Achieving Stability in Networks of IQ Switches Using a Local Online Scheduling Algorithm

Neha Kumar, Shubha U. Nabar
Department of Computer Science
Stanford University
{nehak, sunabar}@stanford.edu

Abstract— **In recent years, several high-throughput low-delay scheduling algorithms have been designed for input-queued (IQ) switches. It has been shown however that scheduling policies based on maximum weight matching (MWM) that perform optimally for an isolated switch are unable to ensure stability in a network of IQ switches [2].**

Although algorithms that ensure stability have been designed [2], [1], they are either not fully local or require prior knowledge of rates or excessive book-keeping for rate estimation, thus making them less desirable. In this report, we propose the use of a local and online switch-scheduling algorithm that incorporates fairness, and prove that it achieves stability in a network of single-server switches.

I. Introduction

A. Input-Queued Switches

The input-queued (IQ) switch architecture is widely used in high-speed switching. This is primarily due to its memory bandwidth requirements that are much lower than those of the output-queued and shared-memory architectures, making it the preferred choice.

In an $N \times N$ IQ switch, packets are queued up at the inputs. Cells arrive at input i for output j at an average rate λ_{ij}. In each time slot, at most one cell arrives at each input and at most one cell can be transferred to an output. The following holds for $\Lambda = [\lambda_{ij}]$ under admissibile traffic conditions:

$$\sum_{j=1}^{N} \lambda_{ij} < 1, \ \ \forall i \qquad \sum_{i=1}^{N} \lambda_{ij} < 1, \ \ \forall j$$

The switch-scheduling problem thus reduces to a matching problem in a weighted bipartite graph with N inputs and N outputs [1].

B. Prior Work

The performance of an IQ switch-scheduling algorithm is evaluated based on the throughput and delay it delivers. The Maximum Weight Matching (MWM) algorithm has been shown to be stable [2] under Bernoulli IID arrivals [12], [16], making it the ideal scheduling algorithm for IQ switches. Practical heuristics to approximate MWM [15], [8] too have been proposed. This result, like most other literature on IQ switches focuses on switches in isolation. Here we are interested in a *network* of IQ switches. It has been shown previously [2] that even under admissible traffic, a network of IQ switches implementing a MWM scheduling policy (under Longest-Queue-First) can exhibit unstable behavior. To counter this, the authors proposed the Longest-In-Network (LIN) policy that achieves 100% throughput under admissible traffic pattern falling under the leaky-bucket model. LIN relies on the exact knowledge of the traffic pattern at each switch however, making it an infeasible scheduling policy.

In [1], the authors proved the existence of stable algorithms which are local in nature but require either prior knowledge of arrival rates or excessive book-keeping to estimate them. These include Birkhoff-von-Neumann decomposition-based scheduling policies and algorithms such as Approximate Oldest-Cell-First that are MWM-based.

C. Report Outline

In [2], the authors present a counter-example on which MWM policies fail to achieve 100% throughput. This being a standard example used for proving instability, we discuss this first in section II, to provide an intuition for the inherent problem in achieving stability in the network-of-switches scenario.

In section III, we formalize the notion of fairness and propose fair algorithms for networks of switches. We then present our model for a network of single-server switches in section IV and prove that incorporating fairness ensures stability of the network, also providing simulation results in section V. In section VI, we consider networks of $N \times N$ IQ switches, followed by conclusions in section VII.

[1] The weight of the edge (i, j) is usually a measure of the level of congestion, e.g. the length of Q_{ij}, or the age of its oldest packet.

[2] A stable algorithm is one that ensures 100% throughput for admissible traffic.

II. Instability of MWM

An optimal algorithm for isolated switches under admissible traffic, MWM was first shown by Andrews & Zhang in [2] to be unstable in networks of switches. In their paper, they present an 8-switch network as a counterexample, and show that queue sizes grow unboundedly with progression of time.

