Backlog Aware Low Complexity Schedulers for Input Queued Packet Switches

Aditya Dua and Nicholas Bambos
Department of Electrical Engineering
Stanford University
350 Serra Mall, Stanford CA 94305
{dua,bambos}@stanford.edu

Wladek Olesinski, Hans Eberle, and Nils Gura
Sun Microsystems Laboratories
16 Network Circle, Menlo Park CA 94025
{wladek.olesinski,hans.eberle,nils.gura}@sun.com

Abstract

We study the problem of packet scheduling in input queued packet switches, with an emphasis on low complexity and ease of implementation. Toward this end, we propose a class of subset based schedulers, wherein an \( N \times N \) switch is operated using only a small set of \( N \) configurations in every time-slot. We show that the performance of subset based scheduling is comparable to that of the benchmark maximum weight matching (MWM) scheduler, albeit at much lower complexity. Next, we relate subset based scheduling to the well known wrapped wavefront arbiter (WWFA) \cite{tamir} and propose BA-WWFA, a backlog aware version of WWFA. The BA-WWFA scheduler significantly enhances the performance of WWFA, while retaining its ease of hardware implementation. The performance gains are especially noteworthy under non-uniform loading of the switch. Given their ease of implementation and MWM like performance, the schedulers proposed in this paper represent an attractive option for high performance packet switching.

1 Introduction

The input queued (IQ) switch architecture has been the subject of much attention in the networking research community. The popularity of the IQ architecture for high speed switching stems from its low memory bandwidth vis-à-vis the output queued (OQ) architecture, which makes it scalable and amenable to implementation. However, “smart” scheduling algorithms are needed to compensate for the low memory bandwidth and ensure efficient switch operation.

Scheduler design for IQ switches has been an active area of research over the past few years. See for instance \cite{dai} for an overview. A “good” scheduler is expected to ensure stability of the switch under all admissible loads (deliver 100% throughput), reduce packet latencies, arbitrate fairly between all ports, have low computational complexity, be robust to different traffic conditions, and be amenable to hardware implementation. Designing a scheduler with all the aforementioned properties is a non-trivial pursuit.

McKeown et al. \cite{mckeown}, and later Dai et al. \cite{dai} established that a scheduler which computes a maximum weight matching (MWM) in the bipartite graph constructed with the input and output ports as its two disjoint vertex sets delivers 100% throughput. The edge weights in the graph are chosen as the backlogs of the queues at the input ports. However, MWM has a high computational complexity of \( O(N^3) \) for an \( N \times N \) switch and is difficult to implement in hardware. Much work has been done on developing algorithms which can deliver 100% throughput at lower complexity. For instance, Giaccone et al. \cite{giaccone} proposed a suite of low complexity randomized algorithms, Ross et al. \cite{ross} proposed a class of local search algorithms based on geometric notions, and Deb et al. \cite{deb} developed an \( O(N^2) \) implementation of MWM based on LP duality theory.

On a different strand of research, several authors have proposed practical low complexity (typically sub-optimal) schedulers with a view toward easy implementation in hardware. Notable amongst these are PIM, proposed by Anderson et al. \cite{anderson} and wrapped wavefront arbiter (WWFA), proposed by Tamir et al. \cite{tamir}. While such schedulers perform well under uniform loading of the switch, their performance degrades significantly under non-uniform loads. Thus, a key challenge is to blend the simple implementation of schedulers like PIM and WWFA with the sophisticated decision making of schedulers like MWM to design efficient, low complexity schedulers.

In this paper, we specifically focus on the WWFA scheduler. WWFA, while very easy to implement in hardware, performs poorly in terms of throughput and packet latencies, especially under non-uniform loading of the switch. This can be attributed to the fact that WWFA is oblivious to backlogs of queues at the input ports. In contrast, most schedulers which deliver 100% throughput account for queue backlogs in their scheduling decisions. Thus, one of our motivations is to design a backlog aware version of
the basic WWFA algorithm, which retains the simplicity of WWFA, but yields better performance.

We initially approach the scheduling problem in a more general framework, wherein we construct a class of subset based schedulers. The key idea in subset based scheduling is to partition the set of possible switch configurations (of size $N^2$) into smaller subsets of size $N$ each and operate the switch using only one of these subsets in every time-slot. We show that when combined with a suitable subset selection rule, our subset based scheduler (MSB-PSS) delivers MWM like performance at significantly lower complexity.

