# Fair scheduling through packet election

Srikanth Jagabathula    Vishal Doshi    Devavrat Shah

***Abstract*—In this paper, we consider the problem of designing a scheduling algorithm for input queued switches, that is both fair as well as throughput optimal. Most of the existing literature on input-queued switch fairness criteria concentrates on flow-based fairness. Since a large fraction of network traffic is about "short-flows", there is a need for packet-based fairness criterion. The significant body of literature developed over the past two decades for packet-based scheduling algorithms is primarily concerned with throughput and delay, but not fairness. One of the reasons for such a state of affairs is the lack of a proper definition for packet-based *fairness*. The difficulty in defining fair stems from the fact that any reasonable notion of fairness must combine the well-known notion of fairness for a single-queue with the scheduling constraint of an input queued switch in an appropriate manner.**

**As one of the main results of this paper, we define a notion of packet-based fair scheduling by identifying it as the selection of a winner in the following ranked election: packets are voters; schedules are candidates and each packet ranks different schedules based on their priorities. Drawing upon the seminal work of Goodman and Markowitz (1952) on ranked elections, we obtain a unique characterization of the fair schedule.**

**Another important contribution of this paper is proving that the thus obtained fair scheduling algorithm is throughput optimal. There is no a priori reason why this should be true, and we introduce some non-standard proof techniques to prove the result. Our results suggest a framework for defining fair scheduling algorithm for a constrained packet network; a non-standard method to prove throughput stability for algorithms, such as ours, that are not based on queue-sizes.**

## I. Introduction

We consider the problem of designing fair scheduling algorithms in constrained packet networks. Specifically, we consider this question in the context of scheduling in a switch.

The primary function of a switch, residing in an Internet router, is to transfer packets from ingress (input) ports to appropriate egress (output) ports through a switch fabric Throughout the paper, we will consider an $N$-port switch, with $N$ input and $N$ output ports, denoted as an $N \times N$ switch. The time is assumed to be slotted, all packets are of unit size and line-speeds are normalized so that at most one packet can arrive (depart) to (from) an input (output) port in a time-slot.
*Input-queued (IQ) switch:* Here buffers are placed only at the input ports and the switch fabric is a cross-bar and hence, in a given time-slot (a) each input can send out at most one packet and (b) each output can receive at most one packet.
*Output-queued (OQ) switch:* Here buffers are placed at the output ports. All packets arriving at input ports are immediately transferred to their respective output ports.

All authors are with the Department of Electrical Engineering and Computer Science at MIT and part of Laboratory of Information and Decision Systems (LIDS). Email: `jskanth@mit.edu`, `vdoshi@mit.edu`, `devavrat@mit.edu`. This work was supported in part by NSF CAREER CNS-0546590 and NSF CCF-0728554.

*Combined input-output queued (CIOQ) switch:* Here, there are buffers at both the input and output ports so that the switch fabric operates at a speedup $S > 1$: that is, upto $S$ packets are transferred from each input and upto $S$ packets are received by each output in a time-slot.

In an OQ switch, a packet contends for bandwidth only with packets buffered at the same output port. Therefore, the notion of fair bandwidth allocation in an OQ switch is equivalent to that for a queue with a single server. In the context of a single queue, fair scheduling has been widely studied, since the early 1990s. In one of the earlier works by Demers, Keshav and Shenker [1], the authors proposed the notion of *Weighted Fair Queueing* (WFQ) and its packetized implementation. Parekh and Gallager [2], [3] analyzed the performance of this packetized implementation and showed it to be a good approximation of *Generalized Processor Sharing* (GPS). Shreedhar and Varghese [4] designed a computationally efficient version of weighted fair queuing called *Deficit Weighted Round Robin* (DWRR). There has been work on fair queueing algorithms by means of designing packet-drop mechanisms for Internet routers such as RED [5], CHoKe [6] and AFD [7].

Clearly, these approaches provide a definition of fair scheduling, and algorithms to realize them in an OQ switch. However, it does not immediately extend for the case of IQ or CIOQ switch due to scheduling constraints. Therefore, one approach is to emulate the performance of such an unconstrained OQ switch by means of a CIOQ switch with minimal speedup. This approach was taken by Prabhakar and McKeown [8] and Chuang, Goel, McKeown and Prabhakar [9] where they showed that, essentially a speedup of 2 is necessary and sufficient for emulating an OQ switch where OQ switch can be operating under various policies like FIFO, WFQ, DWRR, strict priority, etc. Equivalently, if an IQ switch is loaded upto 50% of its capacity and the notion of fairness is defined by policies like FIFO, WFQ, DWRR, strict priority, etc., then by emulating an OQ switch with these policies, it is possible to have fair scheduling for the IQ switch. However, for higher loading this approach will fail due to inability of emulating OQ switch.

This necessitates the need for defining an appropriate notion of fairness that cleverly and in a reasonable manner combines the preferences of packets based on some absolute notions along with the scheduling constraints. In principle, this question is very similar to the question answered by *utility maximization* based framework for bandwidth allocation in a flow network. In fact, most of the existing literature on fair scheduling algorithms for input-queued switches is concerned with the notion of flow-based fairness (see for example [10],

[11] and [12]). Next, we explain why such an approach is inappropriate for packet scheduling.

To address the issue of fairness in network, Kelly, Maullo and Tan [13] proposed a flow-level model for the Internet. Under this model, the resource allocation that maximizes the global network utility provides a notion of fair rate allocation. We refer an interested reader to survey-style papers by Low [14] and Chiang et.al. [15] and the book by Srikant [16] for further details. We take a note of desirable throughput property of the dynamic flow-level resource allocation model (see for example, [17], [18]). This flow-based resource allocation approach does not work for our setup for the following two primary reasons: (a) our unit of data is packet which can either be served or not served, in contrast a flow can be allocated any continous amount of rate; and (b) packets have priorities and they lack explicit utility functions.

We also note that packetized algorithms can be made to mimic flow-based algorithms with arbitrary precision. But, network traffic predominantly contains "short-flows" and arbitrary precision algorithms applied to short-flows can lead to two issues: (a) flow-based approach requires existence of ever-lasting flows and hence can induce huge delays when applied naively; and (b) implementation of such an approach would require large amount of complex data structures, which would be difficult to maintain and update at high speeds (e.g. 10 Gbps). In summary, our question is inherently combinatorial which requires dealing with *hard* combinatorial constraints unlike the resource allocation in a flow network which deals with *soft* capacity constraints in a continuous optimization setup.

## A. Our contribution

Having identified the need for packet-based fair scheduling algorithm, we will propose a notion of packet-based fairness and an algorithm to determine the schedule. Some significant contributions of this paper are:

- A notion of packet-based fairness criterion using a novel analogy between scheduling and ranked-election process.
- An algorithm to determine the fair schedule.
- A proof of throughput optimality of the algorithm by introducing non-traditional techniques that may be extended to the analysis of non-queue based weighted algorithms.

