time. Minimum clearance time policies have recently been applied to stabilize *networks* in [26], [16]. We begin by outlining several known results about clearing backlog from a switch in minimum time.

A. Minimum Clearance Time and Maximum Matchings

Consider a single batch of packets present in the switch at time zero. We represent the initial backlog as an *occupancy matrix* (L_{ij}), where entry L_{ij} represents the number of packets at input port i destined for output port j. Suppose that no new packets enter, and the goal is simply to clear all packets in minimum time by switching according to permutation matrices. The following fundamental result from combinatorial mathematics provides the solution to this problem [27]:

Fact 2. *Let T^* represent the minimum time required to clear backlog associated with occupancy matrix (L_{ij}). Then T^* is exactly given by the maximum sum over any row or column of the matrix (L_{ij}).*

It is clear that the minimum time to clear all backlog can be no smaller than the total number of packets in any row or column, because the corresponding input or output can only serve 1 packet at a time. This minimum time can be achieved by an algorithm similar to the Birkhoff-Von Nuemann algorithm described in [2]. Indeed, The matrix is first augmented with *null packets* so that every row and column has line sum T^*. Using Hall's Theorem [27], it can be shown that the augmented backlog matrix can be cleared by a sequence of T^* *perfect matches* of size N.

Such matchings can be found sequentially using any Maximum Size Matching algorithm, where each match requires at most $O(N^{2.5})$ operations (see [2] [28] [29] [14]). Note that the preliminary matrix augmentation procedure can be accomplished with $O(N)$ computations each timeslot by updating a set of vectors *row_sum* and *column_sum* each timeslot, and then augmenting the matrix at the beginning of each frame by using these row and column sum vectors to sequentially update each row in the next column which does not have a full sum.

It is useful to also consider scheduling according to *maximal matches*, which are matches where no new edges can be added without sharing a node with an already matched edge. Maximal matchings can be found with $O(N^2)$ operations and the computation is easily parallelizable to $O(N)$ complexity [14]. Given a backlog matrix (L_{ij}) with minimum clearance time T^*, the following well known result establishes an upper bound on the time required for backlog to be cleared by maximal matches.

Fact 3. *If the minimum clearance time of a backlog matrix is T^*, then arbitrary maximal matchings will clear all backlog in time less than or equal to $2T^* - 1$.*

A one-sentence proof of this result is given in [14].

B. Fair-Frame Scheduling for Logarithmic Delay

We now present a frame based scheduling algorithm that iteratively clears backlog associated with successive batches of packets. Packets which are not cleared during a frame are marked and handled separately in future frames. The algorithm is "fair" in that when the empirical input rates averaged over a frame are outside of the capacity region of the switch, decisions about which packets to serve are made fairly. We show that if inputs are Poisson and rates are strictly within the capacity region, the switch is stable and yields $O(\log(N))$ delay.

The Fair-Frame Scheduling Algorithm: Timeslots are grouped into frames of size T slots.

1) On the first frame, switching matrices $(S_{ij}(t))$ are chosen randomly so that the probability of serving any particular queue is uniformly $1/N$.
2) On the $(k+1)^{th}$ frame, the matrix $(L_{ij}(kT))$ consisting of packets that arrived during the previous frame is obtained and the maximum row and column sum T^* is computed.
3) If $T^* \leq T$, the matrix is augmented with *null packets* so that all row and column sums are equal to T^*. Else, if $T^* > T$, then some row or column sum of $(L_{ij}(kT))$ exceeds T, and an *overflow* occurs. In the case of an overflow, a subset of the packets are retained to form a new matrix $(\tilde{L}_{ij}(kT))$ that does not violate the row and column constraints. These packets will be scheduled on the next frame, and the remaining packets are marked as *overflow packets*. Choice of which packets to mark is based upon some type of utility function, such as the maximum throughput utility or max-min fair utility described in [2]. We assume only that the resulting $(\tilde{L}_{ij}(kT))$ matrix has the *maximal property*, in that adding back any individual overflow packet would cause a row or column constraint to be violated. The $(\tilde{L}_{ij}(kT))$ matrix is then augmented with null packets so that all rows and columns sum to T.
4) All non-overflow packets are scheduled during frame $(k+1)$ by performing maximum matches every timeslot to strip off permutations from the augmented backlog matrix.
5) If all packets of the augmented backlog matrix are cleared in less than T slots, uniform and random scheduling is performed on the remaining slots to serve the overflow packets remaining in the system from previous overflow frames. Note that the probability of serving any queue (i,j) during such a slot is $1/N$.
6) Repeat from step 2.

Note that the manner in which conforming packets are cleared every frame (using maximum size matches) is not unique, and different sequences of switching decisions will lead to different average waiting times for packets within the frame. The problem of finding the optimal sequence of switching decisions to minimize this average time is considered in [18], where it is shown the optimal algorithm also clears the initial batch of packets in minimum time. The optimal algorithm of [18] has geometric complexity, although lower complexity heuristics are also presented there. The problem of optimally adding "best effort" packets into an existing load requirement is considered in [19] (where NP completeness results are derived), and the algorithms developed there could potentially be used to schedule overflow packets more efficiently, perhaps yielding constant factor delay improvements. For our purposes, we analyze delay under the assumption that *any* minimum clearance time schedule is used on each frame, and hence our $O(\log(N))$ results do not rely on optimizing average waiting times within a frame.

Let $X_{ij}(t)$ represent the number of packets that arrive to port (i,j) during the first t timeslots. If any packet arriving during a frame k is *not* cleared within the next frame, at least one of the following inequalities must have been violated:

$$\sum_j [X_{ij}((k+1)T) - X_{ij}(kT)] \leq T \text{ for all } i \quad (10)$$

$$\sum_i [X_{ij}((k+1)T) - X_{ij}(kT)] \leq T \text{ for all } j \quad (11)$$

Traffic that satisfies the above inequalities during a frame is said to be *conforming* traffic. Packets remaining in the switch because of a violation of these inequalities are defined as *non-conforming* packets and are served on a best effort basis in future frames. Note that, by definition, the Fair-Frame algorithm clears all conforming traffic within $2T$ timeslots.