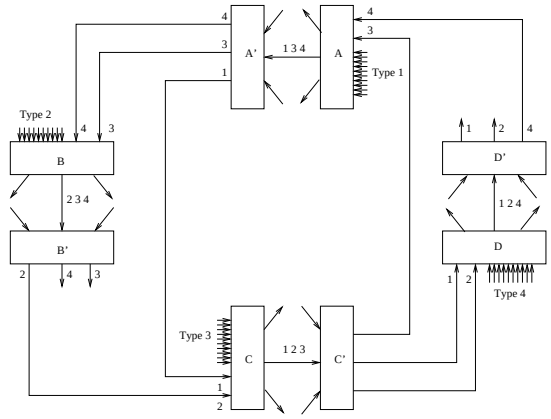

Fig. 1. A Network of 8 Switches

Their original construction is shown in Fig. 1. The network consists of four main switches A, B, C and D and four auxillary switches A', B', C' and D'. Each main switch has twelve input ports and one output port while each auxiliary switch has one input port and three output ports. Packets of four different types are injected into the system from the outside world. Type 1 packets enter the system at switch A and go through switches A', C, C', D and D' after which they leave the system. Similarly, packets of type 2 follow the path B, B', C, C', D and D'. Type 3 packets go through switches C, C', A, A', B and B' while type 4 packets go through switches D, D', A, A', B and B'. Packets of each type enter the first switch on their route via distinct input ports but subsequently, packets of the same type traverse an identical path of input/output ports in the network. The injection rate of each flow is $1/30$, ensuring an admissible traffic pattern for the network. Andrews & Zhang [2] employ a fluid type analysis to prove the instability of this network under MWM. It should be noted that MWM in this counterexample is equivalent to performing Longest Queue First (LQF) on the modified network in Fig. 2 where each switch is a single-server switch.

III. Fairness Incorporated

Our approach to counter instability in the above and in the more general scenario, is to incorporate fairness in LQF for the single-server switch case. We will first present this algorithm, then go on to prove that it does achieve stability in the network. We then present its counterpart for $N \times N$ switches - Fair-MWM - and conjecture that this too achieves stability.

A. Max-Min Fairness

Fairness is a subjective notion, so we begin by formalizing what we mean when we refer to a 'fair' algorithm. We now provide a precise definition based on the concept of *Max-Min fairness* [3].

Definition 1 (Max-Min Fairness) Let there be n flows arriving at a server of capacity C with rates $\lambda_1, \ldots, \lambda_n$ respectively. A rate allocation $r = (r_1, \ldots, r_n)$ is called Max-Min fair iff
(i) $\sum_n r_i \leq C,\ \ r_i \leq \lambda_i$, and
(ii) any r_i can be increased only by reducing r_j s.t. $r_j \leq r_i$.

Definition 2 (Fairness in a Switch) If every VOQ represents a flow, then there are N^2 flows in an $N \times N$ switch. We call $R = [r_{ij}]$ a fair allocation for $\Lambda = [\lambda_{ij}]$ iff it is Max-Min fair for every output.

B. Fair-LQF

Fair-LQF employs fairness to ensure stability in a network of *single-server* switches [10]. For any pre-specified threshold value, this algorithm has been shown to ensure a Max-Min fair allocation of rates on a single-switch in isolation when arriving traffic satisfies the Strong Law of Large Numbers (SLLN) [3]. Instead of using the notion of *congestion* defined by the authors in [10], we differentiate between the treatment we impart to the queues based on whether they are non-empty, i.e. threshold = 0. The pseudo-code follows:

```
//Information Collection
if (queue_i is non-empty)
    add_to_list(queue_i);
// Scheduling
// Step1: scheduling non-vacant ports
m = number-of-nonempty-queues;
while ( m != 0) {
  // Round-Robin
  schedule-nonempty-queues();
  m--;
}
```

Since we simply cycle through the non-empty queues, the algorithm is equivalent to producing a *non-idling round robin* schedule. The algorithm is work-conserving in nature and traffic is admissible, hence a simple Lyapunov analysis can show that it is stable on an isolated switch.

C. Fair-MWM

Now we present Fair-MWM for $N \times N$ switches [10]. When queue sizes are below a given threshold (in our case

[3] This law is stated in Section IV.

