Achieving Stability in Networks of Input-Queued Switches using a Local Online Scheduling Policy

Shubha U. Nabar, Neha Kumar, Mohsen Bayati and Abtin Keshavarzian
Departments of EE & CS at Stanford University
{sunabar, nehak, bayati, abtink}@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 such as Maximum Weight Matching, that perform optimally for an isolated switch, fail to provide stability in a network of IQ switches[2].**

Although there exist algorithms that ensure stability in networks of switches [2] [8], they are either not fully local or require knowledge/estimation of rates, and are thus not desirable. Here we propose a local and online switch-scheduling algorithm and prove that it achieves stability in a network of single-server switches when arriving traffic is admissible and obeys the Strong Law of Large Numbers. We then propose its counterpart for networks of crossbar switches and conjecture that this too is stable. Additionally, we prove that our algorithms provide a Max-Min fair rate allocation for isolated switches even when arriving traffic is inadmissible. We believe that fairness is key to ensuring stability in networks.

Keywords—**Switches & Switching, Queueing Theory, Network Stability.**

I. Introduction

The input-queued (IQ) switch architecture is widely used in high-speed switching, primarily due to its low memory bandwidth requirements. In an $N \times N$ IQ switch, cells arrive at input i for output j at an average rate λ_{ij} and are queued up in virtual output queue (VOQ) Q_{ij} [1]. In each time slot, at most one cell arrives at each input and at most one cell can be transferred to an output. The switch scheduling problem thus reduces to a matching problem in an $N \times N$ bipartite graph. The following holds for $\Lambda = [\lambda_{ij}]$ under admissible traffic conditions:

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

A. Background and Motivation

The performance of a switch-scheduling algorithm is evaluated on the throughput and delay it delivers. The Maximum Weight Matching (MWM) [1] algorithm has been shown to be stable [2] when arriving traffic is admissible and obeys the Strong Law of Large Numbers [3] (SLLN) [6], making it ideal for IQ switch-scheduling.

Interestingly however, scheduling policies that ensure stability in an isolated switch, fail to do so in networks of switches. Although in practice, switches necessarily exist in networks. It has been shown [2] that even under admissible traffic, a network of IQ switches implementing MWM (with queue sizes as weights) can exhibit unstable behavior. To counter this, the authors propose the Longest in Network (LIN) policy that achieves 100% throughput under admissible traffic satisfying leaky-bucket constraints. LIN's weakness is that it is frame-based and requires knowledge of the traffic pattern at every switch. In [8], the authors provide stable local scheduling policies that include Birkhoff-von Neumann decomposition-based policies and algorithms such as Approximate Oldest Cell First that are MWM-based. However, these require either prior knowledge of arrival rates or excessive book-keeping for rate estimation.

The problems above, motivate our search for a scheduling policy that is easily implemented, local in that it does not require knowledge-sharing across switches, *and* online so that scheduling decisions are a function of the present state alone.

B. Outline and Results

In [2], the authors specify a network of switches on which MWM policies fail to achieve 100% throughput. This counter-example provides intuition for the reason why MWM fails to provide stability in a network and is presented in section II.

We describe our model for a network of switches in section III and in section IV, propose a local and online scheduling algorithm - *Busy Round-Robin*. We prove that it is stable in a network of single-server switches, performing a fair rate allocation of service to incoming flows. We extend the algorithm to deal with crossbar IQ switches in section V, and propose *Collective Round-Robin*. We prove that this algorithm provides a fair allocation of service rates to flows traversing an isolated crossbar switch even when traffic is inadmissible and conjecture that it is stable in a network of IQ switches, providing experimental results in support. Conclusions follow in section VI.

II. Instability of MWM

In [2], Andrews & Zhang showed that MWM can be unstable in a network of switches, when the weights used are queue sizes. For this they used the 8-switch counter-example in Fig. 1, and proved that queue sizes grow unboundedly with time.

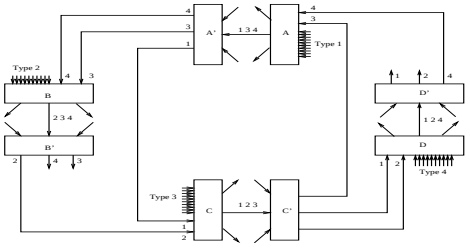

Fig. 1. The 8-Switch Counter-Example

There are four main switches A, B, C and D in this network

S. Nabar was supported by the Sequoia Capital Stanford Graduate Fellowship
M. Bayati was supported by Air Force Grant AF F49620-01-1-0365

[1] The weight of an 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.