Next, recognizing the equivalence between our constructed subsets and “wave patterns” used by WWFA, we propose the BA-WWFA scheduler, which is an enhanced backlog aware version of WWFA. BA-WWFA is as simple to implement as WWFA (with minimal additional hardware required), but delivers significantly better performance.

The remainder of this paper is organized as follows: The basic IQ switch model and the scheduling problem are described in Section 2. The notion of subset based scheduling is exposed in Section 3, and the MSB and MSB-PSS schedulers are proposed in Section 4. Subset based scheduling is combined with the well known WWFA scheduler to design BA-WWFA, a backlog aware version of WWFA. The performance of the proposed schedulers is evaluated experimentally in Section 5 and contrasted to the benchmarks WWFA and MWM. Finally, concluding remarks are furnished in Section 6.

2 Switch Model and the Scheduling Problem

2.1 The input queued switch

Consider an $N \times N$ IQ switch with $N$ input ports and $N$ output ports. There are $N$ virtual output queues (VOQs) at each input port to prevent head-of-line (HOL) blocking, as shown in Fig. 1. The $i^{th}$ VOQ buffers packets destined from input port $[(i-1)/N]+1$ to output port $(i-1) \mod N + 1$ and is denoted $Q_i$. The switch operates in slotted time and is governed by the crossbar constraints, i.e., every input (output) port can be connected to at most one output (input) port in a time-slot. An $N \times N$ switch can be set into $N!$ possible configurations. Each configuration is associated with a unique configuration vector of length $N^2$. Let $v_i = (v_{i,1}, v_{i,2}, \ldots, v_{i,N^2}) \in V$ denote the $i^{th}$ configuration vector, where $V$ is the set of all possible configuration vectors. Then, $v_{i,j} = 1$ if $Q_j$ is served when the switch is set in configuration $v_i$ and $v_{i,j} = 0$ otherwise. For example, the two possible configuration vectors for a $2 \times 2$ switch are $v_1 = (1 \ 0 \ 0 \ 1)$ and $v_2 = (0 \ 1 \ 1 \ 0)$. It is the task of the scheduler to select a switch configuration in every time-slot.

Packets arrive stochastically to $Q_i$ at an average rate of $\lambda_i$ packets per time-slot. In this paper, we focus on the case where packet arrivals are governed by an i.i.d. Bernoulli process, i.e., in every time-slot, a packet arrives to $Q_i$ with probability (w.p.) $\lambda_i$, independent of all other arrivals. We denote by $\lambda = (\lambda_1, \ldots, \lambda_N)$ the vector of all arrival rates and call it the load vector for the switch. We require that $\lambda$ be admissible, i.e., $\lambda \in \Lambda$ where

$$\Lambda = \{ \lambda : \sum_{j=1}^{N} \lambda_{(i-1)N+j} < 1, \sum_{j=1}^{N} \lambda_{(j-1)N+i} < 1, \quad i = 1, \ldots, N, \lambda_i \in [0, 1], \quad i = 1, \ldots, N^2 \}.$$  

(1) is a manifestation of the crossbar constraints and requires that no input/output port is oversubscribed.

Let $q_i(t)$ denote the backlog of $Q_i$, and let $q(t) = (q_1(t), \ldots, q_{N^2}(t))$ denote the composite backlog vector at the beginning of time-slot $t$. If a packet arrives at $Q_i$ (w.p. $\lambda_i$) in time-slot $t$ (denoted $a_i(t) = 1$), its backlog increases by 1. Further, if configuration $v_j$ is chosen by the scheduler in time-slot $t$ and $v_{ji} = 1$ (configuration $v_j$ serves $Q_i$), the backlog of $Q_i$ decreases by 1 (if $Q_i$ is non-empty). Thus, given arrival vector $a(t) = (a_1(t), \ldots, a_{N^2}(t))$ and switch configuration $v(t)$ in time-slot $t$, the switch dynamics are succinctly described by $q(t+1) = (q(t) + a(t) - v(t))^+$, where $x^+ = \max(x, 0)$, elementwise, for vector $x$.