Here we describe the performance of the Fair-Frame algorithm with random inputs. Suppose inputs are Poisson with rates λ_{ij} satisfying:

$$\sum_j \lambda_{ij} \leq \rho \text{ for all } i \ , \ \sum_i \lambda_{ij} \leq \rho \text{ for all } j \quad (12)$$

where ρ represents the maximum loading on any input port or output port. Note that if the sum rate to any input or output exceeds the value 1, the switch is necessarily unstable. In the following, we show that if $\rho < 1$, the Fair-Frame algorithm can be designed to ensure stability with delay that grows logarithmically in the size of the switch. We start by presenting a lemma that guarantees the overflow probability decreases exponentially in the frame length T.

Lemma 2: For a given (arbitrarily small) overflow probability δ, choose an integer frame size T such that:

$$T \geq \frac{\log(2N/\delta)}{\log(1/\gamma)} \tag{13}$$

where $\gamma \triangleq \rho e^{1-\rho}$. Then a switch operating under the Fair-Frame algorithm with a frame size T ensures the probability of a frame overflow is less than δ. All conforming packets have a delay less than $2T$. Further, if $T\sum_{ij}\lambda_{ij} \geq 1$, then the long term fraction of packets that are non-conforming is less than or equal to 2δ.

Proof: Packets are lost during frame k only if one of the $2N$ inequalities of (10), (11) is violated during the previous frame. Let $X(T)$ represent the number of packets arriving from a Poisson stream of rate ρ during an interval of T timeslots. Then any individual inequality of (10) and (11) is violated with probability less than or equal to $Pr[X(T) > T]$. By the Chernov bound, we have for any $r > 0$:

$$\begin{aligned} Pr[X(T) > T] &\leq \mathbb{E}\left\{e^{rX(T)}\right\} e^{-rT} \\ &= \exp(\rho T(e^r - 1) - rT) \end{aligned} \tag{14}$$

where the identity $\mathbb{E}\left\{e^{rX(T)}\right\} = \exp(\rho T(e^r - 1))$ was used for the Poisson variable $X(T)$.

To form the tightest bound, define the exponent in (14) as the function $g(r) = \rho T(e^r - 1) - rT$. Taking derivatives reveals that the optimal exponent for the Chernov bound is achieved when $e^r = 1/\rho$. Using this in (14), we have:

$$Pr[X(T) > T] \leq \left[\rho e^{1-\rho}\right]^T$$

We define $\gamma \triangleq \rho e^{1-\rho}$. The γ parameter is an increasing function of ρ for $0 \leq \rho \leq 1$, being strictly less than 1 whenever $\rho < 1$. By the union bound, the probability that any one of the $2N$ inequalities in (10) and (11) is violated is less than or equal to $2N\gamma^T$. Hence, if we ensure that:

$$2N\gamma^T \leq \delta \tag{15}$$

then each frame successfully delivers all of its packets with probability greater than $1-\delta$. Taking the logarithm of both sides of (15), we obtain the requirement:

$$T \geq \frac{\log(2N/\delta)}{\log(1/\gamma)}$$

Now let $\theta \triangleq T\sum_{ij}\lambda_{ij}$ represent the expected number of packet arrivals during a frame. It can be shown that for Poisson arrivals, the extra amount of packets that arrive given that the number of arrivals is greater than some value T is stochastically less than the original Poisson variable plus 1 (see Appendix B). It follows that the expected number of extra arrivals to a frame in which one of the inequalities (10), (11) is violated is less than or equal to $1 + \theta$. Thus, the ratio of non-conforming packets to total packet arrivals is no more that $\delta(1+\theta)/\theta$. Assuming $\theta \geq 1$, it follows that this ratio is less or equal to 2δ. □

We note that the *same* bound in (13) applies if packets arrive according to Bernoulli processes rather than Poisson processes, as the moment generating function $\mathbb{E}\left\{e^{rX(T)}\right\}$ for a Poisson variable dominates that of a sum of i.i.d. Bernoulli variables with the same rate. The $\log(2N)$ delay bound in (13) arises because of the $2N$ constraints describing the switch capacity region. In a switch with Bernoulli traffic rather than Poisson traffic, no more than one packet can enter any input port. In this case, the constraints in (10) are necessarily satisfied and can be removed from the union bound expression in (15), which reduces the delay bound. A similar argument can be used to prove that logarithmic delay is achievable in any single-hop switching network with a capacity region described by a polynomial number of constraints, as the logarithm of N^k remains $O(\log(N))$.

It is useful to understand how the frame size grows for a fixed overflow probability δ as the loading ρ approaches 1. The formula for the frame size T contains a $\log(1/\gamma)$ term in the denominator. Using the definition of γ and taking a Taylor series expansion about $\rho = 1$ shows that $\log(1/\gamma) = \frac{(1-\rho)^2}{2} + O(1-\rho)^3$. Thus, the denominator is $O((1-\rho)^2)$. This suggests that the cost of achieving $O(\log(N))$ delay is to have a delay which is more sensitive to the loading parameter ρ (confer with eqs. (4)- (6)).

We note that the Poisson assumption is not essential to the proof—a similar proof can be constructed for any independent input streams X_{ij} such that $Pr[\sum_j X_{ij}(T) > T]$ and $Pr[\sum_i X_{ij}(T) > T]$ decreases geometrically with T. It is necessary that the streams be independent for this property to hold. Indeed, consider a situation where all inputs experience the *same* processes, so that $X_{ij}(t) = X(t)$ for all (i, j). Whenever a packet arrives to input 1 destined for output 1, all other inputs receive a packet destined for output 1, and the minimum average delay is $O(N/2)$.

To provide a true delay bound, the delay of non-conforming packets must be accounted for, as accomplished in the theorem below.

Theorem 3: For Poisson inputs strictly interior to the capacity region with loading no more than ρ, a frame size $T \geq 2$ can be selected so that the Fair-Frame algorithm ensures logarithmic average delay.