[3] This law is defined in section III.

and four auxiliary switches A', B', C' and D'. Each main switch has 12 input ports and 1 output port while auxiliary switches have 1 input and 3 output ports. Four types of packets are injected into the system from the outside world. Type 1 packets enter the system at switch A and traverse A', C, C', D and D' before exiting the system. Similarly, type 2, 3 and 4 packets, each follow different routes through the network. Packets of one type enter the first switch on their route via distinct input ports but subsequently 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. The authors employ a fluid analysis to prove the instability of this network under MWM. Note that MWM in this counter-example is equivalent to performing Longest Queue First (LQF) on the modified network in Fig. 2 where each switch is a single-server switch.

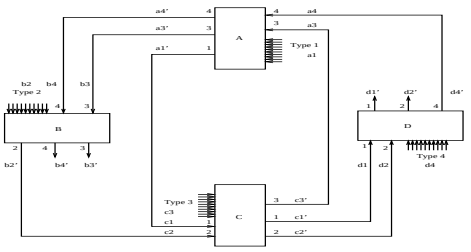

Fig. 2. Equivalent Representation of the 8-Switch Counter-Example

The main reason for LQF's failure in this network is that it rewards flows with high request rates while punishing well-behaved flows. Consider a situation where packets of type 1 and 2 are queued up at switches A and B respectively while all other queues in the system are empty. Now type 1 and 2 packets will each arrive at switch C at a rate of 1. As a result, type 3 packets will not be serviced for as long as the above queues are not emptied and will queue up at C. It can be shown that a stage is reached when the number of queued up type 3 packets exceeds the original number of type 1 and 2 packets in the system.

The key idea in our approach is to avoid such situations by incorporating fairness in the scheduling policy. We guarantee that any flow asking for less than its weighted fair share of service at a switch is granted its request rate regardless of the arrival pattern of other flows at that switch. All that is required is that the arrivals of this flow obey SLLN.

III. A Network of Switches

We now describe the assumptions made by our model for a network of switches. We then dichotomize our study into the single-server switch scenario and the crossbar switch scenario.

A. Switch Architecture

Single-Server Switches: A single-server switch is an IQ switch with N inputs, 1 output and a single server servicing the inputs. Each input port has one queue where packets are buffered and in each time slot, only one packet can be scheduled to leave the switch. Once packets reach the output they are immediately dispatched by a "dispatcher" along different paths to their next hops.

Crossbar Switches: A crossbar switch has N inputs, N outputs and N^2 VOQs. The input and output ports are connected by a crossbar fabric that enables the transfer of up to N packets from the inputs to the outputs in each time slot. At each output port, once again, a dispatcher dispatches the packets along different paths to their respective next hops.

B. Flows

A *flow* is defined as a set of packets that traverse the same path of switches along the network, passing through exactly the same input and output ports. They have a common ingress point from the outside world and leave the network through a common egress point. Our proofs in this paper can easily be extended to the case when flows are allowed to converge, however for ease of exposition, we assume that they do not.

We assume deterministic routing and *per-flow queueing* so that each queue in the system buffers packets of only one flow type. We also allow flows to be re-entrant, i.e. once a packet leaves a switch, it can re-enter the switch with the restriction that it either arrive at a different input port or be destined to a different output port. Thus cyclic paths with respect to the switches themselves are allowed, but not with respect to the VOQs.

C. Arriving Traffic

The first criterion for arrivals is that they obey the Strong Law of Large Numbers (SLLN) stated as follows:

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

Here, $A_i(n)$ is the number of type i packets injected into the system 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:

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

D. Stability

Intuitively, 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:

$$\lim_{n\to\infty} \frac{X_n}{n} = \lim_{n\to\infty} \frac{1}{n} \Sigma_{j=1}^{n} (A(j) - D(j)) = 0 \;\; w.p.1$$

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

IV. Networks of Single-Server Switches

We now present our local and online scheduling algorithm *Busy Round-Robin* ($\mathcal{BRR}$) and prove that it achieves stability in a network of single-server switches.