We will now briefly explain the analogy between switch scheduling and ranked-election process. As noted earlier, the OQ emulation based approach, though provides an immediate way to define fairness for an IQ switch, leads to a reduction in effective capacity. In order to fully utilize the IQ switch capacity, we need to define a notion of fairness for queues in contention. Inherently, each queue prefers to be served. Thus, each queue has preferences over all feasible schedules. We obtain relative order of preferences between various queues with the help of a *shadow* OQ switch (with exactly the same arrival process) with appropriate policy, running in the background (such policy, e.g. FIFO or WFQ at the output queue of OQ decide the departure times of the packets). Here, a packet has higher preference than another packet if its departure time from the shadow OQ switch is earlier than that of the other packet. Since we can use any cardinal preferences to implement the ranked-election algorithm, choice of OQ departure times might seem arbitrary. Intuitively, this choice enables us to leverage the well studied notion of packet-based fairness for single queues.

Under this setup, the problem of fair scheduling is one of choosing a fair "socially" preferred schedule. This is equivalent to the ranked-election problem: packets (or queues) are voters, schedules are candidates and each packet has a ranking of all the schedules. The question of ranked-election is very well-studied in the economics literature (also called theory of social choice). In our setup, the preferences of packets over schedules are naturally quantitative. When preferences over candidates are quantitative (or cardinal in language of economics literature), Goodman and Markowitz [19] prove that under certain socially-desirable postulates(detailed in Section III-A), a simple function of those preferences will give a uniquely preferred outcome. Following this, we show that the preferred schedule is equivalent to a maximum weight matching where the weights are related to preference levels.

Thus, we obtain a definition of fair scheduling by combining the preferences of packets derived from a virtually running shadow OQ switch along with the ranked election algorithm. We establish that such an algorithm is throughput optimal under standard stochastic model of a switch. To prove throughput optimality (rate stability to be precise), we use an appropriate quadratic Lyapunov function. However, we are unable to use the standard stability proof technique based on Foster's criterion, as the Lyapunov function is not a function of queue-size, but is function of *preferences* derived from OQ switch. This makes the analysis rather non-trivial.

To explain the consequences of our algorithm on fair emulation, we present simulations for algorithms based on FIFO OQ switch. Intuitively, our fair algorithm should be able to reduce the queue-size (or delay) as well as get rid of starvation caused by well-known throughput optimal algorithm. Our simulation results clearly confirm this intuition.

In summary, our work provides a framework for designing fair scheduling algorithms for constrained packet networks while achieving high-performance. We establish this claim in the context of an input queued switch. In this paper we restrict ourselves to only throughput analysis and centralized algorithm design. Further, understanding of fairness property as well as decentralized algorithm design will be natural next steps of research.

## II. Model and notation

We consider an $N \times N$ switch operating in discrete slotted time, which is indexed by $n$. There are $N^2$ queues: at each input port $i$, there is a seperate queue for each output port $j$, called virtual output queue (VOQ) whose size is denoted by $Q_{ij}(n)$, $1 \leq i, j \leq N$ at the begining of time $n$. We assume that switch starts empty at time $n = 0$, i.e. $Q_{ij}(0) = 0$ for all $i, j$. At each input $i$, at most one packet arrives in each time slot. Let $A_{ij}(n) \in \{0,1\}$ denote the number of packets arriving at input $i$ for output $j$ at the end of time slot $n$. By definition, we have $\sum_{k=1}^{N} A_{ik}(n) \in \{0,1\}$. We assume that arrival process is Bernoulli i.i.d with rate matrix $\lambda = [\lambda_{ij}]$. That is, $\mathbb{P}(A_{ij}(n) = 1) = \lambda_{ij}$ for all $n$ and $A_{ij}(\cdot)$ form an i.i.d. sequence of random variables. Note that $A_{ij}(n)$ and $A_{ij'}(n)$ are dependent for $j \neq j'$; $A_{ij}(n)$ and $A_{i'j'}(n)$ are independent for $i \neq i'$. Further, $A(n) = [A_{ij}(n)]$ is independent of $A(n')$, for $n \neq n'$. An arrival rate matrix, $\lambda$ is called *strictly admissible* if the following holds: for all $1 \leq i,j \leq N$

$$\sum_{k=1}^{N} \lambda_{ik} < 1, \ \sum_{k=1}^{N} \lambda_{kj} < 1.$$

We assume that the switch is running at a speedup of 1. Thus, in each time slot, at most one packet can be served from each queue and at most one packet can be received at each output. Let $S(n) = [S_{ij}(n)] \in \{0,1\}^{N \times N}$ denote the schedule matrix at time $n$ with the understanding that $S_{ij}(n) = 1$ only if the queue is served at time $n$. By definition of scheduling constraint we have that for all $1 \leq i,j \leq N$,

$$\sum_{k=1}^{N} S_{ik} = 1, \ \sum_{k=1}^{N} S_{kj} = 1$$

Thus, each schedule corresponds to a permutation matrix in this notation. Call this set as $\mathcal{M}$. We will also use the following alternative notation: schedule $S \in \mathcal{M}$ is equivalent to permutation $\sigma \in \mathcal{M}$ where $\sigma$ maps $i$ to $\sigma(i)$ if and only if $S_{i\sigma(i)} = 1$, for $1 \leq i \leq N$.

If $S_{ij}(n) = 1$ and $Q_{ij}(n) > 0$ then a packet departs from $Q_{ij}(\cdot)$. Let $D_{ij}(n)$ denote cumulative departure process, i.e.

$$D_{ij}(n) = \sum_{m \leq n} S_{ij}(m) \mathbf{1}_{\{Q_{ij}(m) > 0\}}.$$

We call a system *rate stable* or simply *stable* in this paper if following holds with probability 1: for all $i,j$

$$\lim_{n \to \infty} \frac{D_{ij}(n)}{n} = \lambda_{ij}.$$

To understand when one can even design a rate stable algorithm, it is important to understand the scheduling constraints. Note that the convex hull, denoted by $\mathcal{S}$, of the set of all schedules $\mathcal{M}$ is precisely the set of all doubly stochastic matrices. This follows from the Birkhoff-Von Neumann's result that the extreme points of the set of doubly stochastic matrices are precisely the permutation matrices. By definition, each strictly admissible arrival rate matrix is strictly doubly sub-stochastic matrix. Therefore, there is a simple Time-Division Multi-Access (TDMA) scheme (based on convex decomposition of $\lambda$) that can allocate rates to each queue strictly higher than the arrival rate and thus have system rate stable.