Proof: It suffices to consider only the case when $T\sum_{ij}\lambda_{ij} \geq 1$.[3] For an overflow probability δ (to be chosen later), we choose the frame size $T = \left\lceil \frac{\log(2N/\delta)}{\log(1/\gamma)} \right\rceil$ so that overflows occur with probability less than or equal to δ. The backlog associated with non-conforming packets for any queue (i, j) can be viewed as entering a *virtual GI/GI/1 queue* with random service opportunities every frame. Let q represent the probability of frame 'underflow': the probability that there is at least one random service opportunity for non-conforming packets during a frame. This is the probability that all backlog of the previous frame can be cleared in less than T slots. Using a Chernov bound argument similar to the one given in the proof of Lemma 2, it can be shown that $Pr[X(T) > T - 1] \leq \frac{1}{\rho}\gamma^T$, and hence:

$$q \geq 1 - \frac{2N}{\rho}\gamma^T \tag{16}$$

Expressed in terms of δ, this means that:

$$\begin{aligned} q &\geq 1 - \frac{2N}{\rho}\gamma^{\frac{\log(2N/\delta)}{\log(1/\gamma)}} \\ &= 1 - \frac{2N}{\rho}\gamma^{\log_\gamma(\delta/(2N))} \\ &= 1 - \delta/\rho \end{aligned} \tag{17}$$

Given a non-conforming service opportunity, any particular queue is served with probability $1/N$. The average delay for non-conforming packets in queue (i, j) is thus less than or equal to T (the size of the frame in which they arrived) plus the average delay associated with a slotted GI/GI/1 queue where a service opportunity arises with probability q/N. Every slot, with probability $1 - \delta$ no new packets arrive to this virtual queue (as all packets are conforming), and with probability δ there are $1+X$ packets that arrive, where X is a Poisson variable with mean ρT (where we again use the Appendix B result that excess arrivals are stochastically less than the original). Note that this is a *very large overbound*, as *all* overflow packets arriving to an input i are treated as if they arrived to queue (i, j). Conforming packets consist of at least a fraction $1 - 2\delta$ of the total data and have a delay bounded by $2T$. Thus, the resulting average delay satisfies:

$$\begin{aligned} \overline{W} &\leq 2T(1-2\delta) + 2\delta(T + TDelay(GI/GI/1)) \\ &\leq 2T + 2\delta T Delay(GI/GI/1) \end{aligned} \tag{18}$$

where $Delay(GI/GI/1)$ represents the average delay of non-conforming packets in the virtual GI/GI/1 queue (normalized to units of frames).

The average delay of a slotted GI/GI/1 queue with Bernoulli service opportunities can be solved exactly. However, we simplify the exact expression by providing the following upper bound, which is easily calculated using standard queueing theoretic techniques:

$$Delay(GI/GI/1) \leq \frac{1 + \mathbb{E}\left\{A^2\right\}/\lambda}{2(\mu - \lambda)} \quad \text{(for } \mu > \lambda\text{)} \tag{19}$$

where, in this context, we have:

$$\lambda = \delta(1 + \rho T) \tag{20}$$

$$\mu = q/N \tag{21}$$

$$\mathbb{E}\left\{A^2\right\} = \delta\mathbb{E}\left\{(1+X)^2\right\} = \delta\left[1 + 3\rho T + \rho^2 T^2\right] \tag{22}$$

The virtual queue is stable provided that $\mu > \lambda$. This is ensured whenever the parameter δ is suitably small. Indeed, we have:

$$\begin{aligned} \mu - \lambda &= \frac{q}{N} - \delta(1 + \rho T) \\ &\geq \frac{1}{N} - \frac{\delta}{\rho N} - \delta(1 + \rho T) \qquad (23) \\ &= \frac{1}{N}\left[1 - \delta\left(\frac{1}{\rho} + N + N\rho T\right)\right] \qquad (24) \end{aligned}$$

where inequality (23) follows from (17). Hence, we have $\mu > \lambda$ whenever the following condition is satisfied:

$$\delta\left(\frac{1}{\rho} + N + N\rho T\right) < 1 \tag{25}$$

Choose $\delta = O(1/N^2)$ and note that $T = \lceil\frac{\log(2N/\delta)}{\log(1/\gamma)}\rceil = O(\log(N^3)) = O(\log(N))$. It follows that the left hand side of (25) can be made arbitrarily small for suitably small δ. In particular, we can find a value δ such that $\delta\left(\frac{1}{\rho} + N + N\rho T\right) \leq 1/2$, so that (24) implies $(\mu - \lambda) \geq 1/(2N)$. In this case, we have from (18) and

[3] If $T\sum_{ij}\lambda_{ij} < 1$, then it can easily be shown that average delay is less than or equal to that of a slotted $M/D/1$ queue with unit service times and loading $\rho = 1/T \leq 1/2$.

(19) that:

$$\begin{aligned}\overline{W} &\leq 2T + 2\delta T \frac{1+\mathbb{E}\left\{A^2\right\}/\lambda}{2(\mu-\lambda)} \qquad (26)\\ &\leq 2T + 2\delta T N\left(1+\frac{1+3\rho T+\rho^2T^2}{1+\rho T}\right)\\ &= 2T + 2\delta T N\left(1+\frac{(1+\rho T)^2+\rho T}{1+\rho T}\right)\\ &\leq 2T + 2\delta T N\left(2+2\rho T\right)\end{aligned}$$

Because $\delta = O(1/N^2)$ and $T = O(\log(N))$, it follows that the resulting average delay is $O(T)$, that is, $Delay \leq O(\log(N))$. □

An explicit delay bound can be obtained for a given loading value ρ as follows: Again define $\gamma \triangleq \rho e^{1-\rho}$, and define the frame size as a function of δ: $T_\delta \triangleq \lceil \frac{\log(2N/\delta)}{\log(1/\gamma)} \rceil$. Using the definitions for λ, μ, and $\mathbb{E}\left\{A^2\right\}$ given in (20)-(24), the average delay bound of (26) can be expressed as a pure function of the parameter δ (as well as the parameter ρ). This bound can be minimized as a function of δ, subject to the constraint that $\delta\left(\frac{1}{\rho}+N+N\rho T_\delta\right) < 1$. The resulting value δ_{min} defines a suitable frame size $T_{\delta_{min}}$ and gives the tightest bound achievable from the above analysis.