0), this algorithm schedules a maximum weight matching in every time slot using queue sizes for weights. Once a VOQ_{ij} that has crossed the threshold is served, it gets blocked [4] for n_j time slots, where n_j is the number of non-empty queues containing packets headed for output j and the remaining VOQs are scheduled using MWM. The pseudo-code for the algorithm follows.

```
//Information Collection
if (voq_ij is non-empty)
    add_to_list(voq_ij);

// Scheduling
// Step1: MWM
MWM_schedule_unblocked_voqs();

//Step2: Blocking information
for (i=0; i<N; i++) {
  for (j=0; j<N; j++) {
   if (voq[i][j] is matched and non-empty)
      cycles_to_block[i][j] =
     number-of-nonempty-voqs-to-output-j();

   else if (cycles_to_block[i][j] > 0)
      cycles_to_block[i][j]--;
  }
}
```

Fair-MWM differs from MWM mainly in its proactive way of limiting the throughput of a queue that has crossed the threshold. The key idea is to prevent it from competing for a maximum weight matching for more than its fair share of time slots, thus giving other queues a chance to be serviced. Note that Fair-LQF is a special $N \times 1$ case of Fair-MWM.

For a single switch in isolation, experiments show that Fair-MWM provides a max-min fair rate allocation for all the flows that pass through any given output port when arrivals are bernouilli i.i.d. For example, for the following arrival matrix:

$$\Lambda = \begin{bmatrix} 0.6 & 0.0 & 0.2 & 0.1 \\ 0.6 & 0.2 & 0.0 & 0.1 \\ 0.0 & 0.6 & 0.0 & 0.1 \\ 0.2 & 0.6 & 0.0 & 0.1 \end{bmatrix}$$

The MWM and Fair-MWM rate allocations for Λ are:

$$R_{mwm} = \begin{bmatrix} 0.47 & 0.0 & 0.2 & 0.1 \\ 0.47 & 0.06 & 0.0 & 0.1 \\ 0.0 & 0.47 & 0.0 & 0.1 \\ 0.06 & 0.47 & 0.0 & 0.1 \end{bmatrix} R_{mmf} = \begin{bmatrix} 0.4 & 0.0 & 0.2 & 0.1 \\ 0.4 & 0.2 & 0.0 & 0.1 \\ 0.0 & 0.4 & 0.0 & 0.1 \\ 0.2 & 0.4 & 0.0 & 0.1 \end{bmatrix}$$

Here, Fair-MWM performs a rate allocation that is Max-Min fair, i.e. $R_{f-mwm} = R_{mmf}$.

[4] Blocking VOQ_{ij} is equivalent to assigning it a weight of 0.

IV. Networks of Single-Server Switches

Our network consists of single-server switches, each of which makes its scheduling decisions locally using Fair-LQF. We first present our model, then proceed with our proof that Fair-LQF is stable, and revisit the 8-switch counter-example in [2] to show that Fair-LQF achieves stability in this specific case.

A. The Model

We now give a detailed description of the model that we assume, discussing the definition of flows, traffic conditions, and the notion of stability.

A.1 Flows

A flow is defined as a set of packets that traverse the same path along the network, sharing an ingress point and an egress point. We assume *per-flow queueing* in our model, which implies that at each physical queue, packets belonging to a particular flow are mapped to one logical queue. Hence, a sequence of different logical queues together constitutes a single physical queue. If C is the number of flows, at switch s, we have ${P_s}^2$ physical queues (VOQs) and up to $C{P_s}^2$ different logical queues.

We also assume *deterministic routing* from one switch to another. This is key because it imposes the condition that all logical queues dedicated to the same flow receive the same workload.

A.2 Arriving Traffic

Our assumptions for arrivals are minimal. The first criteria is that they must obey the Strong Law of Large Numbers (SLLN). The law can be stated as follows:

$$\lim_{n\to\infty} \frac{\Sigma_{j=1}^{n} A_i(j)}{n} \quad = \quad \lambda_i \; \forall i \;\; w.p.1 \qquad (1)$$

Here, $A_i(n)$ is the number of arrivals of type i packets from the outside world at time n.