Such algorithm is not myopic in the sense that it depends on knowledge of $\lambda$. In their seminal work, Tassiulas and Ephremides [20] (and independently obtained by McKeown et. al. [21]) showed that the maximum weight schedule algorithm is rate stable, which chooses schedule $S^*(n)$ so that

$$S^*(n) \in \arg\max_{S \in \mathcal{M}} \sum_{i,j=1}^{N} Q_{ij}(n) S_{ij}.$$

Since these results, there has been a significant work on designing high-performance, implementable packet scheduling algorithms that are derivatives of maximum weight scheduling, where weight is some function of queue-sizes. All of these algorithms are designed to optimize network utilization as well as minimize delay (for example, see recent work by Shah and Wischik [22]). However, these algorithms ignore the requirement of fairness. Specifically, it has been observed that the maximum weight based algorithm can lead to unwanted starvation or very unfair rate allocation when switch is overloaded (for example, see work by Kumar, Pan and Shah [23]).

## III. Fair scheduling algorithm

In this section, we describe our fair scheduling algorithm. First, we describe the standard problem of ranked election and some relevant background. Then, we describe fair algorithm that is derived from ranked elections by establishing equivalence between the question of fair scheduling and ranked election.

### A. Ranked election

*Definition 1 (Ranked election):* There are $M$ voters that vote for $C$ candidates. Vote of each voter consists of a ranking (or permutation) of all $C$ candidates. These votes can additionally carry quantative values associated with their preferences. Let $a_{mc}$ denote the value voter $m$ gives to candidate $c$, for $1 \leq m \leq M, 1 \leq c \leq C$. The goal of the election is to relative order all the $C$ candiates as well as produce the ultimate winner in a manner that is *consistent* with the votes. The key for a good election lies in defining *consistency* of the outcome of election with votes. The following are canonical postulates that are used in the literature on ranked election:

- P1. Between any two candidates $c$ and $c'$, suppose that none of the $M$ voters prefers $c'$ to $c$ and at least one voter prefers $c$ to $c'$. Then $c'$ should not be ranked higher than $c$ in the output of the election. This property corresponds to the economic notion of *weak Pareto optimality*.
- P2. Suppose the voters are renumbered (or renamed) while keeping their votes the same. Then the outcome of election should remain the same. In other words, the election outcome is blind to the identity of the voters, that is election outcome is symmetric.
- P3. Now, consider the setup when the votes are cardinal (i.e., quantitative). Suppose candidate $c$ is preferred to $c'$ by the election. Then, by adding the same fixed constant to all $a_{mc}$ and $a_{mc'}$ for $1 \leq m \leq M$, the relative order of candidates $c$ and $c'$ should not change. This makes sense because what matters is the difference in preference levels for the two candidates, not the actual values.

In the absence of cardinal (or quantitative) preferences, the question of ranked election with postulates P1, P2 (and some additional postulates) was first studied by Arrow [24]. In his celebrated work, he established the (then) very surprising impossibility of the existence of any election scheme that satisfies P1 and P2 simultaneously. We note that this result has been an important corner stone in the field of theory of social choice.

Subsequent to Arrow's impossibility result, many economists started looking for positive results. Among many other celebrated results, the result that is relevant to this paper is that of Goodman and Markowitz [19]. They showed that if voters have cardinal preferences, as in our setup, then there is a unique ordering of candidates that satisfies P1-P2-P3 simultaneously. To describe their result, consider the following: let the net score of a candidate $c$ be $s_c = \sum_{m=1}^{M} a_{mc}$. Goodman and Markowitz obtained the following remarkable result.

*Theorem 1:* Suppose the scores of all candidates are distinct. Rank candidates as follows: candidate $c$ has higher ranking than $c'$ if and only if $s_c > s_{c'}$. This ranking satisfies postulates P1-P2-P3. Further, this is the only such ranking.
For a proof of this result, we refer a reader to [19].

## B. Fair scheduling = ranked election

Now, we shall establish a natural connection between selection of a fair schedule for a switch and the problem of ranked election. We want the reader to pay attention to the fact that, the equivalence used here between fair scheduling and ranked election easily extends to a general network scheduling problem.

To define a fair scheduling algorithm, we will use a shadow output queued switch. Specifically, a copy of every packet arriving to the IQ switch is fed to a virtual shadow OQ switch. That is, (copy of) a packet arriving at input $i$ for output $j$ in the IQ switch immediately joins queue at output $j$ in the OQ switch. The departure from the queues at outputs in OQ switch happens according to an appropriate fair scheduling policy, say $\mathcal{P}$, such as strict priority scheme, last-in-first-out or simply first-in-first-out applied at the output queues of OQ switch. Let $d_{ij}^k$ be the departure time of $k^{th}$ packet arrived at input $i$ for output $j$ from the OQ switch. Clearly, these departure times are dependent on the policy $\mathcal{P}$ used in the switch. If OQ emulation was possible, the IQ switch would like to send out the packets at exactly the same times as $(d_{ij}^k)_{i,j,k}$. However, at speedup 1 it is not possible to emulate OQ perfectly as shown by Chuang et. al. [9]. Thus, the whole challenge is to respect the scheduling constraints of the IQ switch, while letting packets depart from the switch so that they are as faithful to the departure times obtained from shadow OQ as possible.

Now, the scheduling problem can be viewed as that of choosing one of the $N!$ possible schedules from $\mathcal{M}$ every time. At each input, there are various packets waiting, possibly in different VOQs, to depart from it. Let us assume that, in a VOQ, packets are ordered in the increasing order of their departure times from the corresponding shadow OQ. Thus, for $k \neq k'$ if $d_{ij}^k > d_{ij}^{k'}$ then $k'$ is ahead of $k$ in the VOQ$_{ij}$. Thus, the most *urgent* packet as per the shadow OQ is the Head-Of-Line (HOL) packet in a queue. In what follows, queues will have preferences over schedules given by the departure times of the HOL packets. Therefore, for the purpose of selecting a fair schedule, it will be sufficient to consider such HOL packets. For the following discussion, assume that all $N^2$ queues are non-empty in a switch and hence have a HOL packet. Later, we will deal with empty queues appropriately. Let $d_{ij}(n)$ be the departure time w.r.t. the shadow OQ, of the HOL packet in VOQ$_{ij}$ at time $n$.

Now, each HOL packet (or queue) prefers being served and hence prefers all the schedules that serve it (i.e. when queue containing it is part of the schedule) over all those schedules that do not. Further, among those schedules that do serve it, the packet is indifferent. Similarly, it is indifferent to all schedules that do not serve it. Thus, each of the $N^2$ HOL packets have a preference list over the $N!$ schedules or matchings. Consider a schedule $\sigma \in \mathcal{M}$: it matches input $i$ to output $\sigma(i)$ for $1 \leq i \leq N$. Now, when a queue at input $i$ for output $j$ is served, it should provide a value that is higher if $d_{ij}(n)$ is smaller and vice versa. Specifically, we assign the value $-d_{ij}(n)$ to the serving queue at input $i$ for output $j$. Therefore, net value of a schedule $\sigma$ at time $n$ is given as:

$$\mathsf{value}(\sigma)(n) = -\sum_{i=1}^{N} d_{i\sigma(i)}(n).$$

The postulates P1-P2-P3 translate into the following postulates for the switch scheduling.