A. The $\mathcal{BRR}$ Algorithm

Given a set of input queues, $\mathcal{BRR}$ services non-empty queues in round-robin order skipping over empty ones. Note that $\mathcal{BRR}$ is a work conserving policy and thus is stable when applied to an isolated single-server switch under admissible traffic conditions. The pseudo-code for one scheduling time-step of $\mathcal{BRR}$ follows.

```
//rrpointer = round-robin pointer
for i = 1 to n //server does not examine a queue more than once
  if queue_rrpointer is non-empty
```

```
    schedule(queue_rrpointer++);
    break; //queue to be served has been determined
else rrpointer++;
```

This algorithm is easily implementable. Without knowledge of queue sizes, it guarantees stability in a network of single-server switches, as shown by the proof of the following theorem:

Theorem 1: A network of single-server switches with per-flow queuing, implementing $\mathcal{BRR}$ as a scheduling policy achieves 100% throughput whenever the ingress stochastic processes satisfy SLLN 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 $\mathcal{BRR}$ 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 of Lemma 1:

$$\begin{aligned} \lim_{n\to\infty} \frac{\Sigma_{k=1}^n A_i(k)}{n} &= \lambda < 1/N \quad \text{and} \\ \Sigma_{k=1}^n D_i(k) &\leq \Sigma_{k=1}^n A_i(k) \Rightarrow \\ \limsup_{n\to\infty} \frac{\Sigma_{k=1}^n D_i(k)}{n} &\leq \lambda. \end{aligned} \tag{1}$$

All that remains to be shown is

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

This is the main part of the proof of Lemma 1; its underlying ideas are summarized below and explained in greater detail in the proof of Claim 1. If the queue carrying flow i does not receive service in N consecutive time slots, it was necessarily empty at the beginning of this time frame. This implies that the total departures for flow i from this queue equalled the total arrivals until then. Using the fact that $\lambda_i < 1/N$, we show that the queue for flow i will be empty infinitely many times. It follows that $\Sigma_{k=1}^n D_i(k)/n$ approaches $\Sigma_{k=1}^n A_i(k)/n$ as n grows. The following claim completes the proof of Lemma 1.

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

Proof of Claim 1: This claim is proved by contradiction. Suppose the statement is not true. Then $\exists \alpha < \lambda$ s.t. $\liminf_{n\to\infty} \Sigma_{k=1}^n D_i(k)/n \leq \alpha$, i.e. there exists a sequence of positive integers $n_1 < n_2 < n_3 \cdots$ s.t.

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

We can choose ϵ to be small enough so that $0 < \epsilon < \frac{\lambda - \alpha}{3}$. Since $\lim_{n\to\infty} \Sigma_{k=1}^n A_i(k)/n = \lambda, \exists\ K > 0$ s.t.

$$\forall n > K \ \frac{\Sigma_{k=1}^n A_i(k)}{n} > \lambda - \epsilon. \tag{2}$$

Before finishing the proof of Claim 1, consider the following claim for departures:

Claim 2: There are infinitely many time slots in which $X_i = 0$ i.e there exists a sequence of positive integers $m_1 < m_2 < \cdots$ s.t. $\forall k, X_i(m_k) = 0$.

Proof of Claim 2: The proof is by contradiction. Suppose the claim is not true. Then $\exists M$ s.t. $\forall n > M$, $X_i(n) > 0$. But if queue i is non-empty then it gets served at least once every N time slots. Now for any $k > M$,

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

As $k \to \infty$, $\frac{k-M}{kN} \to \frac{1}{N} > \lambda$, so $\limsup_{n\to\infty} \Sigma_{k=1}^n D_i(k)/n > \lambda$, contradicting (1). □

Now pick $m_{k_1} > K$ and $n_{k_2} > \max\{m_{k_1}, 2/\epsilon\}$. Let m_{k_3} be the largest m_k smaller than n_{k_2}. Consider the following facts:

Fact 1: It follows from the definition of the sequence $\{m_k\}$ that during the time slots $m_{k_3}+1, \cdots, n_{k_2}$ flow i gets service at least once in every N time slots.

Fact 2: If $X_i(n) = 0$ then $\Sigma_{r=1}^n D_i(r) = \Sigma_{r=1}^n A_i(r)$.