2.2 The scheduling problem and MWM

Broadly speaking, the objective of the scheduler is to select switch configurations from set $V$ in every time-slot to ensure efficient operation of the switch. More specifically, we want the scheduler to possess the following desirable properties:

1. Ensures small average backlogs for all VOQs, which by Little’s Law [16] translates to low packet latencies.
2. Treats all VOQs “fairly”.

3. Has low computational complexity and is amenable to efficient hardware implementation.

4. Is agnostic to input traffic statistics, since they may not be readily available at the time of decision making.

5. Can keep all VOQs stable (finite average backlog) under any load vector \( \lambda \in \Lambda \). Such a scheduler is said to be throughput optimal, or equivalently, is said to deliver 100% throughput.

A benchmark scheduler which satisfies all but the third requirement and has been thoroughly studied in the literature is the maximum weight matching (MWM) scheduler. The MWM scheduler selects configuration \( v^*(t) \) in time-slot \( t \) such that

\[
v^*(t) = \arg \max_{v \in \mathcal{V}} \langle q(t), v \rangle,
\]

where \( \langle x, y \rangle \) denotes the inner-product between vectors \( x \) and \( y \). In other words, MWM finds the configuration vector with the largest projection on the current backlog vector. However, the best known algorithm for MWM has a complexity of \( O(N^3) \) per time-slot. Moreover, MWM is not amenable to efficient hardware implementation. Hence arises the need for a scheduling algorithm which can deliver MWM like performance at low complexity and is easy to implement.

3 Subset Based Scheduling

With a view toward complexity reduction, we now describe the notion of subset based scheduling [5]. The key idea is to partition the configuration set \( \mathcal{V} \) of size \( N! \) into smaller subsets of size \( N \) each and operate the switch using configurations from only one of these subsets in every time-slot.

3.1 Subset construction

It follows from the crossbar constraints that all configuration vectors are of the form \( v = [e_{\pi(1)} \ e_{\pi(2)} \ldots e_{\pi(N)}] \), where \( \pi \) is a permutation of \( \{1, 2, \ldots, N\} \) and \( e_i \) denotes the \( i^{th} \) standard unit vector in \( \mathbb{R}^N \). Now define the circular shift operator \( \mathcal{C} \) by

\[
\mathcal{C}(v) \triangleq [e_{\pi(N)} \ e_{\pi(1)} \ e_{\pi(2)} \ldots e_{\pi(N-1)}].
\]

Recursively define \( \mathcal{C}^k(v) \triangleq \mathcal{C}(\mathcal{C}^{k-1}(v)) \), \( k \in \mathbb{N} \), which corresponds to applying the circular shift operator \( k \) times to \( v \). By convention, \( \mathcal{C}^0(v) = v \). Also, note that \( \mathcal{C}^k(v) = \mathcal{C}^k \mod N(v) \). Thus, starting with any configuration vector \( v \in \mathcal{V} \), we can generate a set of \( N \) distinct configuration vectors by applying \( \mathcal{C} \) to \( v \) \( N - 1 \) times. We say that \( v \) generates the configuration subset \( \mathcal{S}_v = \{v, \mathcal{C}(v), \ldots, \mathcal{C}^{N-1}(v)\} \subset \mathcal{V} \) and refer to \( v \) as the generator vector. Following the outlined procedure, we can partition \( \mathcal{V} \) into \((N - 1)!\) disjoint configuration subsets of size \( N \) each. For illustration, the configuration subsets for \( N = 3 \) switch are depicted in Fig. 2.

By virtue of our construction, it follows that every VOQ is associated with exactly one configuration in every subset. This property is mathematically captured through the relation

\[
\sum_{j=0}^{N-1} \mathcal{C}^j(v) = 1 \quad \forall \ v \in \mathcal{V},
\]

where \( 1 \) is the all-ones vector.

3.2 Subset based scheduling — MSB

Consider operating the switch using only one configuration subset, say \( \mathcal{S}_v = \{\mathcal{C}^0(v), \mathcal{C}^1(v), \ldots, \mathcal{C}^{N-1}(v)\} \), viz. the subset generated by \( v \). Thus, the switch can be set in only one of \( N \) configurations chosen from \( \mathcal{S}_v \) in every time-slot, rather than one of \( N! \) configurations chosen from \( \mathcal{V} \).