P1'. Between any two matchings $\sigma_1$ and $\sigma_2$, suppose that none of the $N^2$ HOL packets prefer $\sigma_2$ to $\sigma_1$ and at least one HOL packet prefers $\sigma_1$ to $\sigma_2$. Then, we should not choose $\sigma_2$.

P2'. For given HOL packets, let $\sigma$ be the outcome of the election as per the above preferences for schedules. Then, by renumbering queues while retaining the same HOL preferences, the outcome of election should be only renumbered $\sigma$. In other words, the election does not give unfair priority to any port and thus is symmetric in its inputs.

P3'. Suppose matching $\sigma_1$ is preferred to $\sigma_2$ by the election. By adding fixed constant to $-d_{ij}(n)$ for all $i, j$, the outcome of the election should remain unchanged.

The election algorithm of Goodman and Markowitz suggests that the following schedule $S^*(n)$ should be chosen:

$$S^*(n) \in \arg\max_{\sigma \in \mathcal{M}} -\sum_{i=1}^{N} d_{i\sigma(i)}(n).$$

Thus, the algorithm is the *maximum weight schedule* (MWS) where weight is given by $-d_{ij}(n)$ at time $n$.

**Algorithm.** We call the above algorithm as the most urgent cell first (MUCF). We will explicitly denote the dependence of the algorithm on policy, $\mathcal{P}$, used in the shadow OQ switch for scheduling at the output queues. That is, we call it algorithm

MUCF($\mathcal{P}$). Now, using postulate P3', we use weights of non-emtpy VOQ$_{ij}$ as $n - d_{ij}(n)$. Call it urgency of VOQ$_{ij}$ at time $n$, denoted by $U_{ij}(n)$. Now we define urgency for empty queue. Let VOQ$_{ij}$ be empty at time $n$. Then define $U_{ij}(n)$ as $-\max\{0, -\min_{\{i'j':Q_{i'j'}(n)\neq 0\}} U_{i'j'}(n)\}$. Thus, the MUCF algorithm chooses schedule $S^*(n)$ at time $n$ where

$$S^*(n) \in \arg\max_{\sigma\in\mathcal{M}} \sum_{i=1}^{N} U_{i\sigma(i)}(n).$$

## IV. MUCF($\mathcal{P}$) Algorithm: Throughput

The previous section described how we arrived at MUCF algorithm as a fair algorithm based on preferences obtained from a shadow output queued switch. As established in the previous section, Theorem 1 implies that MUCF is the only algorithm that satisfies the desirable postulates P1'-P2'-P3'. In this section, we state and prove the throughput optimality property of the MUCF algorithm. The proof of the algorithm is non-traditional and requires new techniques that may be of interest for analysis of such non-queue based weighted algorithms.

*Theorem 2:* Any $N \times N$ switch operating under MUCF($\mathcal{P}$) algorithm with Bernoulli i.i.d. arrival process with arrival rate matrix being strictly admissible is rate stable when $\mathcal{P}$ is FIFO.

### A. Proof of Theorem 2

Here, we present a proof of Theorem 2 for FIFO policy. The proof is some what involved and uses non-traditional methods for proving the result.

**Notation.** First, some uesful notation. Consider the packet that is HOL for VOQ$_{ij}$ at time $n$: let $a_{ij}(n)$ be its time of arrival at VOQ$_{ij}$; as before $d_{ij}(n)$ be the time of its departure from the shadow OQ switch; $U_{ij}(n)$ be its urgency as defined above and $W_{ij}(n)$ be its waiting time (i.e. $n - a_{ij}(n)$). Also, define $\Delta_{ij}(n) = W_{ij}(n) - U_{ij}(n)$. We note that if VOQ$_{ij}$ is empty, then $W_{ij}(n) = 0$ and $U_{ij}(n)$ is as defined above. Hence, $\Delta_{ij}(n)$ is always non-negative. Let $Q_j(k)$ denote the length at the end of time slot $k$, of the output queue $j$ of the shadow Output Queued switch.

**Fact (Birkhoff-Von Neumann).** Any strictly admissible arrival rate matrix $\lambda$ can be decomposed as follows: for some $\beta \in (0,1)$,

$$\lambda \leq \sum_{k=1}^{N^2} \alpha_k \pi^k; \quad \alpha_k \geq 0, \quad \sum_k \alpha_k = 1-\beta, \ \pi^k \in \mathcal{M}.$$

The proof of Theorem uses the following Lemmas. We will present the proofs of Lemmas after completing proof of Theorem 2 for FIFO policy.

*Lemma 3:* Let $L(n) = \sum_{ij} W_{ij}^2(n)\lambda_{ij}$. Then, under MUCF(FIFO) algorithm with $\lambda$ being strictly admissible, there exists $\varepsilon > 0$ such that

$$\mathbb{E}\left[L(n+1) - L(n)\right] \leq -\varepsilon\mathbb{E}\left[\sum_{ij} W_{ij}(n)\right] + 2\mathbb{E}\left[\sum_{ij} \Delta_{ij}(n)\right] + K,$$

for some constant $K$.

*Lemma 4:* Let $M_j(n)$ denote $\max_{0\leq k\leq n} Q_j(k)$. Then, under the FIFO policy and strictly admissible $\lambda$, the following is true for all $n$,

$$\mathbb{E}[M_j(n)] \leq O(\log n)$$

*Proof of Theorem 2.* We first note that, if VOQ$_{ij}$ is non-empty

$$\Delta_{ij}(n) \leq \max_{0\leq k\leq n} Q_j(k) \tag{1}$$

This is true because $\Delta_{ij}(n)$ denotes the waiting time in the output queued switch of the HOL packet at VOQ$_{ij}$ and hence cannot be more than the size of the queue at the end of the time slot it arrived. Since $\Delta_{ij}(n)$ corresponds to a packet that arrived before the time $n$, the inequality we claim should be true.

Therefore, from lemma 4 it follows that:

$$\mathbb{E}\left[\Delta_{ij}(n)\right] \leq O(\log n) \tag{2}$$

If VOQ$_{ij}$ is empty, then either $\Delta_{ij}(n) = 0$ or $\Delta_{ij}(n) = d_{i'j'}(n) - n$, where VOQ$_{i'j'}$ is non-empty. Since, $\Delta_{lk} \geq 0$ $\forall l,k$, we have from (2) that $\mathbb{E}[\Delta_{ij}(n)] \leq \mathbb{E}[\Delta_{i'j'}(n)] \leq O(\log n)$. Hence, (2) is valid even for empty queues and it follows that:

$$\mathbb{E}\left[\sum_{ij} \Delta_{ij}(n)\right] \leq O(\log n) \tag{3}$$

From lemma 3 and (3), we obtain the following:

$$\mathbb{E}\left[L(n+1) - L(n)\right] \leq -\varepsilon\mathbb{E}\left[\sum_{ij} W_{ij}(n)\right] + O(\log n) + K, \tag{4}$$