We also impose the condition of admissibility on the arriving traffic, that is, if f_x denotes the set of flows passing through port x, then:

$$\sum_{i \in f_x} \lambda_i < 1 \;\; \forall x$$

It may be noted that these conditions are sufficiently generic in nature and allow much freedom in terms of the traffic scenarios to consider.

A.3 Stability

Intuitively speaking, stability implies that the total number of packets in the system remains bounded. Formally,

we say that a network of switches is *rate stable* if it satisfies the following criteria:

$$\lim_{n\to\infty} \frac{X_(n)}{n} = \lim_{n\to\infty} \frac{1}{n}\Sigma_i(A_i - D_i) = 0 \ \ w.p.1$$

Here, X_n represents the queue-lengths vector at time n and D_i and A_i are the departure and arrival vectors at time n, respectively. A system that is rate stable under any admissible traffic conditions is said to achieve *100% throughput.*

B. Fair-LQF is Stable

Theorem 1: A network of single-server switches with per-flow queuing, implementing Fair-LQF as a scheduling policy achieves 100% throughput whenever the ingress stochastic processes satisfy the SLLN (1) and are admissible.

Proof: Consider a switch S in the network. Let $A_i(n)$ be the number of arrivals of packets of type i at this switch in the nth time slot and let $D_i(n)$ be the number of departures of type i packets from S in the nth time slot. Let N be the total number of flows arriving at S. Then the following lemma holds.

Lemma 1: If $\lim_{n\to\infty} \Sigma_{k=1}^n A_i(k)/n = \lambda$ and $\lambda < 1/N$, then Fair-LQF ensures that $\lim_{n\to\infty} \Sigma_{k=1}^n D_i(k)/n$ exists and is λ regardless of the arrival patterns of other flows at S.

Proof: Since

$$\lim_{n\to\infty} \frac{\Sigma_{k=1}^n A_i(k)}{n} = \lambda < \frac{1}{N}$$

and

$$\Sigma_{k=1}^n D_i(k) \leq \Sigma_{k=1}^n A_i(k)$$

it follows that

$$\limsup_{n\to\infty} \frac{\Sigma_{k=1}^n D_i(k)}{n} \leq \lambda \qquad (2)$$

We now need to show that

$$\liminf_{n\to\infty} \frac{\Sigma_{k=1}^n D_i(k)}{n} \geq \lambda$$

and use the following claim and fact for the purpose.

Claim 1: There exists a sequence of positive integers $m_1 < m_2 < \ldots$ s.t. $\forall k, D_i(m_k + 1) = D_i(m_k + 2) = \cdots = D_i(m_k + N) = 0$

Proof: We prove the claim by contradiction. Suppose the claim is not true. Then $\exists M$ s.t. $\forall n > NM$, $D_i(n + 1) + D_i(n + 2) + \ldots + D_i(n + N) \geq 1$.

Now, for any $k > M$,

$$\begin{aligned}\frac{\Sigma_{n=1}^{kN} D_i(n)}{kN} &= \frac{\Sigma_{n=1}^{NM} D_i(n) + \Sigma_{n=NM+1}^{kN} D_i(n)}{kN} \\ &\geq \frac{k-M}{kN}\end{aligned}$$

As $k \to \infty$, $\frac{\Sigma_{n=1}^{kN} D_i(n)}{kN} \to \frac{1}{N} > \lambda$, contradicting (2). ■

Fact 1: Since Fair-LQF at threshold zero is equivalent to a non-idling round robin policy, a non-empty queue will be served at least once in N consecutive time slots. Thus we can say that if $D_i(n+1) = D_i(n+2) = \cdots = D_i(n+N) = 0$ then, $\Sigma_{r=1}^n D_i(r) = \Sigma_{r=1}^n A_i(r)$.

We now proceed to prove that $\liminf_{n\to\infty} \Sigma_{k=1}^n D_i(k)/n \geq \lambda$.