Different algorithms can be employed for selecting one out of \( N \) configurations from \( \mathcal{S}_v \). We specifically consider the maximum weight (MW) family of algorithms. An MW type algorithm selects the configuration with the largest weight in every time-slot. In particular, suppose configuration \( \mathcal{C}^i(v) \in \mathcal{S}_v \) serves VOQs indexed by set \( \mathcal{T}_t = \{j_1^t, \ldots, j_N^t\} \). Then, the weight of configuration \( \mathcal{C}^i(v) \) in time-slot \( t \), namely \( w_i(t) \), is determined by a mapping \( \Gamma : \mathbb{Z}_+^N \rightarrow \mathbb{Z}_+ \), such that \( w_i(t) = \Gamma(q_{j_1^t}(t), \ldots, q_{j_N^t}(t)) \).

Different members of the MW family are generated by varying the mapping \( \Gamma \). We will primarily focus on the following choice for \( \Gamma \) in this paper:

\[
\Gamma(x) = \sum_{i=1}^{N} x_i: \text{ The weight of a configuration is the sum of the backlogs of all VOQs served by the configuration. Thus, the scheduler follows the maximum sum backlog rule (MSB), i.e., it selects configuration } v^*(t) = \mathcal{C}^i(t)(v) \text{ in time-slot } t, \text{ where }
\]

\[
i^*(t) = \arg \max_{i=0, \ldots, N-1} \langle q(t), \mathcal{C}^i(v) \rangle.
\]

Another interesting choice for \( \Gamma \) is \( \Gamma(x) = \max_{i=1, \ldots, N} x_i \).

Remark: MSB can be implemented efficiently using simple addition and comparison logic. Further, the implementation is parallelizable, since the weights of each of the \( N \) configurations can be computed in parallel. The resultant complexity is only \( O(N) \) per time-slot.
3.3 How good is MSB?

Not surprisingly, the simplicity of MSB comes at a cost — loss of throughput. This was expected since MSB operates the switch using only \( N \) out of a rich set of \( N! \) possible configurations. Thus, MSB cannot ensure switch stability for all admissible load vectors \( \lambda \in \Lambda \).

Nevertheless, we can quantify the set of load vectors for which the switch is stable under MSB. To this end, we say that \( \lambda \) is covered by configuration subset \( S_v \) if

\[
\exists \alpha_0, \ldots, \alpha_{N-1} \in (0, 1), \quad \sum_{i=0}^{N-1} \alpha_i = \alpha < 1 \text{ such that } \lambda \leq \sum_{i=0}^{N-1} \alpha_i C^i(v), \text{ elementwise.}
\]

In particular, consider the uniform load vector \( \lambda = \frac{\lambda}{N} \mathbf{1} \) for some \( \lambda \in (0, 1) \). By setting \( \alpha_i = \lambda/N \forall i \) and invoking (4), we see that all uniform load vectors are covered by configuration subset \( S_v \) (for any \( v \)). We now have the following important result:

**Theorem 1.** The switch is stable under MSB (when it uses subset \( S_v \)) for any load vector \( \lambda \) which is covered by \( S_v \). In particular, for any such load, \( \limsup_{t \to \infty} E[q_i(t)] < \infty \forall i \).

**Sketch of Proof:** For the quadratic Lyapunov function \( \mathcal{L}(q(t)) = (q(t), q(t)) \), we establish that

\[
\mathbb{E}[\mathcal{L}(q(t+1)) - \mathcal{L}(q(t)) | q(t) = q_i] \leq -\epsilon(q, 1) + C, \quad (6)
\]

for constants \( \epsilon > 0, C > 0 \) (which only depend on \( \lambda, N \)) and use standard arguments from [8] to conclude stability. The property in (4) plays a key role in establishing (6). We refer the reader to [6] for details, where a similar result is proved in a different context. \( \square \)

**Corollary 1.** The switch is stable under MSB for all uniform admissible loads \( \lambda = \frac{\lambda}{N}, \lambda \in (0, 1) \), irrespective of the choice of configuration subset \( S_v \).

Hence, we have established that operating the switch under the MSB policy (using only one subset with \( N \) configurations) is “good enough” (from a throughput perspective) when the switch is uniformly loaded, regardless of the choice of operational subset. If the switch is non-uniformly loaded, it may be possible to ensure stability (but not guaranteed!) if there exists a subset which covers the load vector.