Telescopic summation of (4) from $1,\ldots,n$, we obtain (after cancellations),

$$\mathbb{E}\left[L(n+1)\right] \ \leq \ \mathbb{E}[L(0)] - \varepsilon\mathbb{E}\left[\sum_{m=1}^{n} |W(m)|\right] + O(n\log n) + nK, \tag{5}$$

where $|W(m)| = \sum_{ij} W_{ij}(m)$ represents the $\ell_1$-norm of matrix $W(m)$. Now switch starts empty at time 0. Therefore, $\mathbb{E}[L(0)] = 0$. Further, $L(\cdot)$ is non-negative function. Therefore, (5) gives us

$$\varepsilon\mathbb{E}\left[\sum_{m=1}^{n} |W(m)|\right] \ \leq \ O(n\log n) + nK. \tag{6}$$

Dividing both sides by $\varepsilon n \log n$, we obtain

$$\mathbb{E}\left[\frac{1}{n\log n}\sum_{m=1}^{n} |W(m)|\right] \ \leq \ O(1) \tag{7}$$

Let $X_n = \frac{1}{n}\sum_{m=1}^{n} |W(m)|$ and $Z_n = \frac{X_n}{\log n}$. From (7), we have $\mathbb{E}[Z_n] \leq O(1) < \infty$ for all $n$. Now, we claim that

$$\Pr\left(\lim_m \frac{1}{m}|W(m)| = 0\right) = 1. \tag{8}$$

The proof of (8) is presented later. Before that, we use it to complete the proof of rate stability of the algorithm. Now, at time $n$ the waiting time of HOL packet of VOQ$_{ij}$ is $W_{ij}(n)$. Under FIFO policy and due to at most one arrival per input port, we have that the queue-size of VOQ$_{ij}$ at time $n$, $Q_{ij}(n) \leq W_{ij}(n)$. From (8), we have that

$$\lim_{n\to\infty} \frac{Q_{ij}(n)}{n} \quad = \quad 0, \quad \text{with probability } 1. \tag{9}$$

Now, $Q_{ij}(n)$ observes the following dynamics:

$$Q_{ij}(n) \quad = \quad Q_{ij}(0) + \sum_{m\leq n} A_{ij}(n) - D_{ij}(n), \tag{10}$$

where the second term on RHS is the cumulative arrival to VOQ$_{ij}$ till time $n$ while the third term is cumulative departure from VOQ$_{ij}$ till time $n$. By strong law of large numbers (SLLN) for Bernoulli i.i.d. process we have that

$$\lim_{n\to\infty} \frac{1}{n}\sum_{m\leq n} A_{ij}(n) = \lambda_{ij}.$$

Using this along with (9) and (10), we obtain

$$\lim_{n\to\infty} \frac{D_{ij}(n)}{n} = \lambda_{ij}, \quad \text{with probability } 1, \ \forall i,j.$$

This completes the proof of Theorem 2 with the remain claim (8), which we prove next.

Suppose (8) is not true. Then, since $|W(m)| \geq 0$ we have that for some $\delta > 0$,

$$\Pr\left(|W(m)| \geq \delta m, \ \text{i.o.}\right) \geq \delta, \tag{11}$$

where "i.o." means infinitely often. Now if $|W(m)| \geq \delta m$, then there exists an HOL packet that has been weighting in the switch for time at least $\delta m/N^2$. This is true because $|W(m)|$ is the sum of weighting times of at most $N^2$ HOL packets. Call this packet $p$. This packet must have arrived at time $\leq m - \delta m/N^2 = m(1-\delta N^{-2})$. Since waiting time of a packet increases only by 1 each time-slot, the waiting time of the packet $p$ must be at least $0.5\delta m/N^2$ in time interval $[m_1, m]$, where $m_1 = m - 0.5\delta m N^{-2} = m(1-0.5\delta N^{-2})$. Now, consider any time $m' \in [m_1, m]$. The packet waiting at the HOL of the queue that contains $p$ must have waiting time higher than that of $p$ due to FIFO ordering policy. Therefore, the contribution to $|W(m')|$ by HOL packets of the queue that contains packet $p$ is at least $0.5\delta m N^{-2}$. Therefore, we obtain

$$\sum_{m'=m_1}^{m} |W(m')| \quad \geq \quad \frac{\delta^2 m^2}{4N^4}. \tag{12}$$

Therefore, by the definition of $X_m$ and non-negativity of $|W(\cdot)|$, we have the following logical implication:

$$|W(m)| \geq \delta m \quad \Rightarrow \quad X_m \geq \frac{\delta^2 m}{4N^4}. \tag{13}$$

Thus, if (11) holds then by (13) we have

$$\Pr\left(X_m \geq \frac{\delta^2 m}{4N^4}, \ \text{i.o.}\right) \geq \delta. \tag{14}$$

Now observe the following relation of $X_n$: since $|W(\cdot)| \geq 0$,

$$X_{n+1} \geq \left(1 - \frac{1}{n+1}\right) X_n.$$

Hence, for $\alpha > 1$,

$$X_{n\alpha} \quad \geq \quad \left(1 - \frac{1}{n}\right)^{(\alpha-1)n} X_n. \tag{15}$$

Since

$$\left(1 - \frac{1}{n}\right)^{(\alpha-1)n} \approx \exp\left(-\alpha + 1\right),$$

we obtain that for there is large enough $n_o$ such that for $n \geq n_o$, for any $n' \in [n, 1.5n)$

$$X_{1.5n} \quad \geq \quad \frac{1}{3} X_{n'}. \tag{16}$$

Define, $Y_k = X_{1.5^k}$ for $k \geq 0$. Then, the following are direct implications of (16): for any $\theta > 0$,

$$X_m \geq \theta m, \ \text{ i.o. } \ \Rightarrow Y_k \geq \theta 1.5^k/3, \ \text{ i.o.};$$
$$Y_k \geq 3\theta 1.5^k, \ \text{ i.o. } \ \Rightarrow X_m \geq \theta m, \ \text{ i.o.}.$$

Therefore, to complete the proof of (8) by contradicting (11), it is sufficient to show that for $\theta = 3\delta$ (since $N \geq 1$),

$$\Pr\left(Y_k \geq \theta 1.5^k, \ \text{i.o.}\right) = 0.$$

For this, let event $E_k = \{Y_k \geq \theta 1.5^k\}$. Then, from $\mathbb{E}[Z_n] \leq O(1)$, relations $Y_k = X_{1.5^k}$, $Z_k = \frac{X_k}{\log k}$ and Markov's inequality we obtain that

$$\Pr(E_k) \leq O\left(\frac{k}{1.5^k}\right).$$

Therefore,

$$\sum_k \Pr(E_k) \quad \leq \quad \sum_k O\left(\frac{k}{1.5^k}\right) \quad < \quad \infty.$$