It now follows:

$$\begin{aligned} \alpha + \epsilon &\geq \frac{\Sigma_{r=1}^{n_{k_2}} D_i(r)}{n_{k_2}} \\ &= \frac{\Sigma_{r=1}^{m_{k_3}} D_i(r) + \Sigma_{r=m_{k_3}+1}^{n_{k_2}} D_i(r)}{n_{k_2}} \\ &= \frac{\Sigma_{r=1}^{m_{k_3}} A_i(r) + \Sigma_{r=m_{k_3}+1}^{n_{k_2}} D_i(r)}{n_{k_2}} \\ &\geq \frac{m_{k_3}(\lambda-\epsilon) + \Sigma_{r=m_{k_3}+1}^{n_{k_2}} D_i(r)}{n_{k_2}} \\ &\geq \frac{m_{k_3}(\lambda-\epsilon) + \lfloor \frac{n_{k_2} - m_{k_3} - 1}{N} \rfloor}{n_{k_2}} \\ &\geq \frac{m_{k_3}(\lambda-\epsilon) + \frac{n_{k_2} - m_{k_3} - (N+1)}{N}}{n_{k_2}} \\ &> \frac{m_{k_3}(\lambda-\epsilon) + (n_{k_2} - m_{k_3})\lambda - 2}{n_{k_2}} \\ &\geq \frac{\lambda - \epsilon - 2}{n_{k_2}} \\ &\geq \lambda - 2\epsilon \\ &> \alpha + \epsilon. \end{aligned}$$

We 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 $\limsup_{n\to\infty} \Sigma_{k=1}^n D_i(k)/n \leq \lambda$, it follows that $\lim_{n\to\infty} \Sigma_{k=1}^n D_i(k)/n = \lambda$. □

In fact, the following general result can be shown:

Lemma 2: Consider a single-server switch running $\mathcal{BRR}$, arrival flows $A_1, \cdots, A_N$ and rates $\lambda_1 < \cdots < \lambda_N$. Let $k < N$ and $\lambda_{k+1} < (1 - \Sigma_{i=1}^k \lambda_i)/(N-k)$. If (i) $\limsup_{n\to\infty} \Sigma_{r=1}^n A_i(r)/n \leq \lambda_i\ \forall i$, (ii) the first k flows are

rate stable and (iii) arrivals for the $(k+1)$st flow satisfy SLLN, i.e. $\lim_{n\to\infty} \Sigma_{r=1}^{n} A_{k+1}(r)/n = \lambda_{k+1}$, then the $(k+1)$st flow is also rate stable.

The proof of Lemma 2 is similar to the proof of Lemma 1 and can be found in the full version of the paper [4].

We now consider flows in increasing order of injection rates into 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. For 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 its arrival rate λ_i.

We then pick the next smallest flow and can apply Lemma 2 to show that the flow exits every switch in the network and the network itself at its arriving rate. Repeating this process until there are no more flows left to consider, we show that our network is rate stable. ■

While the 8-switch counter-example from section II does not conform to our model due to the presence of convergent flows, we can easily extend our proof to handle the case of convergent flows as well and and the reader is referred to the full version of the paper for a proof showing that $\mathcal{BRR}$ is stable in the counter-example.

In [4], the authors show that a policy similar to $\mathcal{BRR}$ produces stability in wireline networks, but their results are limited to leaky-bucket constrained traffic whereas our result holds for more general traffic patterns. In addition to stability, $\mathcal{BRR}$ also guarantees fairness. We first formalize the notion of fairness.

B. Max-Min Fairness

The following notion of *Max-Min fairness* is well-established [3] and commonly used:

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

C. $\mathcal{BRR}$ is Max-Min Fair

It has been shown in [7] that when arriving traffic at an isolated single-server switch obeys SLLN but is inadmissible [5], LQF fails to perform a Max-Min fair rate allocation. We are able to prove that $\mathcal{BRR}$ on the contrary, is Max-Min fair under similar traffic conditions. The proof is omitted due to space constraints but can be found in the full version of the paper.

Theorem 2: When $\mathcal{BRR}$ is applied to a single server switch, the allocation of service rates to flows is Max-Min fair when arrivals satisfy SLLN . This holds even if arrivals are inadmissible.

We attribute the stability of $\mathcal{BRR}$ to its fairness. As was seen in section II, when traffic rates temporarily become inadmissible, load-balancing policies such as LQF worsen the situation by punishing well-behaved flows [6] and rewarding "bad" flows attempting to hog the bandwidth. $\mathcal{BRR}$, on the contrary, ensures that well-behaved flows get their request rates, preventing them from starvation.