3.4 Periodic subset selection and MSB-PSS

Can the loss of throughput under MSB be compensated for without sacrificing its low complexity? The answer is yes, and the goal is accomplished by using an appropriate subset selection rule. With subset selection, the scheduler still selects configurations from only subset of size \( N \) in every time-slot; however, the operational subset is chosen dynamically.

**Remark:** There is a simple randomized subset selection rule, which in conjunction with MSB delivers 100% throughput. However, the probability distribution which governs the randomization is derived from the Birkhoff von Neumann decomposition of the load vector \( \lambda \) [2], which contradicts our goal of designing schedulers agnostic to traffic statistics.

An alternative to the randomized rule is the periodic subset selection rule (PSS), which selects a new configuration subset every \( P \) time-slots. PSS works as follows: Suppose the switch is currently being operated using configurations from the subset \( S_v \). Compute the switch configuration \( v^\star \) based on MWM. If \( v^\star \notin S_v \), start operating the switch using \( S_v \). If \( v^\star \notin S_v \), start operating the switch using the configuration subset generated by \( v^\star \), namely \( S_v^\star = \{C^i(v^\star)\}_{i=0}^{N-1} \).

We now combine MSB with PSS to propose the MSB-PSS scheduling algorithm, which makes scheduling decisions based on MSB, but periodically updates its operational subset based on PSS. Since the complexity of
MWM is $O(N^3)$ per time-slot and the complexity of MSB is $O(N)$ per time-slot, the complexity of MSB-PSS is $O(N + N^3/P)$, which is $O(N^2)$ for $P = N$ and $O(N)$ for $P = N^2$. Further, MSB-PSS delivers 100% throughput. More formally,

**Theorem 2.** The switch is stable under MSB-PSS for any load vector $\lambda \in \Lambda$, i.e., $\limsup_{t \to \infty} \mathbb{E}[q_i(t)] < \infty \forall i$

**Sketch of Proof:** Once again, consider the quadratic Lyapunov function $L(q(t)) = \langle q(t), q(t) \rangle$ and let

$$
\delta_L \triangleq \mathbb{E}[L(q(t + 1) - L(q(t)))q(t) = q] = \sum_{p=0}^{P} \mathbb{E}[L(q(t + p + 1) - L(q(t + p)))q(t) = q].
$$

We bound each of the terms in the sum individually to establish

$$
\delta_L \leq -\epsilon'(q, 1) + C',
$$

for constants $\epsilon', C' > 0$ (which depend only on $\lambda, N, P$) and conclude stability (similar to the proof of Theorem 1). We refer the reader to [6] for details, where a similar result is proved in a different context.

In summary, we have shown that by combining subset based scheduling (MSB) with periodic subset selection (PSS), the switch can be stabilized under any admissible load at only $O(N)$ complexity (compared to $O(N^3)$ for MWM). However, a small penalty is paid in terms of average packet latencies, as we will see in the experimental results in Section 5.

### 4 Subset based scheduling and WWFA

#### 4.1 The WWFA algorithm

In the wrapped wavefront arbiter (WWFA) [14], each VOQ is assigned a request flag, which is set to 1 in the current time-slot if the VOQ is non-empty, and set to 0 else. WWFA first examines the VOQs “swept” by wave 1, as depicted in Fig. 3. In particular, for a $4 \times 4$ switch, wave 1 sweeps VOQs 1, 8, 11, and 14. Note that these VOQs are non-conflicting, i.e., they can all be scheduled simultaneously. A grant is issued to a VOQ swept by wave 1 if it has a request. Next, WWFA examines VOQs swept by wave 2 (indexed 2, 5, 12, 15 for a $4 \times 4$ switch) and issues them grants if they are non-empty, while satisfying the crossbar constraints. The process continues until all 4 waves have been processed. All VOQs to which grants were issued are then scheduled. This procedure, called wavefront propagation, is repeated in every time-slot.

WWFA is attractive due to its simplicity, which stems from the choice of wave shapes. For the chosen wave shapes, each VOQ can make a “local” grant decision based solely on the status of its own request flag and the outcome of the previous wave processing (communicated to it by neighboring VOQs from the left and the top). This local decision making ability vastly simplifies hardware implementation of WWFA.