Therefore, by Borel-Cantelli's Lemma, we have that

$$\Pr(E_k \ \text{i.o.}) = 0.$$

This completes the proof of (8) and that of Theorem 2. $\square$

*Proof of Lemma 3.* Define the following: for all $i,j$

$$\widetilde{W_{ij}}(n+1) = W_{ij}(n) + 1 - S^*(n)\tau_{ij},$$

where $S^*(n)$ is the schedule of MUCF algorithm and $\tau_{ij}$ is the inter-arrival time for the arrival process to VOQ$_{ij}$. When queue is empty, treat $\tau_{ij}$ as an independent r.v. without any meaning, while if queue is not empty then treat it as the inter-arrival time between the packet being served and the packet behind it. In either case, due to FIFO policy the $\tau_{ij}$ is totally independent of the scheduling decisions performed by algorithm till (including) time $n$ and information utilized by the algorithm. Therefore, we will treat it as independent random variable with Geometric distribution of parameter $\lambda_{ij}$ (since arrival process is Bernoulli i.i.d.). Consider the following: for any $i, j$,

$$\begin{aligned}\widetilde{W}_{ij}^2(n+1)\lambda_{ij} &= W_{ij}^2(n)\lambda_{ij} + \lambda_{ij} - 2S_{ij}^*(n)\tau_{ij}\lambda_{ij} \\ &+ S_{ij}^*(n)\tau_{ij}^2\lambda_{ij} + 2W_{ij}(n)\lambda_{ij} - 2W_{ij}(n)S_{ij}^*(n)\tau_{ij}\lambda_{ij}\end{aligned} \tag{17}$$

Here we have used fact that $S_{ij}^*(n) \in \{0, 1\}$. Using (17) and $\tau_{ij}$ being Geometric r.v. with mean $1/\lambda_{ij}$, we have the following:

$$\begin{aligned}&\mathbb{E}\left[\sum_{ij}\widetilde{W}_{ij}^2(n+1)\lambda_{ij} - \sum_{ij}W_{ij}^2(n)\lambda_{ij} \mid W(n)\right] \\ &= 2\sum_{ij}W_{ij}(n)\lambda_{ij} - 2\sum_{ij}W_{ij}(n)S_{ij}^*(n) + \sum_{ij}\lambda_{ij} \\ &\quad - 2\sum_{ij}S_{ij}^*(n) + \sum_{ij}S_{ij}^*(n)\lambda_{ij}^{-1}.\end{aligned} \tag{18}$$

Using fact that $\sum_{ij}\lambda_{ij} \le N$, $\sum_{ij}S_{ij}^*(n) \le N$ and $\lambda_{ij}^{-1} < \infty$ for all $i, j$ such that $\lambda_{ij} \ne 0$, we obtain that

$$\begin{aligned}&\mathbb{E}\left[\sum_{ij}\widetilde{W}_{ij}^2(n+1)\lambda_{ij} - \sum_{ij}W_{ij}^2(n)\lambda_{ij} \mid W(n)\right] \\ &\le 2\sum_{ij}W_{ij}(n)\lambda_{ij} - 2\sum_{ij}W_{ij}(n)S_{ij}^*(n) + K,\end{aligned} \tag{19}$$

where $K$ is some large enough constant. Now, define $S^w(n)$ as

$$S^w(n) = \arg\max_{\sigma \in \mathcal{M}} \sum_i W_{i\sigma(i)}(n).$$

That is, $S^w(n)$ is the schedule whose weight is maximum where weight of VOQ$_{ij}$ is $W_{ij}(n)$. Now, define notation

$$\langle A, B\rangle = \sum_{ij} A_{ij}B_{ij}.$$

Consider the following:

$$\begin{aligned}&\langle W(n), \lambda - S^*(n)\rangle \\ &= \langle W(n), \lambda - S^w(n)\rangle + \langle W(n) - U(n), S^w(n) - S^*(n)\rangle \\ &\quad + \langle U(n), S^w(n) - S^*(n)\rangle.\end{aligned} \tag{20}$$

By definition of $S^*(n), S^w(n)$ and $\Delta(n)(= W(n) - U(n))$, it follows that

$$\langle U(n), S^w(n) - S^*(n)\rangle \quad \le \quad 0, \tag{21}$$
$$\langle W(n) - U(n), S^w(n) - S^*(n)\rangle \quad \le \quad \langle \Delta(n), \mathbf{1}\rangle, \tag{22}$$

where $\mathbf{1} = [1]$, the matrix of all 1. Now, for strictly admissible $\lambda$ using the **Fact**, we obtain that for some $\beta \in (0, 1)$,

$$\begin{aligned}&\langle W(n), \lambda - S^w(n)\rangle \\ &= \langle W(n), \sum_k \alpha_k\pi^k\rangle - \langle W(n), S^w(n)\rangle \\ &= \sum_k \alpha_k\langle W(n), \pi^k\rangle - \langle W(n), S^w(n)\rangle\end{aligned} \tag{23}$$

Since $S^w(n)$ is the maximum weight schedule with weight of VOQ$_{ij}$ as $W_{ij}(n)$:

$$\langle W(n), \pi^k\rangle \le \langle W(n), S^w(n)\rangle \quad \forall k \tag{24}$$

Thus, it follows from (23) and (24):

$$\langle W(n), \lambda - S^w(n)\rangle \le -\beta\langle W(n), S^w(n)\rangle. \tag{25}$$

Now, since all $N^2$ entries can be covered by $N$ distinct matchings, it follows that the weight of maximum weight matching is at least $1/N$ the sum of the weights of all entries. That is,

$$\langle W(n), S^w(n)\rangle \quad \ge \quad \frac{1}{N}\sum_{ij}W_{ij}(n) \quad = \quad \frac{|W(n)|}{N}. \tag{26}$$

Combining (18)-(26) and taking further expection with respect to $W(n)$, we obtain

$$\begin{aligned}&\mathbb{E}\left[\sum_{ij}\widetilde{W}_{ij}^2(n+1)\lambda_{ij} - \sum_{ij}W_{ij}^2(n)\lambda_{ij}\right] \\ &\le \quad -\varepsilon\mathbb{E}\left[|W(n)|\right] + 2\mathbb{E}\left[\sum_{ij}\Delta_{ij}(n)\right] + K,\end{aligned} \tag{27}$$

where $\varepsilon = 2\beta/N$. To complete the proof, note that if VOQ$_{ij}$ is non-empty after service at time $n$, then $\widetilde{W}_{ij}(n+1) = W_{ij}(n+1)$. Else, $W_{ij}(n+1) = 0$. Therefore, $\widetilde{W}_{ij}(n+1)^2 \ge W_{ij}^2(n+1)$. This inequality along with (27) imply the desired claim of Lemma 3. □

*Proof of Lemma 4.* To prove this lemma, we now consider a discrete time single FIFO queue: it has deterministic server that can serve one packet every time-slot and time i.i.d. arrival process with maximum number of packets arriving in a time-slot being $N$ with mean $\rho < 1$. Thus, the queue is under loaded. Let this queue start empty. At the end of any time slot $k$, let $Q_k$ denote the queue length. Let $\eta_k$ denote the number of packets arrived in time slot $k$. Then, the following is a standard Lindley style recursion.