D. Simulations

We performed simulations on the configuration of switches in Fig. 2 The results are shown in Fig. 3 It is clear that under LQF, the number of packets in the network grows without bound whereas under $\mathcal{BRR}$, 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 (BRR)	0.329	0.317	0.317	0.325

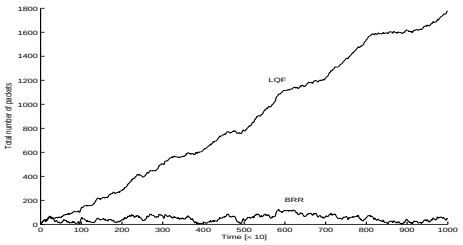

Fig. 3. LQF vs BRR: Comparing Total Packets in System

V. Networks of Crossbar Switches

Andrews & Zhang proved in their seminal paper [2] that MWM (where weights are queue sizes) fails to guarantee stability in a network of IQ switches. This happens because when a switch in the network becomes temporarily overloaded, well-behaved flows end up starving. It has been shown [7] that MWM does not perform a fair rate allocation when incoming traffic to an isolated IQ switch is inadmissible [5]. The definition of fairness in an IQ switch with virtual output queueing is given below.

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

To achieve fairness in rate allocations at an IQ switch, we propose *Collective Round-Robin* ($\mathcal{CRR}$) - a variation of an algorithm proposed in [5] for wireless network flows. While the results in [5] imply Max-Min fairness of $\mathcal{CRR}$ under leaky-bucket constrained, inadmissible traffic, we show that $\mathcal{CRR}$ is Max-Min fair for more general traffic satisfying SLLN .

A. The $\mathcal{CRR}$ Algorithm

As with single-server switches, we would like to view each output of a crossbar switch as an individual server that can service its flows in round-robin fashion. The challenge here is that two outputs might choose to serve VOQs at the same input simultaneously, when only one can actually be served due to the crossbar constraints. We get around this by introducing a *token* allocation policy similar to $\mathcal{BRR}$. Each output assigns service

[4] http://www.stanford.edu/∼sunabar/crr.pdf

[5] Inadmissibility in IQ switches arises only from overloading of output ports since admissibility at input ports is guarranteed by line rate constraints

[6] Well-behaved flows request at most their fair share of service.

tokens to its flows in round-robin fashion and at most one token is generated per output in one iteration of the algorithm. A token is only assigned to a VOQ if the queue has seen an arrival since the last time it was given a token. The tokens correspond to the service credit available to each flow, and at each time step the collection of non-conflicting flows with maximum service credit is selected for scheduling. When a flow is served, a token is removed from its credit. The pseudo-code for one scheduling time-step of the algorithm is given below. Note that for an $N \times 1$ switch, $\mathcal{CRR}$ is identical to $\mathcal{BRR}$.

```
//Token Generation
for i = 1 to N
  //rrpointer = round-robin pointer for output i
  for l = 1 to N // i does not examine a VOQ more than once
    if voq[rrpointer][i] has pkt not accounted for
       token[rrpointer++][i]++;
       break; //i has generated a token and stops
    else rrpointer++;

//Scheduling
//MWM with weights of voqs = # tokens
MWM_schedule_voqs();

//Token Reduction
for i = 1 to N
  for j = 1 to N
    if voq[i][j] is scheduled
    token[i][j]--;
```

We now show that $\mathcal{CRR}$ provides a Max-Min fair allocation of service rates to the flows passing through a switch.

B. $\mathcal{CRR}$ is Max-Min Fair

Theorem 3: When $\mathcal{CRR}$ is applied to a crossbar switch, the allocation of service rates to flows is Max-Min fair if arrivals satisfy SLLN. This holds even for inadmissible arriving traffic.

Proof: The packet transmission process is equivalent to the following: every time an output port generates a token for a VOQ, a packet is *released* from the VOQ for transmission. A released packet, however, is actually transmitted only when the corresponding flow is selected in the maximum weight matching of the flows. Since the token generation process can be viewed as though each output port were performing $\mathcal{BRR}$ on its flows, Theorem 2 implies that packets are released for transmission at a Max-Min fair rate and the packet release process is admissible. It follows [6] that MWM with the number of tokens/released packets as weights will allocate a Max-Min fair rate to the flows traversing the switch. ■