On the downside, WWFA does not incorporate VOQ backlogs into its scheduling decisions. This results in performance degradation, especially under non-uniform loading of the switch.

#### 4.2 Wave patterns and configuration subsets

Observe that each step of the wavefront propagation procedure in WWFA is equivalent to applying the circular shift operator $C$, which was used to generate configuration subsets in Section 3.1. Each wave in WWFA corresponds to a unique configuration in a particular configuration subset. For our $4 \times 4$ example, wave 1 corresponds to the configuration $v_1 = [e_1 \ e_4 \ e_3 \ e_2]$, wave 2 corresponds to the configuration $v_2 = [e_2 \ e_1 \ e_4 \ e_3] = C(v_1)$, and so on, as illustrated in Fig. 3.

More generally, for an $N \times N$ switch, the $i^{th}$ wave corresponds to configuration $C^{i-1}(v_W)$, where $v_W = [e_1 \ e_N \ldots e_2]$. We will use $S_W = \{C(v_W)\}_{i=0}^{N-1}$ to denote the set of $N$ configurations generated by $v_W$. These are the basic $N$ configurations/waves used by the WWFA scheduler.

#### 4.3 Backlog aware WWFA — BA-WWFA

The equivalence between waves and configurations suggests similarities between MSB and WWFA. A similar analogy can also be drawn between MSB and the 2DRR scheduler proposed by LaMaire et al. [9], as well as between MSB-PSS and the enhanced version of 2DRR proposed in the same paper. However, two key differences between MSB and WWFA must be noted:

1. WWFA produces a maximal matching. Thus, the matching produced by WWFA need not correspond to any of the configurations in $S_W$. On the other hand, MSB finds the configuration with the maximum weight contained in $S_W$. Clearly, such a matching need not be maximal.

2. WWFA does not account for VOQ backlogs, while MSB does.

In the above, (1) highlights a desirable feature of WWFA (maximal matching), while (2) highlights a desirable feature
Wavefront propagation

Figure 3. The left side shows the wave pattern used by WWFA for a $4 \times 4$ IQ switch (implemented as a crossbar). The $j^{th}$ crosspoint in the $i^{th}$ row maps to VOQ index $N(i-1)+j$. The right side shows the configuration subset $S_W$ and also the one-one correspondence between waves and configurations.

of MSB (backlog awareness). We combine both features to propose **backlog aware WWFA** (BA-WWFA):

- Use MSB to select a configuration $v(t) \in S_W$.
- Execute WWFA, using the wave corresponding to $v(t)$ (recall the one-one correspondence between waves and configurations) as the first wave in the wave pattern.

By construction, BA-WWFA produces a maximal configuration which is “bigger” and “heavier” than the configuration selected by either WWFA or MSB alone. It follows from Theorem 1:

**Corollary 2.** The switch is stable under BA-WWFA for any load vector $\lambda$ which is covered by $S_W$.

In particular, the switch is stable under BA-WWFA for all admissible uniform loads, i.e, $\lambda = \frac{\lambda}{N}$, $\lambda \in (0, 1)$.

4.4 **BA-WWFA is easy to implement**

An important extension of Corollary 2 (with proof similar to Theorem 1) is that the switch is stable under BA-WWFA for all load vectors covered by $S_W$ even if old backlog information, namely $q(t-\tau)$ for $\tau > 0$, rather than current backlog information $q(t)$ is used in time-slot $t$ to execute MSB.

By virtue of the above, BA-WWFA can retain the simple implementation of WWFA, with the added advantage of backlog awareness as follows: In time-slot $t$, BA-WWFA selects the wave with the largest weight, which is computed based on $q(t-1)$. The wavefront propagation procedure is then initiated with the wave with the largest weight. While the waves are being processed, their current weight is computed and stored for use in the next time-slot. Minimal additional hardware is needed to compute and store the weight of each wave.

BA-WWFA uses the same set of $N$ waves/configurations as WWFA (in particular, those in the set $S_W$ of Section 4.2). Thus, BW-WWFA retains the local decision making (via neighbor communication) property of WWFA, and hence also its ease of hardware implementation. The implementation can be further improved by exploiting parallelization, as proposed by Olesinski et al. in [12] for basic WWFA.