Claim 2: $\liminf_{n\to\infty} \Sigma_{k=1}^n D_i(k)/n \geq \lambda$

Proof: This claim is again proved by contradiction. Suppose the statement is not true. Then there exists a sequence of positive integers $n_1 < n_2 < n_3 \ldots$ and $\alpha < \lambda$ s.t.

$$\forall k \ \frac{\Sigma_{r=1}^{n_k} D_i(r)}{n_k} \leq \alpha$$

Let $0 < \epsilon < \frac{\lambda-\alpha}{2}$.
Since $\lim_{n\to\infty} \Sigma_{k=1}^n A_i(k)/n = \lambda, \exists\, W > 0$ s.t.

$$\forall n > W \ \frac{\Sigma_{k=1}^n A_i(k)}{n} > \lambda - \epsilon \qquad (3)$$

Now pick $m_{\tilde{k}} > W$ and $n_{\tilde{k}} > m_{\tilde{k}}$ also satisfying $n_{\tilde{k}} > 4/(\lambda - \alpha)$. Let $m_{\hat{k}}$ be the largest m_k that is smaller than $n_{\tilde{k}}$. Then the following fact holds:

Fact 2: Amongst $D_i(m_{\hat{k}+1}), D_i(m_{\hat{k}+2}), \ldots, D_i(n_{\tilde{k}})$ there is no block of N consecutive zeros. It now follows:

$$\begin{aligned}\alpha &\geq \frac{\Sigma_{r=1}^{n_{\tilde{k}}} D_i(r)}{n_{\tilde{k}}} \\ &= \frac{\Sigma_{r=1}^{m_{\hat{k}}} D_i(r) + \Sigma_{r=m_{\hat{k}}+1}^{n_{\tilde{k}}} D_i(r)}{n_{\tilde{k}}} \\ &= \frac{\Sigma_{r=1}^{m_{\hat{k}}} A_i(r) + \Sigma_{r=m_{\hat{k}}+1}^{n_{\tilde{k}}} D_i(r)}{n_{\tilde{k}}} && \text{using Fact1} \\ &> \frac{m_{\hat{k}}(\lambda-\epsilon) + \Sigma_{r=m_{\hat{k}}+1}^{n_{\tilde{k}}} D_i(r)}{n_{\tilde{k}}} && \text{using (3)} \\ &\geq \frac{m_{\hat{k}}(\lambda-\epsilon) + \lfloor \frac{n_{\tilde{k}} - m_{\hat{k}} - N}{N} \rfloor}{n_{\tilde{k}}} && \text{using Fact2} \\ &> \frac{m_{\hat{k}}(\lambda-\epsilon) + \frac{n_{\tilde{k}} - m_{\hat{k}} - N}{N} - 1}{n_{\tilde{k}}}\end{aligned}$$

$$
\begin{aligned}
&= \frac{m_{\hat{k}}(\lambda-\epsilon)+\frac{n_{\tilde{k}}-m_{\hat{k}}}{N}}{n_{\tilde{k}}} - \frac{2}{n_{\tilde{k}}} \\
&\geq \frac{m_{\hat{k}}(\lambda-\epsilon)+(n_{\tilde{k}}-m_{\hat{k}})(\lambda-\epsilon)}{n_{\tilde{k}}} - \frac{2}{n_{\tilde{k}}} \quad \because \lambda < 1/N \\
&= \lambda-\epsilon-\frac{2}{n_{\tilde{k}}} \\
&> \lambda - \frac{\lambda-\alpha}{2} - \frac{\lambda-\alpha}{2} \\
&= \alpha
\end{aligned}
$$

We thus arrive at a contradiction, proving hence that $\liminf_{n\to\infty} \Sigma_{k=1}^{n} D_i(k)/n \geq \lambda$.

