Fair Matching Algorithm: An Optimal Scheduling Algorithm for the AAPN network

AAPN Technical Report 2004-5

Nahid Saberi and Mark J. Coates
Department of Electrical and Computer Engineering
McGill University

Sept. 2005

Abstract

The internal switches in all-photonic networks do not perform data conversion into the electronic domain, thereby eliminating a potential capacity bottleneck, but the inability to perform efficient optical buffering introduces network scheduling challenges. In this technical report we focus on the problem of scheduling fixed-length frames in all-photonic star-topology networks with the goal of minimizing rejected demand. We describe the Fair Matching Algorithm, a novel scheduling technique for fixed-length frames. FMA guarantees 100% throughput provided the arrivals to the network induce an admissible demand matrix, and results in an allocation that is weighted max-min fair. We compare through OPNET simulation the delay and throughput performance of FMA with the less computationally-complex Minimum Cost Search algorithm. We also describe the Minimum Rejection Algorithm (MRA), which minimizes total rejection, and demonstrate that the Fair Matching Algorithm (FMA) minimizes the maximum percentage rejection of any connection. We analyze through simulation the rejection and delay performance.

1 Introduction

Electronic switches in high-speed networks are increasingly proving to be a capacity bottleneck. Replacement with all-photonic switches is attractive, particularly as photonic devices with sub-microsecond switching capability become available. The inability of the photonic switches to perform queuing introduces network design challenges. Control functionality is required to reduce or eliminate the potential of contention for egress ports. Burst switching and just-in-time reservation approaches [1], and routing and wavelength assignment techniques [2], are some of the many approaches that have been used in general mesh topologies. An alternative approach is to focus on a simpler architecture that reduces the complexity of the control challenge.

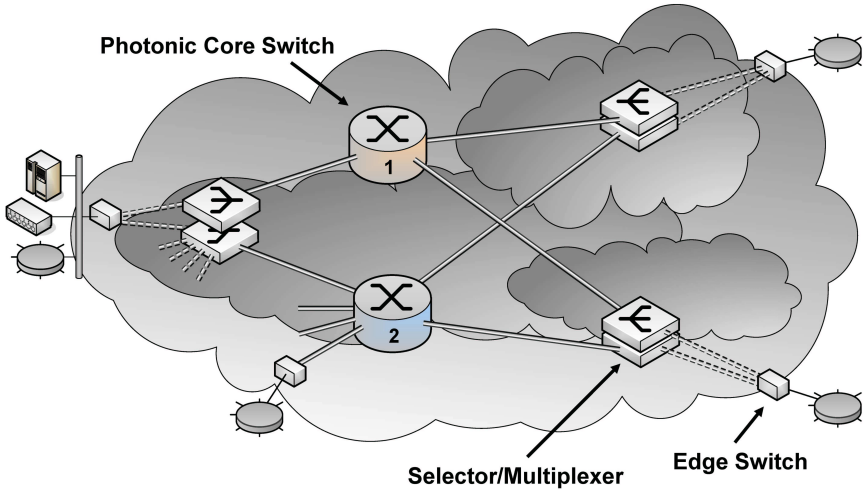

Figure 1. Architecture of the Agile All-Photonic Network described in [3, 4]. Edge nodes perform electronic-to-optical conversion and transmit scheduling requests to the core photonic node(s). Selectors/multiplexor devices are used to merge traffic from multiple sources onto single fibres and to extract traffic targetted to a specific destination. The structure forms an overlaid star topology (see Figure 2).

In this research, we focus on the overlaid star topology, as specified in the design for the agile all-photonic network (AAPN) architecture of [3, 4]. This architecture (see Figure 1) consists of edge nodes, where the optical electronic conversion takes place, connected via selector/multiplexor devices to photonic core crossbar switches. The overlaid star topology facilitates the introduction of various approaches to time-sharing link capacity and dramatically reduces the complexity of the control problem. The core switches act independently, so the control problem becomes one of scheduling the switch configurations to achieve a good match with the traffic arrival pattern at the edge nodes.

The star topology also makes the introduction of accurate network-wide synchronization much more feasible [5], and this enables the application of a range of Optical Time Division Multiplexing (OTDM) techniques for sharing link and switch capacity. A source edge-node must be aware of when it has ownership of a given time-slot and is allowed to transmit to a specific destination edge node. By suitably allowing for the differing propagation delays between various edge nodes and the core, time slots arrive at the core crossbar switch at the same time and can be switched to their appropriate destinations without output port collisions.

The slot allocation can be fixed and deterministic, or it can adapt to the traffic arrivals through signalling

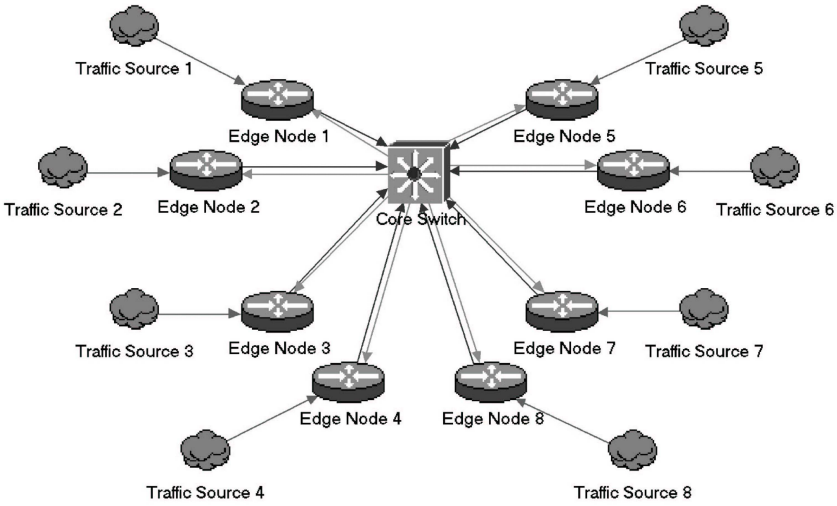

Figure 2. The star topology induced by the agile all-photonic network architecture.

between the edge nodes and the core switch. In the latter case, adaptation can be performed on a per frame basis (a block of slots) or per time-slot basis. Frame-based scheduling is more appropriate for wide-area networks since the impact of propagation delay is reduced (bandwidth is reserved for predicted traffic demand in advance of the traffic arrivals) [6]. We focus on fixed-length frames, because this simplifies protocol design and implementation of control functions.