Simulations also attest to the fairness of $\mathcal{CRR}$. 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 $\mathcal{CRR}$ service rate allocations for Λ are:

$$R_{MWM} = \begin{bmatrix} 0.47 & 0.0 & 0.2 & 0.1 \\ 0.47 & 0.06 & 0.0 & 0.1 \\ 0.0 & 0.47 & 0.0 & 0.1 \\ 0.06 & 0.47 & 0.0 & 0.1 \end{bmatrix} R_{\mathcal{CRR}} = \begin{bmatrix} 0.4 & 0.0 & 0.2 & 0.1 \\ 0.4 & 0.2 & 0.0 & 0.1 \\ 0.0 & 0.4 & 0.0 & 0.1 \\ 0.2 & 0.4 & 0.0 & 0.1 \end{bmatrix}$$

Note that $\mathcal{CRR}$ performs a rate allocation that is Max-Min fair for every output. Following the same line of argument as with $\mathcal{BRR}$, we believe that because this algorithm is fair, it will ensure stability of the system:

Conjecture 1: A network of crossbar switches running $\mathcal{CRR}$ is stable whenever the arrivals to the network are admissible and satisfy SLLN .

C. Simulations

We performed many simulations on networks of 12×12 switches similar to the network in Fig. 1. In addition to the flows shown in Fig. 1, we inserted additional flows from the inputs to the unused output ports while maintaing admissibility, so that contention at input ports would result in more complex scheduling decisions than the ones faced by $\mathcal{BRR}$ and LQF. The results for one set of experiments are shown in Fig. 4. It is clear that under MWM the number of packets in the network grows without bound whereas queue sizes remain bounded under $\mathcal{CRR}$.

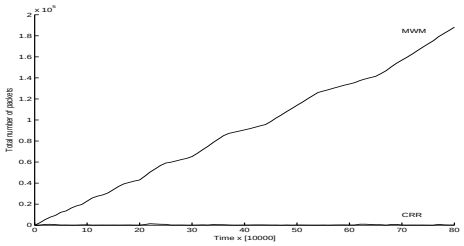

Fig. 4. MWM vs CRR: Comparing Total Packets in System

VI. Conclusions

In recent years, isolated input-queued switches have been the focus of much research. Since stable algorithms for IQ switches are unstable in networks, the study of stability in networks of switches merits consideration.

To tackle the problem of stability in single-server switches, we proposed *Busy Round-Robin* - an easy to implement, local and online scheduling policy. It guarantees a Max-Min fair rate allocation at every switch, preventing smaller flows from starvation due to larger flows. Our assumptions on arriving traffic were that it be admissible and satisfy the Strong Law of Large Numbers. An additional assumption in our model was per-flow queueing, which future work could seek to eliminate.

Since fairness guarantees that small flows are not discriminated against - a property inherently absent in load-balancing policies such as LQF - we believe this to be the solution to instability. Hence, we followed the same approach with crossbar IQ switches (satisfying similar model constraints) and proposed *Collective Round-Robin* - a local and online scheduling policy. Even under inadmissible traffic conditions, we proved that it provides a Max-Min fair rate allocation to incoming flows at an isolated switch, when arrivals satisfy the Strong Law of Large Numbers. The focus of future research in this area would be to prove stability of this algorithm in an arbitrary network of crossbar switches.

References

[1] Y. Tamir, G. L. Frazier, "High-performance multi-queue buffers for VLSI communications switches" in *Proceedings of ACM SIGARCH*, May 1988, vol. 16, no. 2, pp. 343-354.

[2] M. Andrews, L. Zhang, "Achieving Stability in Networks of Input-Queued Switches" in *Proceedings of IEEE INFOCOM*, Apr. 2001, pp. 1673-1679.

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

[4] E. Hahne, "Round-Robin Scheduling for Max-Min Fairness in Data Networks" in *Journal on Selected Areas in Communications*, Sep. 1991, pp. 1024-1039.

[5] L. Tassiulas, S. Sarkar, "Maxmin Fair Scheduling in Wireless Networks" in *Proceedings of IEEE INFOCOM*, 2002, pp. 763-772.

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

[7] N. Kumar, R. Pan, D. Shah, “Fair Scheduling in Input-Queued Switches under Inadmissible Traffic”, To appear in *IEEE GLOBECOM*, Dec. 2004.
[8] M. Ajmone Marsan, P. Giaccone, E. Leonardi, F. Neri, “Stability of Local Scheduling Policies in Networks of Packet Switches With Input Queues” in *Proceedings of IEEE INFOCOM*, Apr. 2003.