In Fig. 2 we plot the resulting delay bound as a function of N for the fixed loading value $\rho = 0.7$. The delay bound for the Fair-Frame algorithm follows a logarithmic profile exactly (the plot is linear when a logarithmic scale is used for the horizontal axis). The bound is plotted next to the exact average delay expressed in (4) for the queue length-independent randomized algorithm.[4] Recall that both curves correspond to any rate matrix (λ_{ij}) that satisfies (12). Note the rapid growth in delay as a function of the switch size for the randomized algorithm, as compared to the relatively slow growth of the Fair-Frame bound. From the plot, the curves cross when the switch size is approximately 200. However, the Fair-Frame curve represents only a simple upper bound. Tighter delay analysis would likely reveal that the Fair-Frame algorithm is preferable even for much smaller switch sizes (see also the simulations in Section VI).

We note that although only average delay is compared, the Fair-Frame algorithm has the property that all conforming packets have a worst case delay that is less than or equal to $2T$ (where T is logarithmic in N), and the fraction of conforming packets is at least $1 - O(1/N^2)$. That is, worst case delay is logarithmic for all but a negligible fraction of all packets served.

[4]The delay expression (4) for the randomized algorithm is almost identical to the bound obtained for the MWM algorithm in [9], and hence the plot in Fig. 2 can also be viewed as a comparison between the MWM bound and the Fair-Frame bound.

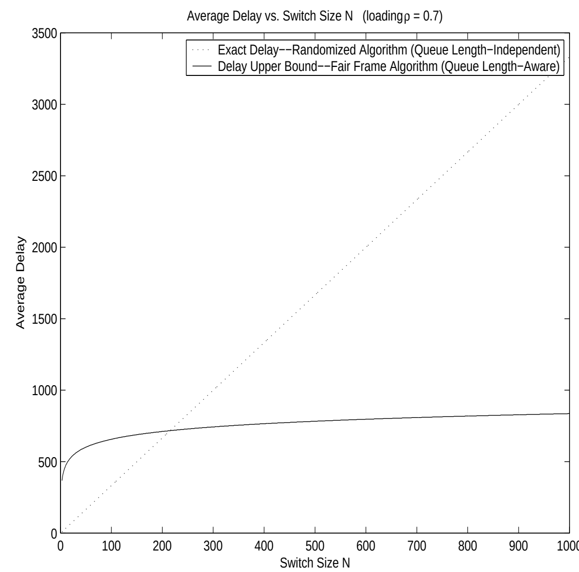

Fig. 2. The logarithmic delay bound for the Fair-Frame algorithm as a function of the switch size N, as compared to the $O(N)$ delay of the randomized algorithm.

C. *Robustness to Changing Input Rates*

Note that the Fair-Frame algorithm requires a loading bound ρ on each input but otherwise does not require knowledge of the exact input rates. For this reason, it can be shown that the Fair-Frame algorithm is *robust to time varying input rates*. Indeed, it is not difficult to show that the Chernov bound of (14) applies even when rates are *arbitrarily changing every timeslot*, provided that on each timeslot the new rates always satisfy the constraints in (12). In the case when input rates are outside of the capacity region, it is not possible to stabilize the switch. However, the utility metric of the Fair-Frame algorithm can be used to make fair scheduling decisions in this situation.

V. Implementation Complexity

Here we elaborate on Steps 2-4 of Fair-Frame.

A. *Steps 2 and 3*

In these steps of the Fair-Frame algorithm, the backlog matrix of the previous frame is modified by adding null packets and/or by removing some packets that are non-conforming. The complexity of this procedure depends on the fairness criterion used for marking overflow packets. The simplest procedure is the "First-Come-First-Served Fairness Rule" (FCFS), where vectors *row_sum* and *col_sum* are updated every timeslot, and any set of packets arriving to a particular input port are marked as overflow packets if they cause the corresponding *row_sum* or *col_sum* entries to exceed the frame size T.

It is not difficult to implement such a scheme with $O(N)$ operations per timeslot (requiring only $O(1)$ operations per port). The addition of null packets is also very simple and can be done in a greedy manner. The complexity bottleneck occurs at Step 4 of the algorithm.

B. Step 4

The Fair-Frame algorithm relies on Maximum Size Matchings every timeslot. Such matchings can be performed using the algorithm in [29] which requires $O(MN^{1/2})$ operations, where M is the number of nonzero entries of the backlog matrix. For backlog matrices with many nonzero entries, M can be as large as N^2. However, the Fair-Frame algorithm by definition performs maximum matchings on a backlog matrix for which the total number of packets at any input is no more than T, where T is $O(\log(N))$. It follows that the number of nonzero entries is less than or equal to NT, i.e., M is $O(N\log(N))$. Thus, the Fair-Frame algorithm achieves logarithmic delay and requires $O(N^{1.5}\log(N))$ total operations every timeslot. This $(delay, complexity)$ operating point lies below the delay-complexity curve established for the class of stable algorithms given in [20]. Indeed, in [20] it is shown that for any parameter choice α such that $0 \leq \alpha \leq 3$, a stable scheduling algorithm can be developed requiring $O(N^{\alpha})$ per-timeslot computation complexity and ensuring $O(N^{4-\alpha})$ average delay. Thus, the Fair-Frame algorithm reduces delay by approximately $O(N^{2.5})$ at the $O(N^{1.5}\log(N))$ complexity level. We conjecture that a new complexity-delay tradeoff curve can be established using the techniques given in [20].

The complexity of *maximal matchings* is also reduced when using the backlog matrices of the Fair-Frame algorithm. Indeed, it can be seen that maximal matchings can be performed using only $O(N\log(N))$ operations. From Fact 3, it is not difficult to show that implementing the Fair-Frame algorithm with these low complexity maximal matches yields stability and logarithmic delay whenever the switch is at most half-loaded (i.e, $\rho < 1/2$), or whenever the switch has a speedup of at least 2.