The general objectives of bandwidth sharing are to achieve minimum loss (rejected requests) or maximum throughput, and minimum end-to-end delay, whilst maintaining fairness in the network. Minimizing the number of rejected requests (those not accommodated by the scheduled frame) has highest priority. A secondary objective is to minimize the number of switching operations in a frame in order to reduce the power consumed by the core switch. In this report we introduce a scheduling algorithm called the *Fair Matching Algorithm (FMA)* and compare its performance (utilization and delay behavior) with that of our previously suggested algorithms, *Minimum Cost Search (MCS)* scheduling algorithm [7], and *Parallel Iterative Matching (PIM)* algorithm [6]. We also introduce the Minimum Rejection Algorithm (MRA), an algorithm for minimizing total rejection, and compare its performance with that of FMA. We demonstrate that FMA minimizes the maximum percentage rejection of any connection.

This technical report is structured as follows. In Section 2 we define the scheduling problem that we address. In Section 3 we present a literature review on the variable length scheduling problem. In Section 4 we present a general solution for the problem of designing a schedule of fixed frame length. Section 5 details our proposed frame-based scheduling approaches. Section 6 describes the simulation experiments we have performed to compare the scheduling approaches and analyzes the results. Finally, Section 7 draws conclusions and indicates intended extensions.

2 Fixed Frame Length Scheduling: Problem Definition

The AAPN architecture is an overlaid star-topology of N edge nodes that operates over multiple wavelengths [4]. It permits each node to transmit to one destination node and receive from one source node simultaneously *on each wavelength*. We consider that (flow-based) load balancing has been conducted to divide incoming traffic amongst the various stars. The remaining task is to schedule the traffic for each star. We are presented with a demand matrix D, where D_{ij} is the number of slots requested by source node i for destination j during the next fixed-length frame. We consider a frame of length F time slots with W available wavelengths, such that there are $L = FW$ slots for each destination node available for allocation. Herein we focus on the case where $W = 1$ for clarity, but the algorithms and results are easily extended.

We are presented with a demand matrix D, where D_{ij} is the number of slots requested by source node i for destination j during the next fixed-length frame. We define the following *line sums* of the demand matrix. The *row-sum*, $r_i(D) = \sum_{j=1}^{N} D_{ij}$, is the total demand at source i, and the *column-sum*, $c_j(D) = \sum_{i=1}^{N} D_{ij}$, is the total demand for destination j. It is important to achieve zero rejection if the demand is *admissible*.

Definition 1. *A demand matrix D is* admissible *if*

$$\max\{\max_i\{r_i(D)\}, \max_j\{c_j(D)\}\} \leq L, \tag{1}$$

where L is the frame-length, and $r_i(D)$ and $c_j(D)$ are the i-th row-sum and j-th column-sum of the demand matrix, respectively.

Our aim is to devise a schedule S such that the element S_{jk} identifies the source node allocated to the k-th time slot associated with destination j in the frame. The schedule should minimize the number of rejections $REJ(S, D, L)$ whilst also attempting to minimize the number of times that the switch must reconfigure, $N_s(S)$. A switch reconfiguration occurs between two consecutive time slots k and $k + 1$ if the allocated source node to any destination j is altered; $N_s(S)$ counts the number of switch reconfigurations in the entire schedule, not merely those within the frame.

The number of rejections is defined as:

$$REJ(S, D, L) = \sum_i \sum_j \max(0, D_{ij} - \sum_{k=1}^{L} \mathbb{I}[S_{jk} = i]), \tag{2}$$

where $\mathbb{I}$ is the indicator function. We can define an objective function (the cost of transmission) as:

$$C(S, D, L) \; = \; REJ(S, D, L) \; + \; g \; . \; N_s(S) \; , \tag{3}$$

where g is a constant that determines the relative importance of reducing the number of switch reconfigurations.

We identify two scheduling problems that address bandwidth allocation in an AAPN:

PROBLEM 1: For an admissible demand matrix D and frame of length L, generate a schedule S that achieves zero rejection, $REJ(S, D, L) = 0$, and allocates spare capacity in the network to the connections in a (weighted) max-min fair manner.

PROBLEM 2: Solve the following optimization problem for a frame of fixed length L with $C(S, D, L)$ defined by (3) to identify a frame schedule.

$$S_1^* = \arg\min_S C(S, D, L) \tag{4}$$

The most closely related work to the optimization embodied in *PROBLEM 1* and *and PROBLEM 2* is the problem of finding an optimum schedule for a variable-length frame, which has been extensively studied in WDM and satellite systems [8–12]. The goal is to minimize the overall transmission time T:

$$T(S) = T_x(S) + \tau \, . \, N_s(S), \tag{5}$$

where N_s is the number of switch reconfigurations, τ is the switching time, and T_x is the time spent transmitting the traffic [9, 11]. The minimum traffic transmission time $T_x^* = \max\{\max_i\{r_i\}, \max_j\{c_j\}\}$ [13]. All times are measured in slots.

PROBLEM 3: Solve the following optimization problem for a frame of variable length with total transmission time $T(S)$ defined by (5), observing the constraint that $S \in \mathcal{S}$, the set of schedules that satisfy the demand matrix, i.e., $REJ(S, D, T_x(S)) = 0$.

$$S_2^* = \arg\min_{S \in \mathcal{S}} T(S) \tag{6}$$

PROBLEM 3 is NP-hard for non-negligible values of τ [11, 14]. Crescenzi et al. demonstrate that it cannot be approximated by a polynomial algorithm within a factor less than $\frac{7}{6}$ [14]. For small values of τ the problem can be closely approximated by the minimization of T_x, which is solvable in polynomial time [15–17]. The minimum traffic transmission time is [13]:

$$T_x^* = \max\{\max_i\{r_i\}, \max_j\{c_j\}\}.$$

We can then establish:

Claim 1. *A schedule S_x that minimizes the traffic transmission time, i.e., $T_x(S_x) = T_x^*$, solves PROBLEM 3 to within an approximation factor of $1 + \tau$.*

Proof. The number of switch reconfigurations $N_s(S) < T_x(S)$ and $T(S_2^*) = T_x(S_2) + \tau N_s(S_2) > T_x^*$. Hence if S_x minimizes the traffic transmission time, it satisfies $T(S_x) < T_x^*(1 + \tau) < T(S_2^*)(1 + \tau)$. □

For the special case of small τ, approximate algorithms that attempt to minimize N_s subject to the constraint that T_x is minimum have been proposed in [8, 9, 12, 14]. The algorithms achieve minimum traffic transmission time, T_x^*, but do not guarantee minimum *total* transmission time, $T(S_2^*)$, unless the switching overhead is completely neglected. The *EXACT* algorithm, presented in [12, 14], achieves a minimum traffic transmission time, T_x^* and the derived schedule has at most $N_s = N^2 - 2N + 2$ switch configurations [12]. In the case of an admissible demand matrix, the *EXACT* algorithm generates a schedule S that has length less than L and therefore zero rejection. The *EXACT* algorithm is an iterative procedure that repeatedly performs maximum cardinality bipartite matching (MCBM) to obtain the schedule. It lies at the heart of the algorithms we present in this report for the case of fixed-length frames.