Since $\liminf_{n\to\infty} \Sigma_{k=1}^{n} D_i(k)/n \geq \lambda$ and $\limsup_{n\to\infty} \Sigma_{k=1}^{n} D_i(k)/n \leq \lambda$, we have $\lim_{n\to\infty} \Sigma_{k=1}^{n} D_i(k)/n = \lambda$. ■

In general, if a switch has a service rate of μ and N flows arrive at its input ports, Lemma 1 can easily be extended to show that any flow satisfying the SLLN and asking for less than its fair share of μ/N is serviced at exactly its requesting rate regardless of other arrivals at the switch.

We now consider flows in decreasing order of their injection rates in to the system. Choose the flow i with the smallest arrival rate, λ_i. Let the flow pass through switches $S_1, S_2, \ldots, S_k$ in that order. Consider S_1. We know that $\lim_{n\to\infty} \Sigma_{j=1}^{n} A_i^{S_1}(j)/n = \lambda_i < 1/N$. The inequality follows from the fact that i is the smallest flow passing through switch S_1 and traffic would be inadmissible if the inequality were not satisfied. From Lemma 1, we know that $\lim_{n\to\infty} \Sigma_{j=1}^{n} D_i^{S_1}(j)/n = \lambda_i < 1/N$. Since the departures from S_1 form the arrivals at S_2 and so on, and at each of the switches i should be asking for less than its fair share because of the admissibility constraints, we can repeatedly apply Lemma 1 to show that i departs from the system at exactly its arrival rate λ_i. We remove this flow from consideration and reduce the service rates of $S_1, S_2, \ldots, S_k$ by λ_i. We then pick the next smallest flow and repeat. At every step, we remove one flow from the network and until there are no more flows left to consider, we can always find a new flow asking for less than its fair share at all the switches it passes through. Each flow gets to depart the system at exactly its arrival rate. Thus, it follows that our network is rate stable. ■

C. The 8-Switch Counter-Example

The counter-example in [2] can be reduced to a network of single server switches A, B, C and D, as shown in Fig. 2. Each switch has twelve input ports and a single ouptut port. Flow of type 1 passes through switches A, C and D in succession. Flow of type 2 passes through switches B, C and D. Type 3 packets flow through switches C, A and B while type 4 packets flow through switches D, A and B. We now analyze the performance of Fair-LQF on these single-server switches.

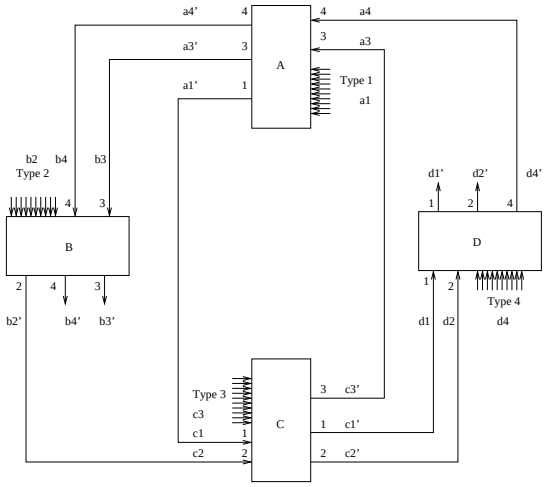

Fig. 2. A Network of 4 Switches with Rates

Let a_1, a_3, a_4 be the rates at which packets of type 1, 3 and 4 respectively enter switch A and a_1', a_3', a_4' be the rates at which they leave switch A. Similarly, we define $b_2, b_3, b_4, b_2', b_3', b_4', c_1, c_2, c_3, c_1', c_2', c_3', d_1, d_2, d_4, d_1', d_2'$ and d_4'. We first observe that if the queues are initially empty, all the flows leave the network at the same rate at which they are injected and the network is stable.