C. Improvements and FIFO Service

A simple improvement that preserves the analytical delay properties of Fair-Frame is as follows: If the switch is scheduled to serve a null packet from port (i, j) during a slot when there is an actual packet waiting at this port, this actual packet can be served (thus reducing either the total number of packets served in the next frame or the number of buffered overflow packets). Further note that the Fair-Frame algorithm serves conforming packets with higher priority than non-conforming packets. Hence, it is possible for a non-conforming packet that arrived in a previous frame to be served *after* a conforming packet that arrived to the same port at a later time. However, FIFO service can easily be enforced by appropriately exchanging the identities of conforming and non-conforming packets. This would not change the number of conforming or non-conforming packets at any queue at any time, and would not change any sample path of system dynamics. Thus, average occupancy and (by Little's Theorem) average delay would not change.

VI. SIMULATIONS

Here we demonstrate the performance of the Fair-Frame scheduling algorithm under both uniform and non-uniform traffic.

A. Poisson Inputs

We consider independent Poisson inputs with a port loading set to $\rho = 0.7$. For each switch size N, the simulations were run for 1000 frames, with a frame size $T = \lceil \frac{\log(2N/\delta_{min})}{\log(1/\gamma)} \rceil$, where δ_{min} was chosen to minimize the bound in equation (26) under the constraint in equation (25). We first consider uniform input traffic, i.e., $\lambda_{ij} = 0.7/N$ for all $N \times N$ virtual input queues. The results are shown in Fig. 3. The dashed line in Fig. 3 is the theoretical average delay for the randomized matching algorithm of Section II (given in equation (4)), and the concave curve without marks shows the theoretical logarithmic delay bound for the Fair-Frame algorithm derived in Section IV. It is clear from the figure that the simulated average delay under the Fair-Frame scheduling algorithm indeed increases sublinearly with N, and that it sits well inside the logarithmic analytical bound.

The bottom curve in Fig. 3 illustrates the simulated performance of a very simple improvement to the Fair-Frame algorithm that allows for *dynamic frame sizing*. The improvement only differs from the Fair-Frame scheduling algorithm in Step 5: When all backlogs are cleared within T slots and there are no overflow packets, the switch starts another frame immediately (rather than continuing to randomly choose switching configurations for the remainder of the frame). It can be analytically shown that the average delay of this modified algorithm is upper bounded by the same logarithmic curve derived for the Fair-Frame algorithm. However, as seen by the figure, this modification yields significantly less average delay in our experiments, as expected. We note that we have also conducted limited simulations of the original Maximum Weight Match (MWM) algorithm for several

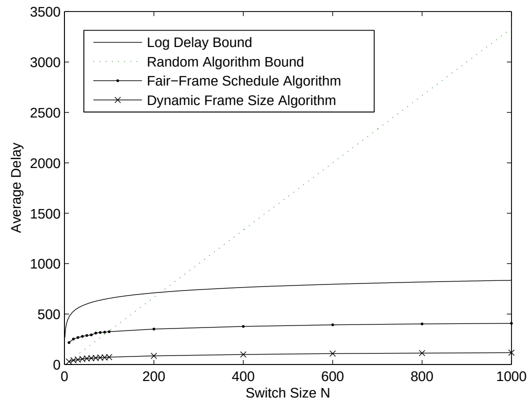

Fig. 3. Simulation results of Fair-Frame scheduling algorithm with uniform input traffic, $\rho = 0.7$.

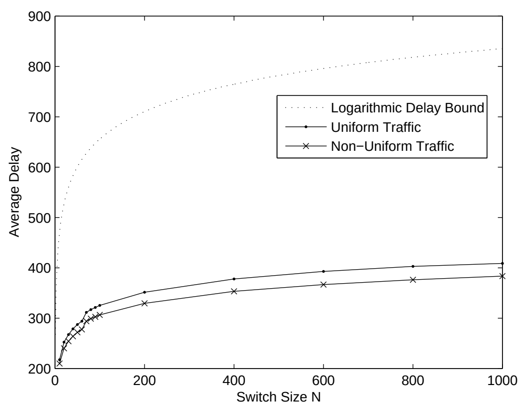

Fig. 4. Simulation results of Fair-Frame scheduling algorithm with non-uniform input traffic, $\rho = 0.7$

small values of N. While there is no logarithmic bound for MWM, we found that MWM produced lower delay than all other algorithms tested. We expect this trend to persist even for large values of N. However, it is difficult to run extensive simulations of MWM for large N due to the $O(N^3)$ complexity associated with computing a Maximum Weight Match, whereas the Fair-Frame algorithm is easier to implement for large N due to the complexity savings discussed in Section V.

Fig. 4 shows the influences of non-uniform input traffic for the Fair-Frame scheduling algorithm. We let $\lambda_{ij} = \frac{\rho}{2N}$ when $i \neq j$ and $\lambda_{ij} = \frac{N+1}{2N}\rho$ when $i = j$. These values satisfy the $2N$ constraints of (12). The figure illustrates that delay in this case is slightly better than the uniform traffic case. This verifies that Fair-Frame scheduling is indeed robust under nonuniform traffic, as proven in the analysis of Section IV.

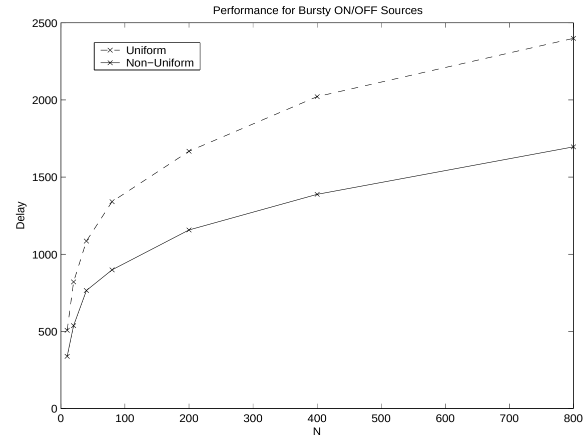

Fig. 5. Delay performance for N^2 independent ON/OFF sources for both uniform and non-uniform traffic. Note that it is possible for N packets to arrive to a single input port (one for each destination).