When τ is very large (on the order of maximum transmission time), the problem is reduced to minimizing T_D subject to the constraint that N_s is minimum. Approximate algorithms for this special case have been proposed in [9, 11]. The intermediate scenario, when it is desirable to obtain near minimum solutions for both the number of switchings and the traffic transmission time, has been addressed in [12].

3 Literature review

Depending on the value of τ the problem of finding a minimum schedule length given by equation 5 is usually reduced to the three following cases.

1) When τ is negligible compared to the duration of a time slot, the problem is reduced to the problem of minimizing T_D, which has been studied in [15–17]. This problem can be solved in polynomial time. In a more precise manner the problem can be reduced to minimizing N_s subject to the constraint that T_D is minimum [8, 9].

2) When τ is very large (in the order of minimum transmission time), the problem is reduced to minimizing T_D subject to the constraint that N_s is minimum [9, 11].

3) When τ is moderately large, it is more desirable to obtain near minimum solutions for both the number of switchings and the traffic transmission time [12]. In [14] it is shown that one cannot approximate this problem within a factor less than $\frac{7}{6}$.

The three problems stated above have been formulated as variants of open shop scheduling problem in literature [10–13, 18].

3.1 Open Shop Formulation

First, we describe a scheme for classifying scheduling problems developed by Graham et al. [19]. Suppose that M machines or processors P_k $(k = 1, ..., M)$ have to process N jobs J_i $(i = 1, ..., N)$. A schedule is an allocation of one or more machines to each job. A schedule is *feasible* if at any time, there is at most one job on each machine, each job is run on at most one machine, and it satisfies a number of requirements concerning the machine environment and the job characteristics. A schedule is optimum if it minimizes (or maximizes) a given optimality criterion.

A shop scheduling problem consists of a set of M processors. Each of these processors performs a different task. There are N jobs, each consisting of M tasks. Each task j of job i denoted by $t_{i,j}$ is to be processed on processor j for a total duration of $dur(t_{i,j})$. In each time the following restrictions must be satisfied for the machines and the jobs: (i) each machine can execute at most one task at any given time, and (ii) for each job at most one task is to be assigned. Depending on the orders by which the jobs and the tasks should be performed the shop scheduling problem is usually classified to three basic groups:

1. When there is no ordering constraints on operations of the jobs the scheduling is described as an *open* shop scheduling.

2. When operations of the tasks of each job should follow a specific order the shop scheduling is called *job* shop.

3. When every job goes through all M machines in a unidirectional order the shop scheduling is *flow* shop. Each job has exactly M tasks. The first task of every job is done on machine 1, second task on machine 2 and so on. However, the processing time each task spends on a machine varies depending on the job that the task belongs to.

When the shop is $open$ the jobs and tasks can be executed in any order. The scheduling algorithms can be designed for two different categories based on how the tasks deal with interruption: *preemptive* and

nonpreemptive schedules. A preemptive schedule is the one which does not restrict the tasks to be executed continuously. A nonpreemptive schedule is the one in which the tasks can not be interrupted once they have begun execution [9].

For a given open shop problem, we try to obtain an optimal finish time (OFT). An OFT minimizes the $makespan$ or the time required to complete all the jobs. In general a *preemptive open shop* problem can be solved in polynomial time, while a nonpreemptive open shop problem is shown to be NP-hard [1] [11] for more than three machines [13, 20], but many heuristics exist to obtain near-optimal finish time for this case [21, 22]. The open shop scheduling problem of N jobs and M machines is denoted by $N \mid M \mid openshop \mid OFT$ [9].

3.2 Analogy

The scheduling problem for an $N \times N$ optical switch can be translated into an open shop scheduling problem with $M = N$ processors and N jobs. The jobs correspond to the inputs of the switch and the processors correspond to the outputs. Each input-output traffic demand, D_{ij}, is represented by a task $t_{i,j}$. Similar to the open shop formulation each task, D_{ij}, belongs to a specific job (input i) and is to be processed by a specific processor (output j) [9]. The scheduling problem obtained is *N | N | open shop | OFT*. The scheduling constraint in an optical switch operating on one wavelength can be translated directly to the open shop scheduling. The constraint in an optical switch is that at each given time (i.e., a time slot) at most one request from input i can be serviced at each output j:

• One assignment of each input per slot is equivalent to the constraint that a job can not be processed by more than one processor at any given time.

• One assignment of each output per slot is equivalent to the constraint that a processor can perform at most one task at any given time [9].

3.3 Scheduling Algorithms

In this section we describe several solutions for open shop scheduling problem which have been designed for satellite systems and passive star networks. We develop the algorithms for an $N \times N$ nonblocking optical switch. The first group of the algorithms aim at minimizing the number of switchings for a minimum duration schedule [8, 9]. The second group of the algorithms try to minimize the schedule length when the number of switchings is minimum [9, 11]. The third group of algorithms provide near-optimum solutions with bounds on the number of switchings and the schedule length [12].

3.3.1 Minimal Duration Scheduling

The problem of finding a schedule with minimum duration has been studied in many research areas such as networks, computers, and satellite systems [8, 9, 12, 17], and was shown to be solvable in polynomial time. The algorithms achieve a minimum traffic transmission time, $T_{D_{min}}$, but the minimum schedule length, T_{min}, for non-negligible switching overhead is not guaranteed. In this section we present a simple algorithm which obtains at most $N_s = N^2 - 2N + 2$ switch configurations per port [12]. However, this algorithm is a $pseudo$polynomial time algorithm not a polynomial-time one [14]. A similar algorithm proposed in [9, 13] overcomes the problem noted above by obtaining a weight-regular graph from the original one. A

[1] A problem is NP-hard if solving it in polynomial time would make it possible to solve all problems in class NP in polynomial time.

weight-regular graph is defined as a graph whose vertices have the same number of incident edges. Another algorithm proposed by Pomalaza [10], tries to reduce the number of switchings by the weight matching algorithm. All of these algorithms assume that the assignments for each demand do not need to be continues, which translates our scheduling problem to solving a $preemptive\ N \mid N \mid openshop \mid OFT$ problem.