$$Q_k = \max_{1 \le s \le k+1}\left\{\sum_s^k \eta_i - (k - s + 1)\right\}. \tag{28}$$

For completeness, we provide sketch of the proof for (28) as follows. The proof essentially is induction of the claim over $k$. For $k = 0$, since system starts empty the claim is trivially verified. Suppose the (28) is true till some $k \ge 0$. Now, we wish to establish it for $k+1$. The following equation is readily verifiable:

$$Q_{k+1} = (Q_k + \eta_{k+1} - 1)^+ \tag{29}$$

From (29) and the induction hypothesis we have

$$Q_{k+1} = \left(\max_{1 \le s \le k+1}\left\{\sum_s^k \eta_i - (k - s + 1)\right\} + \eta_{k+1} - 1\right)^+$$

But,

$$\begin{aligned}&\left(\max_{1\le s\le k+1}\left\{\sum_s^k \eta_i-(k-s+1)\right\}+\eta_{k+1}-1\right)^+\\&=\max\left\{\max_{1\le s\le k}\left\{\sum_s^{k+1}\eta_i-(k+2-s)\right\},0\right\}\\&=\max_{1\le s\le k+2}\left\{(k+1-s)-\sum_s^{k+1}\eta_i\right\}.\end{aligned}$$

The first equality follows from the fact that $\eta_{k+1}-1$ is independent of $s$ and the last equality follows because the expression $\sum_s^{k+1}\eta_i-(k-s+2)$ evaluates to zero for $s=k+2$. This completes the justification of (28).

Now, define $X(s,k)\triangleq\sum_s^k\eta_i-(k-s+1)$ for $1\le s\le k+1$. Since $\eta_i$'s are i.i.d. it follows that $X(s+1,k+1)$ has the same distribution as that of $X(s,k)$ for $1\le s\le k+1$. Define

$$Y_k=\max_{1\le s\le k+1}\{X(s,k)\};\quad Y_k^{'}=\max_{1\le s\le k+1}\{X(s+1,k+1)\}$$

$$\text{and}\quad Y_{k+1}=\max\{Y_k^{'},X(1,k+1)\}.$$

Now, $Y_k$ and $Y_k^{'}$ have the same distribution. Since $Y_{k+1}=\max\{Y_k^{'},X(1,k+1)\}$, $Y_{k+1}$ stochastically dominates $Y_k$ and $Y_k^{'}$. Now,

$$Q_k=Y_k\ \forall\ k\in\mathbb{N}.$$

Hence, $Q_k\preceq Q_{k+1}$ i.e., $Q_{k+1}$ stochastically dominates $Q_k$. Thus it follows that

$$\mathbb{E}[Q_k]\le\mathbb{E}[Q_{k+1}].$$

Recursing this we obtain that $\forall\ n\ge k$,

$$\mathbb{E}[Q_k]\le\mathbb{E}[Q_n].$$

Since the single FIFO queue defined above has arrival process which is mixture of $N$ Bernoulli i.i.d. processes (since the FIFO queue corresponds to an output queue in OQ switch), we have that the queue Markov process converges to a unique invariant distribution. Therefore, taking $n$ to $\infty$ we obtain

$$\mathbb{E}[Q_k]\le\mathbb{E}[Q_\infty]$$

Given the distributional assumptions on arrival and service processes, standard Large Deviation arguments will imply that [25], for $t$ large enough

$$\Pr(Q_\infty>t)\le A\exp(-Bt);\quad \text{for some } A,B>0$$

Using this, the union bound and the formula $\mathbb{E}[X]=\sum_t\Pr(X>t)$, we obtain that

$$\mathbb{E}\left[\max_{k\le M}Q_k\right]\le O(\log M)\tag{30}$$

## V. Experiments

We carried out simulations to compare the performance of our algorithm, MUCF($\mathcal{FIFO}$), with Longest Queue First (LQF) and Oldest Cell First (OCF). We used a fixed-length packet switch simulator available at http://klamath.stanford.edu/tools/SIM/.

We first explain the simulation settings: The switch size is $N=16$. The buffer sizes are infinite. The policy used is FIFO. All inputs are equally loaded on a normalized scale, and $\rho\in(0,1)$ denotes the normalized load. The arrival process is Bernoulli i.i.d. We use a *Uniform* load matrix, i.e., $\lambda_{ij}=\rho/N\ \forall i,j$. We ran our simulation for 2.1 million time steps removing the first 100,000 time steps to achieve steady-state.

Because we are approaching this problem from the perspective of fairness, we evaluate the aforementioned switching algorithms in terms of Latency and Output-Queue (OQ) Delay. OQ delay is defined as the difference of the departure times of a cell in the input queued switch and the shadow OQ switch. Further, the goal cannot only be to achieve a better expected latency, but in fact, we wish to value consistency, or relatively few deviations from the mean. One measure for this are higher moments of the variables. Thus, here we provide plots for the logarithm of first and second moments of both Latency and the OQ Delay versus a uniform load of $\rho$.

Figures 1 and 2 correspond to latency and figures 3 and 4 correspond to OQ delay. We observe that MUCF performs better than the other two algorithms for both the metrics at all the loads, especially for the second moments illustrating the fairness. Thus, the simulations illustrate that MUCF better tracks the performance of an OQ switch than LQF and OCF.

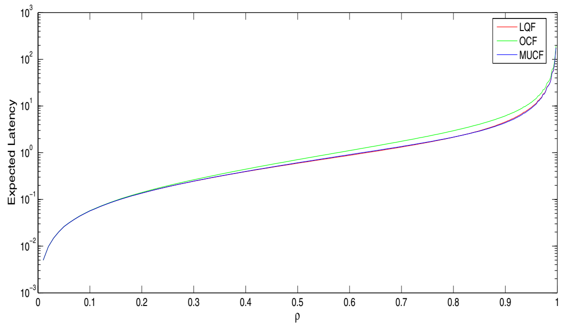

Fig. 1. Comparison of the logarithm of Expected latencies of different scheduling algorithms.

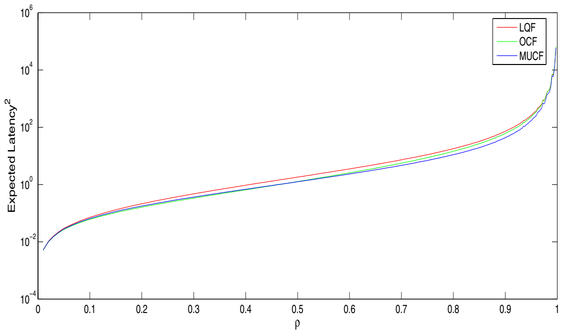

Fig. 2. Comparison of the logarithm of second moments of the latencies of different scheduling algorithms.