Consider flow of type 1. It enters switch A at rate $a_1 = 1/3$. This entering flow is actually split up over 10 input ports each of which receives packets at rate $1/30$. Each of these 10 input ports thus asks for a service rate that is less than its fair share of $1/12$ and therefore gets served at exactly its requesting rate. Thus $a_1' = a_1 = 1/3$. Similarly, $b_2 = b_2' = 1/3$, $c_3 = c_3' = 1/3$ and $d_4 = d_4' = 1/3$. Now consider switch A. Here $a_3 = c_3' = 1/3$ and $a_4 = d_4' = 1/3$. Under Fair-LQF each one of the input queues will get a service rate equal to its Max-Min fair share. Since the sum of the request rates at switch A is equal to 1 the Max-Min fair share for each queue equals its requesting rate. Thus $a_1' = a_3' = a_4' = 1/3$. Now consider switch B. Here $b_3 = a_3' = 1/3$ and $b_4 = a_4' = 1/3$. Also, type 2 packets arrive at each of their ten input ports at rate $1/30$ from the outside world. The sum of the arrival rates at switch B is thus equal to 1 and the queues will be served according to their Max-Min fair shares which will be exactly equal to their requesting rates. Thus $b_2' = b_3' = b_4' = 1/3$.

Packets 3 and 4 enter the network at rates $c_3 = 1/3$ and $d_4 = 1/3$ respectively and leave the network at rates $b_3' = 1/3$ and $b_4' = 1/3$ respectively. Packets of type 3 and 4 thus leave the system at precisely their rate of injection, and the queues that they pass through are therefore stable. We can similarly show that packets of type 1 and 2

do not queue up at any of the switches in the network. This concludes our application of Fair-LQF to the counter- example, proving its stability in this specific case as well.

V. Simulations

We ran simulations to verify the correctness of our intuition. The simulations were run on the configuration of switches in Fig. 1 with different initial queue sizes and different admissible injection rates. The results of the simulations for two different experiments are shown in Fig. 3 and Fig. 4. It is clear that under LQF the number of packets in the network grows without bound whereas under Fair-LQF the number of packets remains bounded.

Experiment 1: All queues are initially empty

	Flow 1	Flow 2	Flow 3	Flow 4
Injection Rate	0.33	0.318	0.318	0.326
Departure Rate (LQF)	0.275	0.277	0.277	0.285
Departure Rate (F-LQF)	0.329	0.317	0.317	0.325

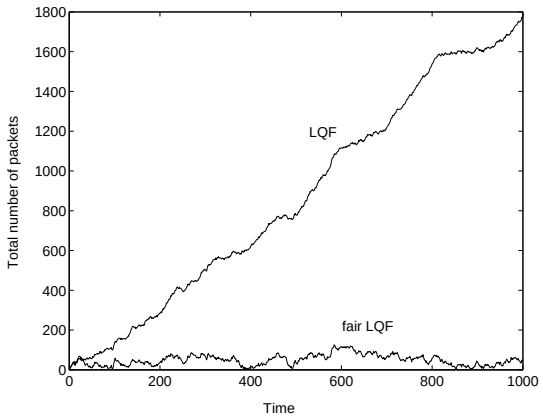

Fig. 3. LQF vs Fair-LQF: Comparing Total Packets in System

Experiment 2: Queue sizes are chosen randomly

	Flow 1	Flow 2	Flow 3	Flow 4
Injection Rate	0.743	0.124	0.125	0.122
Departure Rate (LQF)	0.744	0.128	0.091	0.097
Departure Rate (F-LQF)	0.723	0.138	0.126	0.125

VI. Stability in Networks of $N \times N$ Switches

We believe that the Fair-MWM algorithm (discussed earlier), when used for scheduling in networks of $N \times N$ IQ switches, results in network stability for all admissible arrival traffic obeying SLLN (1).

If we could prove a statement akin to that of Lemma 1 for Fair-MWM on a single switch in isolation with per-flow queuing at the VOQs, the stability of Fair-MWM in networks of IQ switches with per-flow queuing at the VOQs can be proved in a manner similar to Theorem 1.