Given a set of N jobs with task times $D_{ij}, 1 \leq i \leq N$ and $1 \leq j \leq N$, for an N-processor open shop problem, we define the following quantities:

$$\begin{aligned} r_i &= \sum_{1 \leq j \leq N} D_{ij} \ = \ length\ of\ job\ i, \\ c_j &= \sum_{1 \leq i \leq N} D_{ij} = \ total\ time\ needed\ on\ processor\ j, \end{aligned} \tag{7}$$

where r_i, is the amount of the demand at input i, and c_j is the total demand for output j. It is easily understood that $T_{D_{min}} = \max\{max_i\{r_i\}, max_j\{c_j\}\}$ (the maximum Line-sum in the traffic matrix).

Before describing the algorithms, we review some terminology and fundamental definitions concerning bipartite graphs. The following definitions are presented from [13, 23].

Definition 2. *A graph G is a pair $G = (Y, E)$ where Y is a finite set of nodes or vertices and the elements in E consist of subsets of Y of cardinality two called edges. If $e = [v_1, v_2] \in E$, then we say that e is incident upon v_1 (and v_2). The degree of a vertex v of G is the number of edges incident upon v.*

Definition 3. *A graph G in which each edge has been assigned a number, or a weight, is called a weighted graph. In a weighted graph, the weight of a vertex v is defined as the sum of the edge weights of all edges incident to v in graph G.*

Definition 4. *Let $G = (Y, E)$ is a graph that has the following property: the set of vertices Y can be partitioned into two sets, V and U, and each edge in E has one vertex in V and one vertex in U. Then G is called a bipartite graph and is usually denoted by $G = (V \cup U, E)$. Figure 3-(a) shows a bipartite graph of 8 vertices and 9 edges.*

Definition 5. *Let $G = (V \cup U, E)$ be a bipartite graph with vertex sets V and U, and edge set E. A set $I \subseteq E$ is a matching if no vertex $w \in V \cup U$ is incident with more than one edge in I. A matching of maximum cardinality is called a $maximum\ matching$. A matching is called a $complete\ matching$ (or perfect matching) of V into U if the cardinality (size) of I equals the number of vertices in U. Figure 3-(b) shows a matching of maximum cardinality obtained from the graph of figure 3-(a).*

Definition 6. *An augmenting path P relative to a matching M in G is a path in G such that the first and the last vertices in P are not covered by M, and the edges in P alternate between being in M and not being in M.*

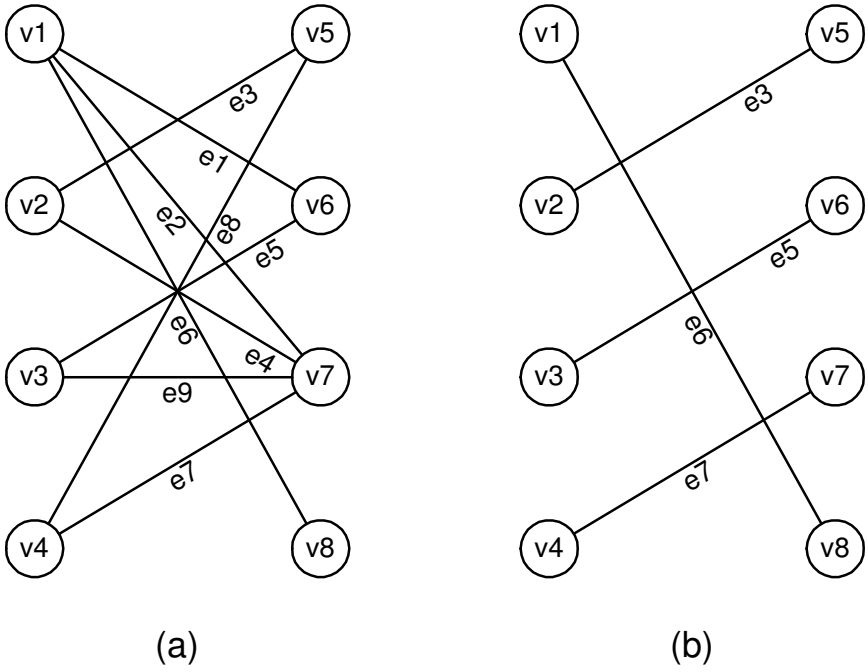

Figure 3. (a) A bipartite graph of 8 vertices and 9 edges. (b) A matching of maximum cardinality obtained from the bipartite graph. This matching is a perfect matching.

Note: If there exists such an augmenting path it can be proved that a matching of greater cardinality can be found in G by augmenting M with augmenting path P. Consequently, a matching is of maximum cardinality if and only if it permits no augmenting path [23].

Definition 7. *The degree of a graph $G = (Y, E)$ is defined as the maximum degree of all its vertices. For example in figure 3-(a) the degree of the graph is 3.*

Maximum Cardinality Matching Algorithm for Bipartite Graph (MCB): The algorithm starts with an arbitrary matching Q in graph G. An augmenting path P with respect to Q is found. Then a new matching is constructed by taking those edges of Q or P that are not in both Q and P. The process is repeated and the matching is maximal when no augmenting path is found.

EXACT Covering Algorithm. The *EXACT* algorithm, presented in [12, 14], is based on finding the maximum cardinality matching in bipartite graphs. We construct a bipartite graph $G = (V \cup U, E)$, where V is the set of vertices corresponding to the N jobs (inputs of the optical switch), U is the sets of vertices corresponding to the N processors (outputs of the optical switch), and E is the set of edges incident to each input-output pair. The algorithm repeatedly performs maximum cardinality matching on the nonzero elements of the traffic matrix D. The weight of each matching, which corresponds to each configuration, is equal to the minimum weight of the edges participating in the matching. The weight of each edge (v_i, u_j) is the amount of the requested traffic from input i to output j.

Algorithm EXACT.