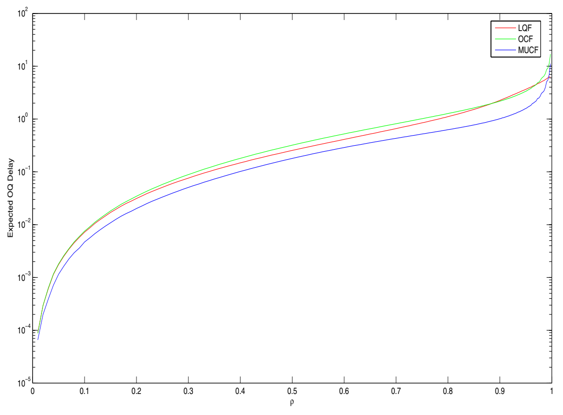

Fig. 3. Comparison of the logarithm of Expected output queued delays of different scheduling algorithms.

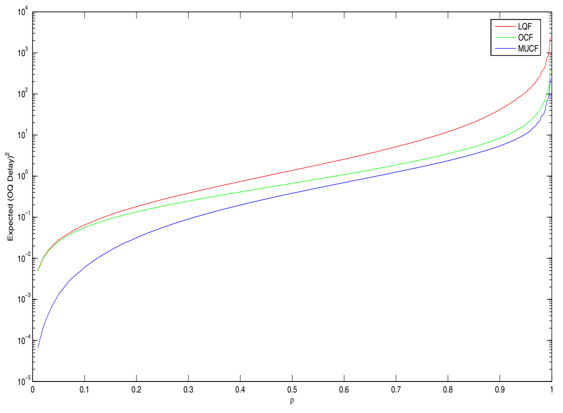

Fig. 4. Comparison of the logarithm of second moments of the output queued delays of different scheduling algorithms.

## VI. Conclusion

In this paper, we proposed a new notion of fair scheduling algorithm for constrained packet network such as input queued switch. We obtain such an algorithm through an equivalence between ranked election and fair scheduling. Our algorithm, though presented for input queued switch, can easily extend for any constrained packet network. We established that our algorithm is throughput maximal when it is derived based on a FIFO OQ switch. We strongly believe that our proof extends to an arbitrary work-conserving policy since the difference between urgencies for FIFO and any work conserving policy must be stochastically bounded for the reason that the second moment of busy cycle for any work-conserving single queue is bounded under friendly arrival process.

## References

[1] A. Demers, S. Keshav, and S. Shenker, "Analysis and simulation of a fair queueing algorithm," in *SIGCOMM '89: Symposium proceedings on Communications architectures & protocols*. New York, NY, USA: ACM, 1989, pp. 1–12.

[2] A. Parekh and R. Gallager, "A generalized processor sharing approach to flow control inintegrated services networks: the single-node case," *Networking, IEEE/ACM Transactions on*, vol. 1, no. 3, pp. 344–357, 1993.

[3] ——, "A generalized processor sharing approach to flow control inintegrated services networks: the multiple node case," *Networking, IEEE/ACM Transactions on*, vol. 2, no. 2, pp. 137–150, 1994.

[4] M. Shreedhar and G. Varghese, "Efficient fair queueing using deficit round-robin," *IEEE/ACM Transactions on Networking (TON)*, vol. 4, no. 3, pp. 375–385, 1996.

[5] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Trans. Netw.*, vol. 1, no. 4, pp. 397–413, 1993.

[6] R. Pan, B. Prabhakar, and K. Psounis, "Choke: a stateless aqm scheme for approximating fair bandwidth allocation," in *IEEE Infocom*, 2000.

[7] R. Pan, B. Prabhakar, L. Breslau, and S. Shenker, "Approximate fair allocation of link bandwidth," *IEEE Micro*, vol. 23, no. 1, pp. 36–43, 2003.

[8] B. Prabhakar and N. McKeown, "On the speedup required for combined input and output queued switching," *Automatica*, vol. 35, no. 12, pp. 1909–1920, 1999.

[9] S.-T. Chuang, A. Goel, N. McKeown, and B. Prabhakar, "Matching output queueing with a combined input output queued switch," *IEEE Journal on Selected Areas in Communications*, vol. 17, no. 6, pp. 1030–1039, 1999.

[10] X. Zhang and L. Bhuyan, "Deficit Round-Robin Scheduling for Input-Queued Switches," *IEEE Journal on Selected Areas in Communications*, vol. 21, no. 4, pp. 584–594, May 2003.

[11] D. P. Y. Yang, "Max-min fair bandwidth allocation algorithms for packet switches," *Parallel and Distributed Processing Symposium, 2007. IPDPS 2007. IEEE International*, pp. 1–10, 26-30 March 2007.

[12] M. Hosaagrahara and H. Sethu, "Max-Min Fairness in Input-Queued Switches," *ACM SIGCOMM poster session, Philadelphia, PA, USA, August*, 2005.

[13] F. Kelly, A. Maulloo, and D. Tan, "Rate control for communication networks: shadow prices, proportional fairness and stability," *Journal of the Operational Research Society*, vol. 49, no. 3, pp. 237–252, 1998.

[14] S. H. Low, "A duality model of tcp and queue management algorithms," *IEEE/ACM Trans. on Networking*, vol. 11, no. 4, pp. 525–536, 2003.

[15] M. Chiang, S. Low, A. Calderbank, and J. Doyle, "Layering as optimization decomposition: A mathematical theory of network architectures," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 255–312, Jan. 2007.

[16] R. Srikant, *The Mathematics of Internet Congestion Control*. Birkhäuser, 2004.

[17] T. Bonald and L. Massoulié, "Impact of fairness on Internet performance," *Proceedings of the 2001 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pp. 82–91, 2001.

[18] G. de Veciana, T. Konstantopoulos, and T. Lee, "Stability and performance analysis of networks supporting elastic services," *IEEE/ACM Transactions on Networking (TON)*, vol. 9, no. 1, pp. 2–14, 2001.

[19] L. A. Goodman and H. Markowitz, "Social welfare functions based on individual rankings," *The American Journal of Sociology*, vol. 58, no. 3, pp. 257–262, Nov. 1952.

[20] L. Tassiulas, "Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks," *IEEE Transactions on Automatic Control*, vol. 37, pp. 1936–1948, 1992.

[21] N. McKeown, A. Mekkittikul, V. Anantharam, and J. Walrand, "Achieving 100% throughput in an input-queued switch," *Communications, IEEE Transactions on*, vol. 47, no. 8, pp. 1260–1267, 1999.

[22] D. Shah and D. Wischik, "Optimal scheduling algorithms for input-queued switches," in *IEEE Infocom*, 2006.

[23] N. Kumar, R. Pan, and D. Shah, "Fair scheduling in input-queued switches under inadmissible traffic," in *IEEE Globecom*, 2004.

[24] K. Arrow, *Social Choice and Individual Values*. Yale University Press, 1951.

[25] A. Ganesh, N. O'Connell, and D. Wischik, *Big Queues*. Springer-Verlag, 2004.