B. Bursty Traffic

Here we consider the case when all N^2 inputs have independent but non-Poisson traffic. Specifically, we assume the traffic for each input port is described by an independent Markov modulated ON/OFF process. A traffic stream in the ON state during a timeslot produces a single packet on that slot, and produces no packet if in the OFF state. For each (i, j), the probability of transitioning from ON to OFF is ϵ_{ij}, and the probability of transitioning from OFF to ON is δ_{ij}. Throughout, we assume that $\epsilon_{ij} = \epsilon = 0.1$, so that the average duration of an ON period is 10 timeslots, and this average does not change as the network size N is increased. The values δ_{ij} were selected to ensure the desired time average arrival rates λ_{ij} (so that $\delta_{ij} = \epsilon\lambda_{ij}/(1 - \lambda_{ij})$).

As our analysis of the necessary frame size T was conducted for Poisson inputs, the same frame size may not be valid for this bursty scenario. Thus, simulations were conducted using the dynamic frame sizing approach as in the previous example for Poisson inputs, but no bound on the maximum frame size was specified. It was observed that, indeed, this bursty traffic generally produced larger frame lengths than the corresponding Poisson simulations, and overall delay was larger but still sub-linear in N. Fig. 5 illustrates results for both uniform and non-uniform traffic, with $\rho = 0.7$ and with the same λ_{ij} values as in the Poisson examples.

VII. Conclusions

We have considered scheduling in $N \times N$ packet switches with random traffic. It was shown that queue length-independent algorithms, such as those using randomized or periodic schedules designed for known input rates, necessarily incur average delay of at least $O(N)$.

However, a simple queue length-aware algorithm was constructed and shown to provide average delay of $O(\log(N))$. This is the first analytical demonstration that sublinear delay is possible in a packet switch, and proves that high quality packet switching with the crossbar architecture is feasible even for very large switches of size $N > 1000$. The Fair-Frame algorithm provided here is based on well established framing techniques and requires only $O(N^{1.5}\log(N))$ computations every timeslot. Our logarithmic delay analysis can similarly be applied to other single-hop networks with capacity regions that are described by a polynomial number of constraints. An important question for future research is that of developing delay-optimal scheduling. Such scheduling would yield delay that is upper bounded by $O(\log(N))$ and lower bounded by $O(1)$, which now serve as the tightest known bounds on optimal delay.

Appendix A

Here we prove Lemma 1: For a switch with general arrival processes, any stationary scheduling algorithm that operates independently of the input streams yields $\overline{L}_{ij} \geq \overline{U}_{ij}$, where U_{ij} represents the *unfinished work* (or "fractional packets") in a system with the same inputs but with constant server rates of μ_{ij} packets/slot, for some rates μ_{ij} satisfying $\sum_j \mu_{ij} \leq 1$ for all i, and $\sum_i \mu_{ij} \leq 1$ for all j.

Proof: For the course of this proof, it is useful to consider queueing analysis in continuous time, so that $S_{ij}(t)$ is defined for all real number times $t \geq 0$, but is constant on unit intervals (so that $S_{ij}(t) = S_{ij}(\lfloor t \rfloor)$). Consider a queue occupancy process $\tilde{L}_{ij}(t)$ representing the *unfinished work* (or fractional packets) in a queue with the same input and server processes $X_{ij}(t)$ and $S_{ij}(t)$, but operating without the timeslot structure. In this way, if $S_{ij}(t) = 1$ for $t \in [0, 2]$ and a single packet arrives to an empty system at time 0.5, the packet will start service immediately in the new system, but will wait until the start of the next slot in the original (slotted) system. Thus, $\tilde{L}_{ij}(1) = 0.5$ and $\tilde{L}_{ij}(1.5) = 0$, while $L_{ij}(1) = L_{ij}(1.5) = 1$. Because the original system delays service until the next slot and holds packet occupancy $L_{ij}(t)$ at a fixed integer value until a service completion, it is not difficult to show that:

$$L_{ij}(t) \geq \tilde{L}_{ij}(t) \text{ for all } t \tag{27}$$

The continuous time process $\tilde{L}_{ij}(t)$ can be written:

$$\tilde{L}_{ij}(t) = \sup_{\tau \geq 0}\left[(X_{ij}(t) - X_{ij}(t-\tau)) - \int_{t-\tau}^{t} S_{ij}(v)dv\right] \tag{28}$$

The above expression is a well known queueing result that is easily verified: $\tilde{L}_{ij}(t)$ is at least as large as the difference between the number of packet arrivals and service opportunities over any interval, and the bound is met with equality on the interval defined by the starting time of the current busy period.

Taking expectations of the queue occupancy $L_{ij}(t)$ over the stochastic arrival and server processes $X_{ij}(t)$ and $S_{ij}(t)$ and using (27) and (28), we have:

$$\begin{aligned}
\mathbb{E}\{L_{ij}(t)\} &\geq \mathbb{E}\left\{\tilde{L}_{ij}(t)\right\} \\
&= \mathbb{E}_X \mathbb{E}_{S|X}\left\{\sup_{\tau\geq 0}\left[X_{ij}(t) - X_{ij}(t-\tau) - \int_{t-\tau}^{t} S_{ij}(v)dv\right] \middle| X\right\} \\
&\geq \mathbb{E}_X\left\{\sup_{\tau\geq 0}\left[X_{ij}(t) - X_{ij}(t-\tau) - \int_{t-\tau}^{t}\mathbb{E}_{S|X}\{S_{ij}(v) \mid X\}\, dv\right]\right\} \quad (29) \\
&= \mathbb{E}_X\left\{\sup_{\tau\geq 0}[X_{ij}(t) - X_{ij}(t-\tau) - \tau\mu_{ij}]\right\} \quad (30)
\end{aligned}$$

where $\mu_{ij} \triangleq \mathbb{E}\{S_{ij}(0)\}$ is the expected rate of service to queue (i, j). Inequality (29) follows from convexity of the $\sup\{\}$ function together with Jensen's inequality, and (30) follows because $S_{ij}(v) = S_{ij}(\lfloor v \rfloor)$, and the server process $S_{ij}(\lfloor v \rfloor)$ is stationary and independent of the arrival process (so that $\mathbb{E}_{S|X}\{S_{ij}(v)|X\} = \mu_{ij}$). Notice that at any time v, the sum of any row or column of the matrix $(S_{ij}(v))$ is less than or equal to 1. Hence, from its definition, the (μ_{ij}) matrix inherits this same property.