%initialization
$i = 1, A = D$
create graph $G = (V \cup U, E)$ from A
% find a maximum matching M of this graph using algorithm MCB
while $A \; ! = \overline{0}$;
$M = MCB(V \cup U, E)$;
% schedule construction
$P(i) = M$; %construct a permutation matrix
$w(i) = min\{w(e) : e \in M\}$;
%update the graph
$A = A - w(i)p(i)$;
%update the graph
$A = A - w(i)p(i)$;
i=i+1;
%finish when all of the elements in A are zero
end

It has been shown in [12, 17] that at most $N_s = N^2 - 2N + 2$ switch configurations are necessary to cover the traffic matrix. However, this approach provides a $pseudo$polynomial-time algorithm, since its running time depends linearly on the weights of G [14]. In [9, 13] a similar algorithm, namely Complete Matching Algorithm (CMA) has been proposed which obtains the schedule in a polynomial time. In this approach the bipartite graph is constructed using $N + M$ real nodes and $N + M$ fictitious nodes. The idea is to make each processor to have the same load ($T_{D_{min}}$), and each job to have the same processing requirement ($T_{D_{min}}$), hence obtaining a weight-regular graph, that is, a graph whose vertices have equal weights. A weight-regular graph guarantees the existence of a complete matching. With complete matchings it is guaranteed that all processors are being used and all jobs are being processed in each iteration [13]. Therefore it can be proved that the algorithm CMA always produces a minimum length schedule of duration $T_{D_{min}}$ [9, 13]. The time complexity of this algorithm is $O(\|V\|\|E\|)$ where $\|V\|$ and $\|E\|$ are the number of vertices and the number of edges in the Bipartite graph respectively. The size of a maximum cardinality matching, and thus the maximum number of iterations required to compute it, is $O(\|V\|)$, and the complexity of a graph search procedure is $O(\|E\|)$.

Using parallel processing methods, the fastest known algorithm has a complexity of $O(\sqrt{\|V\|}\ \|E\|)$ [24]. This algorithm reduces the number of iterations from $O(\|V\|)$ to $O(\sqrt{\|V\|})$, by looking for a set of disjoint M-augmenting paths per iteration, then augmenting along all the discovered paths. In other words, the algorithm runs the search procedure from all unmatched vertices simultaneously rather than one by one .

3.3.2 Minimum Number of Switchings

Recall that the objective function to minimize the overall transmission time is $T = \tau N_s + T_D$, where τ is the switching overhead, T_D is the traffic transmission time, and N_s is the number of switchings that might be taken to cover the traffic demand. Algorithms described in this section give a schedule with minimum

number of switchings. The algorithms aim at minimizing T_D subject to the constraint that N_s is minimum. The problem in this case is formulated as a *nonpreemptive open shop* scheduling problem. Nonpreemptive scheduling guarantees that the minimum number of switchings is always obtained. In [25] it has been proved that minimizing the makespan (total transmission time) in a nonpreemptive open shop scheduling problem when $M > 2$ is NP-complete, where M is the number of processors (for an $N \times N$ switch $M = N$).

The minimum number of switchings is determined by:

$$N_{s_{min}} = \max\{max_i\{\|r_i\|\}, max_j\{\|cj\|\}\}, \tag{8}$$

where $\|r_i\|$ is the number of nonzero entries in row i and $\|cj\|$ is the number of nonzero entries in column j.

Nonpreemptive Open shop Scheduling Algorithm. The algorithm presented in [9] is based on the most number of tasks (demands) first heuristic. The algorithm starts by the output which is requested by the largest number of inputs. At a given time slot if the j-th output is available and the inputs i and k are free, if the total number of tasks of input i is greater than the total number of tasks of input k, then the demand D_{ij} is processed before the demand $d_{k,j}$. Recall that the total number of tasks of input i is the total number of outputs for which input i has requests (the number of nonzero entries in row i of a demand matrix). If the total number of tasks of input i is equal to the total number of tasks of input k, and if the total number of requests by input i is smaller than the total number of requests by input k, then D_{ij} is processed before the demand $d_{k,j}$. The total number of requests by input i is the sum of the total demands requested by input i. This algorithm always produces the minimum number of switching matrices.

3.3.3 Near-Optimum Solutions

The algorithms described in sections 3.3.1 and 3.3.2 provide solutions for reducing the number of switchings and the transmission time respectively, but there is no guarantee on the performance of the proposed algorithms when the amount of τ is neither negligible nor very large. Using the proof described in [23] it can be shown that the algorithm described in 3.3.2 has an unbounded approximation ratio for the transmission time. On the other hand the algorithms described in 3.3.1 provide the minimum traffic transmission time, but there is not a tight bound on the number of switchings. Consequently, the overall transmission time for the case that the switching overhead is not negligible is not close to optimal.

In [14] this problem is described as the preemptive bipartite scheduling and shown to be NP-hard using the proof in [11]. Also it is shown that one cannot approximate this problem within a factor less than $\frac{7}{6}$, but the best algorithm they proposed approximates this problem within a factor of 2.

The approximation algorithm described in this section does not restrict the optimal schedule length or the minimum number of switchings. Instead, by allowing twice as many as the minimum number of switchings, it achieves a near-optimum schedule length within a ratio of two.

Graham's List Scheduling. List scheduling (LIST) introduced by R. L. Graham [18] is a greedy algorithm[2] that approximates the optimal open shop problem within a ratio of two. LIST starts by assigning a job to each processor. If multiple jobs are contending for the same processor one of them is chosen arbitrarily. Once a processor is idle, one of the free jobs which has a task for the corresponding processor is chosen arbitrarily. This procedure continues until all of the jobs are processed.

[2] A greedy algorithm is an algorithm that optimizes the choice at each stage without regard to previous choices, with the hope of finding the global optimum.

The maximum schedule length produced by LIST is bounded by the sum of the time to process the longest job (equal to the maximum row-sum in the traffic demand), and the time that the most heavily loaded processor needs (equal to the maximum column-sum in the traffic demand) [26]. Therefore this algorithm tends to a delay overhead of at most $2\tau N_{s_{min}}$, though the number of switchings at each port is $N_{s_{min}}$.

4 Fixed Frame Length Scheduling

4.1 Terminology and Definitions

We now define some terminology that will be used throughout the report and recall some definitions. We denote the line-sum of line ℓ of the demand matrix D by LS_ℓ. Note that line ℓ consists of a set of source-destination demands (connections). Each of these connections belongs to two lines (a row and a column). The i-th row represents a link from source i to the optical switch at the core, and the j-th column represents the link from the core to destination node j.

For an inadmissible demand matrix, we denote the set of overflowing rows of the demand matrix (rows with $r_i(D) > L$) as O_r, and the set of overflowing columns ($c_j(D) > L$) as O_c. The set of overflowing lines, $O_\ell = \{\ell : \quad LS_\ell > L\}$ is the union of O_r and O_c. We define a *critical connection*, or critical demand element, as any demand entry D_{hp} such that $h \in O_r$ and $p \in O_c$. The remaining entries constitute *non-critical* connections/demands.

We now recall the definitions of *feasibility* of rate allocation and *weighted max-min fairness* [27, 28].