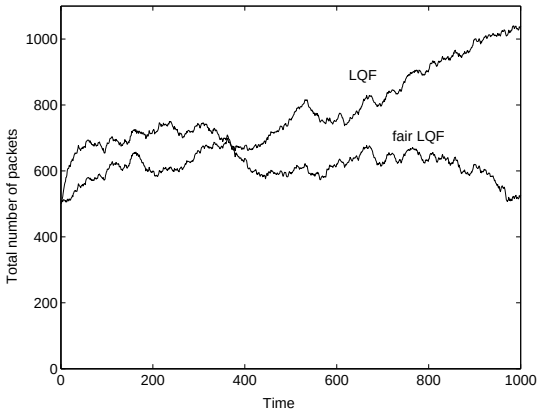

Fig. 4. LQF vs Fair-LQF: Comparing Total Packets in System with Arbitrary Initial Queue-Sizes

VII. Conclusions

We have shown that Fair-LQF is a stable online scheduling policy on networks of single server switches with per flow queuing when arrival rates satisfy the SLLN and traffic is admissible. We believe it can also be shown that Fair-MWM is a stable online scheduling policy for networks of $N \times N$ IQ switches with per-flow queuing at the VOQs when arrival rates satisfy the SLLN and traffic is admissible.

VIII. Acknowledgements

We thank Abtin Keshavarzian for introducing us to this challenging area of research and for encouraging and assisting us through the course of the project. We are also grateful to Mohsen Bayati for his mathematical expertise, without which our proof would have been incomplete.

References

[1] Ajmone Marsan, M., Giaccone, P., Leonardi, E., Neri, F., "On the Stability of Local Scheduling Policies in Networks of Packet Switches With Input Queues", *IEEE INFOCOM 2003*

[2] Andrews, M., Zhang, L., "Achieving Stability in Networks of Input-Queued Switches", *IEEE/ACM Trans. on Networking, vol. 11, no. 5*, 2003.

[3] Bertsekas, D., Gallager, R., "Data Networks" *Prentice Hall*, 1992, pp. 526

[4] Chang, C.S., Chen, W.J., Huang, H.Y., "Birkhoff-von Neumann input buffered crossbar switches", *IEEE INFOCOM 2000*, Mar. 2000, pp. 1614-1623

[5] Dai, J.G., "Stability of fluid and stochastic networks", Miscellenea Publication, No. 9, Center for Mathematical Physics and Stochastics, Denmark, `http://www.maphysto.dk`, January 1999.

[6] Dai, J.G., Prabhakar, B., "The Throughput of Data Switches with and without Speedup", *IEEE INFOCOM 2000*, Mar. 2000, pp. 556-564

[7] Demers, A., Keshav, S., Shenker, S., "Analysis and simulation of a fair queueing algorithm", *Journal of Internetworking Research*

and Experience, pp 3-26, Oct. 1990. Also in Proceedings of ACM SIGCOMM'89, pp 3-12.

[8] Giaccone, P., Prabhakar, B., Shah, D., "Switching under Energy Constraints", *Asilomar*, Oct. 2002.

[9] Karol, M., Hluchyj, M., Morgan, S., "Input versus output queueing on a space division switch", *IEEE Trans. on Communications*, vol. 35, n. 12, Dec. 1987, pp. 1347-1356

[10] Kumar, N., Pan, R., Shah, D., "Fair Scheduling in Input-Queued Switches under Inadmissible Traffic", *Submitted*

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

[12] McKeown, N., Mekkittikul, A., Anantharam, V., Walrand, J., "Achieving 100% throughput in an input-queued switch", *IEEE Trans. on Communications*, vol. 47, n. 8, Aug. 1999, pp. 1260-1267

[13] Shah, D., Kopikare, M., "Delay bounds for approximate Maximum weight matching algorithms for input-queued switches", *IEEE INFOCOM 2002*.

[14] Shah, D., Wischik, D., "Switch under Heavy Traffic", Under preparation, 2003.

[15] Tassiulas, L., "Linear complexity algorithms for maximum throughput in radio networks and input queued switches", *IEEE INFOCOM'98*, vol. 2, New York, 1998, pp. 533-539

[16] Tassiulas, L., Ephremides, A., "Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks", *IEEE Trans. on Automatic Control*, vol. 37, n. 12, Dec. 1992, pp. 1936-1948