The expression on the right hand side of inequality (30) represents the expected unfinished work $\overline{U}_{ij}$ in a continuous time queue with input $X_{ij}(t)$ and fixed service rate μ_{ij} (compare with (28)), and the proof is complete. □

Appendix B

Here we show that the excess packets from a Poisson stream of rate λ are stochastically less than the original Poisson stream.

Theorem 4: For a Poisson random variable X and for all integers $n, r \geq 0$, we have:

$$Pr[X \geq n + r \mid X \geq r] \leq Pr[X \geq n] \tag{31}$$

We prove this result by means of the following two lemmas.

Lemma 3: Let $a, b, c, d > 0$. If $\frac{a}{b} \geq \frac{c}{d}$, then $\frac{a}{b} \geq \frac{a+c}{b+d} \geq \frac{c}{d}$.

Proof: Omitted for brevity. □

Lemma 4: Let $\{a_k\} > 0, \{b_k\} > 0$, and assume that $\sum_{k=0}^{\infty} a_k < \infty, \sum_{k=0}^{\infty} b_k < \infty$. Suppose that $\frac{a_k}{b_k}$ is *decreasing* in k. Then:

$$\frac{\sum_{k=0}^{\infty} a_k}{\sum_{k=0}^{\infty} b_k} \geq \frac{\sum_{k=r}^{\infty} a_k}{\sum_{k=r}^{\infty} b_k}$$

Proof: Choose an arbitrary positive integer K. Because $\frac{a_k}{b_k}$ is decreasing in k, we have that $\frac{a_K}{b_K} \leq \frac{a_{K-1}}{b_{K-1}}$. By the preceding lemma, we thus know that:

$$\frac{a_K}{b_K} \leq \frac{a_K + a_{K-1}}{b_K + b_{K-1}} \leq \frac{a_{K-1}}{b_{K-1}} \leq \frac{a_{K-2}}{b_{K-2}}$$

where the last inequality follows because we again use the fact that $\frac{a_k}{b_k}$ is decreasing in k. Applying the lemma to the last and third to last inequalities in the chain above, we have:

$$\begin{aligned} \frac{a_K}{b_K} \leq \frac{a_K + a_{K-1}}{b_K + b_{K-1}} \leq \frac{a_K + a_{K-1} + a_{K-2}}{b_K + b_{K-1} + b_{K-2}} &\leq \frac{a_{K-2}}{b_{K-2}} \\ &\leq \frac{a_{K-3}}{b_{K-3}} \end{aligned}$$

Proceeding recursively, it follows that for any K and any $r \leq K$:

$$\frac{\sum_{k=r}^{K} a_k}{\sum_{k=r}^{K} b_k} \leq \frac{\sum_{k=0}^{K} a_k}{\sum_{k=0}^{K} b_k}$$

Taking limits as $K \to \infty$ and using the fact that each of the individual sums converge yields the result. □

We can now prove the theorem. Note that the desired result (31) is equivalent to:

$$\frac{\sum_{k=r}^{\infty} \frac{\lambda^{k+n}}{(k+n)!}}{\sum_{k=r}^{\infty} \frac{\lambda^k}{k!}} \leq \frac{\sum_{k=0}^{\infty} \frac{\lambda^{k+n}}{(k+n)!}}{\sum_{k=0}^{\infty} \frac{\lambda^k}{k!}}$$

To prove the above inequality, define $a_k = \frac{\lambda^{k+n}}{(k+n)!}$ and $b_k = \frac{\lambda^k}{k!}$. From Lemma 4, it suffices to show that $\frac{a_k}{b_k}$ is decreasing in k. We have:

$$\frac{a_k}{b_k} = \frac{\lambda^n}{n!} \begin{pmatrix} k+n \\ n \end{pmatrix}^{-1}$$

which indeed decreases with k, proving the theorem.

References

[1] M. J. Neely and E. Modiano. Logarithmic delay for $n \times n$ packet switches. *IEEE Workshop on High Performance Switching and Routing*, April 2004.

[2] C-S Chang, W-J Chen, and H-Y Huang. Birkhoff-von neumann input buffered crossbar switches. *IEEE Proceedings of Infocom*, 2000.

[3] C-S Chang, D-S Lee, and C-Y Yue. Providing guaranteed rate services in the load balanced birkhoff-von neumann switchies. *IEEE Proceedings of INFOCOM*, April 2003.

[4] E. Leonardi, M. Mellia, F. Neri, and M. A. Marsan. On the stability of input-queued switches with speedup. *IEEE/ACM Transactions on Networking*, Vol. 9, n.1, Feb. 2001.

[5] C.E. Koksal. *Providing Quality of Service over Electronic and Optical Switches*. PhD thesis, Massachusetts Institute of Technology, Laboratory for Information and Decision Systems (LIDS), Sept. 2002.

[6] M. Andrews and M. Vojnović. Scheduling reserved traffic in input-queued switches: New delay bounds via probabilistic techniques. *IEEE Proceedings of INFOCOM*, April 2003.

[7] N. McKeown, V. Anantharam, and J. Walrand. Achieving 100% throughput in an input-queued switch. *Proc. INFOCOM*, 1996.

[8] 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.

[9] E. Leonardi, M. Mellia, F. Neri, and M. Ajmone Marsan. Bounds on average delays and queue size averages and variances in input-queued cell-based switches. *Proc. INFOCOM*, 2001.