Definition 8. Feasibility*: Consider an arbitrary network as a set of links $\mathcal{L}$ where each link $\ell \in \mathcal{L}$ has a capacity $C_\ell > 0$. Let $\{1, \cdots, \zeta\}$ be the set of connections in the network. Let D_u be the demand (request) of connection u and υ_u be its assigned rate. We call a rate allocation $\{\upsilon_1, \upsilon_2, \cdots, \upsilon_\zeta\}$* feasible*, when for every link ℓ we have:*

$$\sum_{u \in H_\ell} \upsilon_u \leq C_\ell \quad \forall \ell \in \pounds. \tag{9}$$

Definition 9. Weighted max-min fairness*: Let $\omega_u(\upsilon_u)$ be an increasing function representing the weights assigned to connection u at rate υ_u. An allocation $\{\upsilon_1, \upsilon_2, \cdots, \upsilon_\zeta\}$ is* weighted max-min fair *if for each connection u any increase in υ_u would cause a decrease in transmission rate of connection z satisfying $\omega_z(\upsilon_z) \leq \omega_u(\upsilon_u)$. The special case of max-min fairness is obtained by $\omega_u(\upsilon_u) = \upsilon_u$.*

Definition 10. Bottleneck Link*: Given a feasible rate vector υ and a weight vector ω, we say that link ℓ is a* bottleneck link *with respect to (υ , ω) for a connection u crossing ℓ, if $C_\ell = \sum_k \upsilon_k \triangleq F_\ell$ and $\omega_u \geq \omega_k$ for all connections k crossing ℓ.*

Lemma 1. *A feasible rate vector υ with weight vector $\omega = \{\frac{\upsilon_u}{R_u}\}$ is weighted max-min fair if and only if each connection has a bottleneck link with respect to (υ , ω).*

See the Appendix (Section 8.1) for a proof.

4.2 Relationship to Variable-Length Frame Scheduling

The *EXACT* algorithm, presented in [12, 14], addresses schedule design for variable length frames, primarily for the case of negligible τ, and achieves a minimum traffic transmission time, T_x^*. Thus in the case of

admissible demand matrices, the *EXACT* algorithm generates a schedule S that has length less than L and therefore satisfies the first requirement of *PROBLEM 1*. The *EXACT* algorithm is an iterative procedure that repeatedly performs maximum cardinality bipartite matching (MCBM) to obtain the schedule. It lies at the heart of the algorithms we present in this report for the case of fixed-length frames.

We establish two results concerning the complexity of *PROBLEM 1*:

Claim 2. *If the demand matrix D is admissible and contains no zero entries (for an $N \times N$ switch and frame of length L) then the EXACT algorithm provides a solution S_E to PROBLEM 2 such that $C(S_E) < C(S_1^*) + g(N^2 - 3N + 2)$.*

Proof. Since the demand matrix is admissible, $T_x^* < L$. Hence the schedule devised by *EXACT* results in zero rejections, $REJ(S, D, L) = 0$. *EXACT* ensures that the number of switch reconfigurations in this solution is less than $N^2 - 2N + 2$. The minimum number of switch reconfigurations for any schedule under the constraint of no zero-entries in the demand matrix is N [25]. Hence the maximum discrepancy is $N^2 - 3N + 2$. □

Theorem 1. *For large g, such that $g > \max(||D||_1 - L, 0)$, where $||D||_1 = \sum_i \sum_j D_{ij}$, PROBLEM 2 is reduced to the problem of minimizing $REJ(S, D, L)$ subject to the constraint that $N_s(S)$ is minimized. For this range of g, PROBLEM 2 is NP-hard.*

See the Appendix (Section 8.2) for a proof.

5 AAPN Scheduling Algorithms

In a practical scenario, although it is desirable to reduce power expenditure by minimizing the number of switchings, minimizing the number of rejections is far more important. Hence we address the scheduling problem (*PROBLEM 1*) when g is small. In this case, we can rewrite the problem as:
MINREJ(D,L): For a frame of fixed length L with demand matrix D identify a frame schedule S_1^* that satisfies:

$$S_1^* = \arg\min_S REJ(S, D, L) \tag{10}$$

In this section, we describe two algorithms for bandwidth reservation in the AAPN architecture that address fixed-length frame scheduling. The Fair Matching Algorithm minimizes the maximum percentage rejection experienced by any demand, while the Minimum Rejection Algorithm minimizes the total rejection (that is, it provides a solution to *MINREJ(D,L)*).

5.1 Fair Matching Algorithm (FMA)

The *EXACT* algorithm can be applied directly to the case of fixed length frames (Claim 2 states that it provides a solution for *PROBLEM 1* when g is small and demand admissible). When the demand matrix is inadmissible, the schedule determined by the EXACT algorithm must be truncated after L time slots. This can lead to starvation of some source-destination traffic, and result in unfairness (such as substantially different average service times for traffic arriving at different nodes).

If the demand matrix is admissible, FMA incrementally assign additional demand to all elements until one of the links reaches capacity (its line-sum is equal to L). At that point, the demand elements contributing to that line are clamped. Extra demand is then gradually added to the remaining elements in the matrix until another link (line) reaches its capacity and it too is clamped. The procedure referred to as the *water-filling*

procedure repeats until all lines have reached capacity . FMA assigns extra capacity *in proportion to the original demand*.

This algorithm can be implemented by processing one line at a time. We first choose the most constrained line (the line that would reach its capacity first under the water-filling procedure) and increase its demand to capacity. Then we choose the next most constrained line and increase its demand to capacity. We repeat until all lines have reached capacity.

A similar procedure can be used for the case of an inadmissible demand matrix. In this case FMA identifies the most overloaded line and reduces the demands on that line such that they sum to capacity (L). Demand reduction is proportional to the original demand, i.e. each adjusted demand experiences the same *percentage reduction*. In subsequent iterations, FMA identifies the next most constrained line, taking into account the effect of any previous adjustments, and clamps its demand to capacity. It repeats the process until no lines exceed capacity. When there are both overloaded and under-utilized lines, the overloaded lines are adjusted first.