5 **Performance Evaluation**

In this section, we experimentally contrast the proposed schedulers MSB, MSB-PSS, and BA-WWFA to the benchmark schedulers WWFA and MWM. For WWFA, the wave priorities were rotated every time-slot in the interest of fairness [14]. We did not explore the effect of using more sophisticated methods to ensure fairness, like the one proposed by Tian et al. in [15]. All results presented here are for a $16 \times 16$ switch. We employ *long run average delay per packet* (averaged over all VOQs) as a performance metric for comparing different schedulers. Every data point reported here is based on a simulation length of 30,000 time-slots. Packets were assumed to arrive to each VOQ according to an i.i.d. Bernoulli process, with possibly different rates. We considered three distinct loading scenarios:

5.0.1 **Uniform loading**

In this case, packets arrive at rate $\lambda/16$ to all VOQs. $\lambda$ is varied from 0.1 to 0.9 to vary the load per input port. The average delay per packet vs. $\lambda$ is plotted in Fig. 4.
All schedulers perform well under uniform loading, with MWM predictably yielding the best performance. WWFA is marginally outperformed by MSB-PSS.

5.0.2 Non-uniform loading A

In this case, packets arrive at rate $\lambda_1$ to VOQs which correspond to input port $i$ and output port $16 - i + 2 \mod 16$ (by convention, $16 \mod 16 = 16$), and at rate $\lambda_2 = 0.5/15$ to all the other VOQs. Thus, the total load per input port, viz. $\lambda = \lambda_1 + 15\lambda_2 = \lambda_1 + 0.5$ is varied by varying $\lambda_1$ from 0.05 to 0.45. The average delay per packet vs. $\lambda$ is plotted in Fig. 5. The switch is unstable under WWFA for $\lambda > 0.85$.

5.0.3 Non-uniform loading B

In this case, packets arrive at rate $\lambda_1$ to VOQs which correspond to input port $i$ and output port $i$, and at rate $\lambda_2 = 0.5/15$ to the other VOQs. The total load per input port, viz. $\lambda = \lambda_1 + 15\lambda_2 = \lambda_1 + 0.5$ is varied by varying $\lambda_1$ from 0.05 to 0.45. The average delay per packet vs. $\lambda$ is plotted in Fig. 6. For this loading scenario, the switch is unstable under WWFA for $\lambda > 0.85$. The performance of MSB is very poor and hence not depicted here.

The key difference between non-uniform loading scenarios A and B is that all load vectors in A are covered by the configuration subset $S_W$ (Section 4.2), while this is not true for B. Consequently, MSB yields poor performance in scenario B when operated using configuration subset $S_W$.

The throughput performance of WWFA degrades with increasing switch load, especially under non-uniform loading. However, as visible in the results, the stability region of WWFA can be increased by making it backlog aware.

MSB has the worst average delay performance amongst all schedulers. This is attributed to the fact that MSB can set the switch in only 16 possible configurations, which are not always maximal, resulting in loss of throughput.

While we have only reported an initial representative set of results here, we expect similar performance gains from the proposed schedulers for bigger switches, different non-uniform loading scenarios, and non-Bernoulli traffic. A more thorough performance evaluation of the proposed schedulers under varied conditions is the attention of our ongoing work.

A note on fairness

While backlog aware policies have their advantages, they can sometimes lead to starvation of one or more VOQs. However, as proposed in [11], the starvation problem can be circumvented by using the age of the oldest packet in a VOQ (rather than backlog) as its weight when computing a switch configuration. Incorporating appropriate starvation prevention mechanisms in our proposed algorithms is a topic of ongoing research. We are also focusing on evaluating the short-term and long-term fairness properties of the proposed schedulers.
6 Conclusions

This paper studied the problem of packet scheduling for the input-queued switch architecture, with a view toward low complexity and efficient hardware implementation. A subset operation based scheduler as well as a backlog aware version of the well known WWFA scheduler were proposed. Both schedulers deliver an MWM like performance at significantly lower complexity, and like MWM, do not depend on the knowledge of input traffic statistics. The two key conclusions of the study are: (i) good performance can be ensured at low complexity by cleverly selecting a small number of switch configurations to operate with (rather than using the entire available set of size $N$!), and (ii) incorporating VOQ backlogs into scheduling decisions pays rich dividends.

References