[10] M. A. Marsan, A. Bianco, P. Giaccone, E. Leonardi, and F. Neri. Multicast traffic in input-queued switches: Optimal scheduling and maximum throughput. *IEEE/ACM Transactions on Networking*, Vol. 11 (3), June 2003.

[11] M. Gamvrili and D. N. Serpanos. Multicast schedulers for atm switches with multiple input queues. *Proceedings of ISSPIT*, 2004.

[12] S-T Chuang, A. Goel, N. McKeown, and B. Prabhakar. Matching output queueing with a combined input output queued swtich. *IEEE Proceedings of Infocom*, 1998.

[13] S. Sarkar. Optimum scheduling and memory management in input queued switches with finite buffer space. *IEEE Proceedings of INFOCOM*, April 2003.

[14] T. Weller and B. Hajek. Scheduling nonuniform traffic in a packet switching system with small propagation delay. *IEEE/ACM Transactions on Networking*, 5 (6), 588-598, 1998.

[15] M. Andrews and L. Zhang. Achieving stability in networks of input-queued switches. *IEEE Proceedings of INFOCOM*, 2001.

[16] M. J. Neely, J. Sun, and E. Modiano. Delay and complexity tradeoffs for routing and power allocation in a wireless network. *Allerton Conference on Communication, Control, and Computing*, Oct. 2002.

[17] S. Iyer and N. McKeown. Maximum size matchings and input queued switches. *Proceedings of the 40th Annual Allerton Conference on Communication, Control, and Computing*, October 2002.

[18] I. Gopal, D. Coppersmith, and C. K. Wong. Minimizing packet waiting time in a multibeam satellite system. *IEEE Transactions on Communications*, COM-30 (2), February 1982.

[19] M. A. Bonuccelli, I. Gopal, and C. K. Wong. Incremental time-slot assignment in ss/tdma satellite systems. *IEEE Transactions on Communications*, Vol. 39 (2), July 1991.

[20] M. J. Neely, E. Modiano, and C. E. Rohrs. Tradeoffs in delay guarantees and computation complexity for $n \times n$ packet switches. *Proceedings of the Conference on Information Sciences and Systems, Princeton University*, March 2002.

[21] D. Shah and M. Kopikare. Delay bounds for the approximate maximum weight matching algorithm for input queued switches. *IEEE Proceedings of INFOCOM*, 2002.

[22] G. Birkhoff. Tres observaciones sobre el algebra lineal. *Univ. Nac. Tucumán Rev. Ser. A*, Vol. 5, pp. 147-151, 1946.

[23] J. von Neumann. A certain zero-sum two-person game equivalent to the optimal assignment problem. *Contributions to the Theory of Games*, Vol. 2, pp. 5-12, 1953.

[24] D. P. Bertsekas and R. Gallager. *Data Networks*. New Jersey: Prentice-Hall, Inc., 1992.

[25] P. Humblet. Determinism minimizes waiting time. *MIT LIDS Technical Report P-1207*, May 1982.

[26] D. Gamarnik. Stability of adaptive and non-adaptive packet routing policies in adversarial queueing networks. *Proc. of 31st ACM Symposium on the Theory of Computing*, 1999.

[27] M. Hall Jr. *Combinatorial Theory*. Waltham, MA: Blaisedell, 1969.

[28] C.H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization: Algorithm and Complexity*. New Jersey: Prentice Hall, 1982.

[29] J. Hopcroft and R. Karp. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM J. Comput.*, Dec. 1973.

Michael J. Neely received B.S. degrees in both Electrical Engineering and Mathematics from the University of Maryland, College Park, in 1997. He was then awarded a 3 year Department of Defense NDSEG Fellowship for graduate study at the Massachusetts Institute of Technology, where he received an M.S. degree in 1999 and a Ph.D. in 2003, both in Electrical Engineering. During the Summer of 2002, he worked in the Distributed Sensor Networks group at Draper Labs in Cambridge. In 2004 he joined the faculty of the Electrical Engineering Department at the University of Southern California, where he is currently an Assistant Professor. His research is in the area of stochastic network optimization for satellite and wireless networks, mobile ad-hoc networks, and queueing systems. Michael is a member of Tau Beta Pi and Phi Beta Kappa.

Eytan Modiano received his B.S. degree in Electrical Engineering and Computer Science from the University of Connecticut at Storrs in 1986 and his M.S. and PhD degrees, both in Electrical Engineering, from the University of Maryland, College Park, MD, in 1989 and 1992 respectively. He was a Naval Research Laboratory Fellow between 1987 and 1992 and a National Research Council Post Doctoral Fellow during 1992-1993 while he was conducting research on security and performance issues in distributed network protocols.

Between 1993 and 1999 he was with the Communications Division at MIT Lincoln Laboratory where he designed communication protocols for satellite, wireless, and optical networks and was the project leader for MIT Lincoln Laboratory's Next Generation Internet (NGI) project. He joined the MIT faculty in 1999, where he is presently an Associate Professor in the Department of Aeronautics and Astronautics and the Laboratory for Information and Decision Systems (LIDS). His research is on communication networks and protocols with emphasis on satellite, wireless, and optical networks.

He is currently an Associate Editor for Communication Networks for IEEE Transactions on Information Theory and for The International Journal of Satellite Communications. He had served as a guest editor for IEEE JSAC special issue on WDM network architectures; the Computer Networks Journal special issue on Broadband Internet Access; the Journal of Communications and Networks special issue on Wireless Ad-Hoc Networks; and for IEEE Journal of Lightwave Technology special issue on Optical Networks. He is the Technical Program co-chair for Wiopt 2006 and Infocom 2007.

Yuan-sheng Cheng received the B.S.E. degree in Electrical Engineering from Taiwan University, Taipei, Taiwan in 1998 and M.S. degree in Electrical Engineering in 2003 from University of Southern California. He is currently an P.H.D student at the University of Southern California. His research focuses primarily on applying message passing algorithms to basic networking problems, including weighted matching problems and low complexity cross-layer power control.