Here we describe how FMA treats demands belonging to the adjustable lines in the set $U_\ell = \{\ell : LS_\ell(0) \neq L\}$, where $LS_\ell(0)$ is the line sum of line ℓ at the beginning of calculations. We define $\mathcal{A}_D \subseteq U_\ell$ as the set of unmodified lines and $\mathcal{B}_D \subseteq U_\ell$ as the set of modified lines. Initially $\mathcal{A}_D$ contains all lines in U_ℓ and $\mathcal{B}_D$ is empty. Similarly, we define a_ℓ as the set of unmodified demands in line ℓ and b_ℓ as the set of modified demands. Initially, a_ℓ contains all the demands and b_ℓ is empty. Define $S_{a_\ell} \triangleq \sum_{(i,j)\in a_\ell} D_{ij}$ and $S_{b_\ell} \triangleq \sum_{(i,j)\in b_\ell} D'_{ij}$. We always have $S_{a_\ell} + S_{b_\ell} = LS_\ell$, and D'_{ij} is obtained from the following line adjustment:

$$D'_{ij} = D_{ij} \times \frac{L - S_{b_\ell}}{S_{a_\ell}} \quad \forall\, (i,j) \in a_\ell \tag{11}$$

Note that when demand D_{ij} belongs to an overloaded line, $\frac{L-S_{b_\ell}}{S_{a_\ell}} < 1$, and when D_{ij} belongs to an under utilized line $\frac{L-S_{b_\ell}}{S_{a_\ell}} > 1$. Define for each of line in $\mathcal{A}_D$ the value $G_\ell \triangleq \frac{L-LS_\ell}{S_{a_\ell}}$.

Algorithm 1 FMA

while $\mathcal{A}_D \neq \emptyset$ **do**
  Identify the line $\ell^* = \arg\min_{\ell\in\mathcal{A}_D} G_\ell$.
  Apply (11) to line ℓ^*.
  Transfer ℓ^* from $\mathcal{A}_D$ to $\mathcal{B}_D$.
  Update a_ℓ and b_ℓ for all lines $\ell \in \mathcal{A}_D$.
  Re-evaluate LS_ℓ for all lines in $\mathcal{A}_D$.
  Transfer lines γ with $LS_\gamma = L$ from $\mathcal{A}_D$ to $\mathcal{B}_D$.
end while
Apply *EXACT* to $\lfloor D' \rfloor$ to generate S.

The following theorem states that prior to rounding, FMA achieves weighted max-min fair allocation of capacity (weighted relative to the original demand). See the Appendix (Section 8.3) for the proof of the theorem.

Theorem 2. *FMA generates an adjusted demand matrix D' with weighted max-min fair allocation, where the weight is $\omega(D'_{ij}) = \frac{D'_{ij}}{D_{ij}}$.*

If the demand matrix contains zero entries, then an algorithm that adjusts requests multiplicatively (such as FMA) cannot always generate full utilization; there can be *natural blocking* because there is no

demand. We now present some properties of the demand matrix $D^{'} = \{D^{'}_{ij}\}$ obtained by Algorithm 1 prior to rounding.

Property 1 Algorithm 1 guarantees full allocation of all links provided D contains no zero elements.

Property 2 If there is no natural blocking the maximum total throughput of the network is obtained:

$$\sum_i \sum_j D^{'}_{ij} = N.L. \tag{12}$$

Property 3 The while-loop in Algorithm 1 has $O(N^2)$ computational complexity in terms of the number of edge nodes ($2N$ iterations with a minimization over N elements in each iteration). The best current implementation of the *EXACT* algorithm has complexity $O(N^{\frac{5}{2}})$, and hence this is also the complexity of Algorithm 1.

Property 4 Algorithm 1 guarantees minimum rejection if no connections cross two different overloaded links, i.e., if the overloaded links correspond entirely to rows (input links) or entirely to columns (output links) of D. In this case:

$$\min(REJ) = \sum_{\ell} (LS_\ell \ - \ L) \quad \forall \ell \in O, \tag{13}$$

where O is the set of overflowing lines.

Define the *percentage rejection* as $1 - \frac{D'_{ij}}{D_{ij}}$. Consider the set of demands that experience the highest percentage rejection (i.e. the demands on the most overloaded line). Since the weight ω is a monotonically increasing function of allocated rate D'_{ij}, weighted max-min fairness implies that it is impossible to increase the rate allocated to these demands (or decrease the maximum percentage rejection) without violating feasibility. Decreasing the rejection of any of those demands requires increasing the rejection of another demand on the same line, and hence the maximum percentage rejection increases. We thus have the following corollary:

Corollary 1. *Subject to the capacity constraints, FMA generates a schedule that minimizes the maximum percentage rejection experienced by any connection.*

5.2 Equal Share Algorithm (ESA)

The water-filling procedure can be implemented by assigning equal amount of extra capacity to the connections passing underloaded lines. This approach is similar to the max-min fair rate allocation of ABR connections in ATM networks proposed by Charney et. al [29]. Similarly the overloaded lines can be adjusted by reducing equal amounts from the demands of the connections passing these lines. We define for each line the values $H_\ell \triangleq \frac{L - LS_\ell}{|a_\ell|}$, where $|a_\ell|$ is the cardinality of a_ℓ.

The line with minimum H_ℓ is the most constrained line. Repeatedly the most constrained line is defined and the demands of its connections are adjusted. The demand adjustment we perform on each line is:

$$D^{'}_{ij} = D_{ij} + \frac{L - LS_\ell}{|a_\ell|} \quad \forall \ (i,j) \in a_\ell \tag{14}$$

The following theorem states that prior to rounding, ESA achieves max-min fair allocation of capacity. See the Appendix (Section 8.4) for the proof of the theorem

Theorem 3. *ESA generates an adjusted demand matrix D' with max-min fair allocation of extra-capacity, where the weight of the connection between source i and destination j is $\omega_{ij} = D_{ij} - D'_{ij}$.*

5.3 Minimum Cost Search Algorithm

This section reviews an alternative approach, the minimum cost search (MCS) algorithm, which we first described in [7]. In order to reduce signalling overhead and to reduce scheduling complexity, the algorithm satisfies the transparency property [30]. This requires that the scheduling is only modified for new requests or tear-downs (if D_{ij} decreases or increases).

The minimum cost search algorithm we propose does not achieve optimal utilization, because it does not consider the global allocation problem; instead it allocates requests sequentially on a single time slot basis. The algorithm operates by repeatedly visiting the (i, j) entries in the traffic demand matrix D in a round-robin fashion; at each visit, if the requested number of slots has not yet been assigned, the algorithm attempts to allocate a single time slot to the (i, j) request. The round-robin allocation results in an approximately fair assignment of slots to each pair.

In order to determine which slot to allocate to the request, we define a *cost* for the allocation of a (i, j) source-destination pair to a time slot pair t_k for k in $1, \ldots L$. This cost is determined entirely by the extant, partial frame schedule. The cost function is:

$$C_{ij}(t_k) = N_{fs}(t_k) + \lambda K_{ij}(t_k), \tag{15}$$

where $N_{fs}(t_k)$ is the number of free sources at this time slot, i.e., the number of sources not transmitting to any other destinations, λ is a small positive constant, and $K_{ij}(t_k) = \{0, 1, 2\}$ is the number of additional switching operations that the core switch must perform to accommodate the allocation. The motivation behind this cost function is simple. The first term represents the current flexibility of that time slot (the number of free sources for future allocation) and reflects the desirability of retaining flexibility by allocating demands to the most constrained slots where possible. The second term reflects the desirability of minimizing the power consumption of the optical switch, which is partially determined by the number of switching operations that it must perform each frame.

The scheduling of a single (i, j) time slot request is performed by first identifying the (i, j)-*eligible* slots in the frame, which are defined as the free time slots during which i is not transmitting to any other destination and j is not receiving from another source. The cost $C_{ij}(t_k)$ of each of these eligible time slots is evaluated, and the demand is assigned to the slot incurring minimum cost. In the case of ties, the demand is assigned to the earliest slot and the lowest wavelength (assuming wavelengths are ordered in some fashion). Deallocation is implemented by a reverse procedure, in which we seek and release the most costly currently-allocated time slot. This algorithm has a worst case time complexity of $O(N.L)$. (green numbers)

5.4 Minimum Rejection Algorithm

In this section we describe an algorithm that generates a schedule that minimizes total rejection. We first develop a theorem that helps to identify a procedure for solving *MINREJ(D,L)*. We commence by defining *MAXFLOW(D,X,L)*, a max-flow linear programming problem.

Problem Y = MAXFLOW(D,X,L): D is a demand matrix, X is a non-negative matrix that specifies capacity bounds, and L is the frame-length (available capacity on each row/column). Matrices D, X and Y are all of size $N \times N$. Identify a nonnegative matrix Y such that $\sum_{h \in O_r} \sum_{p \in O_c} Y_{hp}$ is maximized, subject to

the following constraints:

$$\begin{aligned} Y_{hp} &= 0 \quad \text{if } h \notin O_r \text{ or } p \notin O_c \\ Y_{hp} &\leq X_{hp} \quad \forall\, (h,p) \text{ s.t. } h \in O_r \text{ and } p \in O_c \\ \sum_{p \in O_c} Y_{hp} &\leq r_h(D) - L \quad \forall\, h \in O_r \\ \sum_{h \in O_r} Y_{hp} &\leq c_p(D) - L \quad \forall\, p \in O_c \end{aligned}$$

The following theorem establishes a relationship between a solution to the problem *MAXFLOW(D,D,L)* and a solution to the minimum rejection problem *MINREJ(D,L)*. The proof is in the Appendix (Section 8.5).

Theorem 4. *Set A = MAXFLOW(D,D,L). Construct a rejection matrix $D'' = A+Q$, where Q is an arbitrary non-negative matrix such that $Q_{hp} \leq D - A \;\; \forall \;\; (h,p)$, $r_h(Q) = r_h(D) - L - r_h(A) \;\; \forall \;\; h \in O_r$, and $c_p(Q) = c_p(D) - L - c_p(A) \;\; \forall \;\; p \in O_c$. Then if S is a schedule that generates the decomposition $D = D' + D''$, it is a solution to the problem MINREJ(S,D,L).*

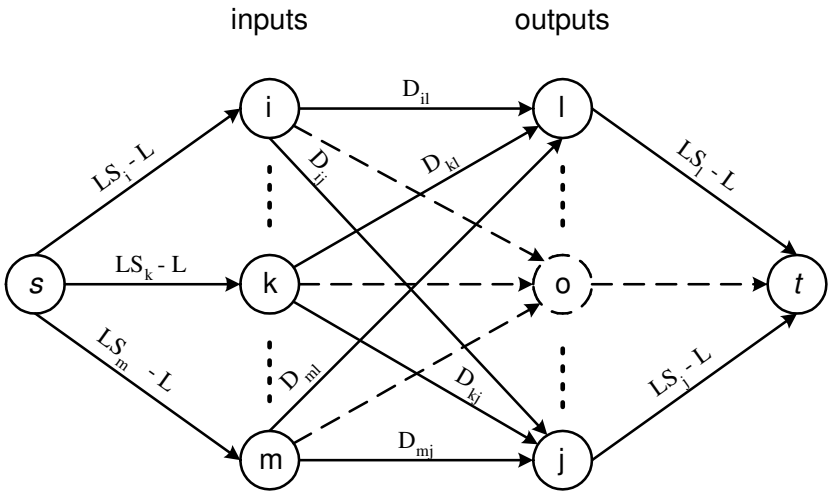

Figure 4. $s \rightarrow t$ network: In this example the input vertices correspond to the overflowing rows of an arbitrary demand matrix D $(i,k,m \in O_r)$, and the output vertices correspond to the overflowing columns of D $(l,o,j \in O_c)$. The numbers over the edges show the edge capacities which correspond to the upper bounds of flows in our maximization problem. The capacity of each edge (not connected to the source or sink) is equal to the upper bound on the amount of rejection that can be assigned to the corresponding critical connection.

We now describe an algorithm to identify a solution A to *MAXFLOW(D,D,L)*. The corresponding maximum flow problem is depicted in Figure 4. We define a network with a source s and a sink t and try to maximize the flow between them. A network flow is a vector $\mathbf{f} = (f_{ij})$ where each f_{ij} is a positive real number representing the flow on arc (i,j), i.e., the flow from i to j. A flow $\mathbf{f}$ is feasible if it satisfies the capacity constraints and it is conserved at all nodes (total flow out of a node equals total flow in). In our problem, the total amount of flow emitted from source s (and therefore arriving at sink t) is equal to the total amount of rejection contributed by A at the critical connections. The rejection at any specific critical

connection (A_{hp}) is equal to the flow on arc (h, p). The capacities of the edges (upper bounds) are dictated by the constraints in *MAXFLOW(D,D,L)*. We denote the upper bound on arc (i, j) by $\kappa(i, j)$. So we have:

$$\kappa(s, h) = LS_h - L \quad \forall\, h \in\ O_r$$

$$\kappa(p, t) = LS_p - L \quad \forall\, p \in\ O_c$$

For a feasible flow vector $\mathbf{f}$, an *augmenting path* is a simple path from s to t that can be used to increase flow from s to t. Note that this path is not necessarily directed. On forward arcs in this path ((i, j) points in the direction $s \rightarrow t$) the flow f_{ij} must satisfy $0 \leq f_{ij} < \kappa_(i, j)$, and on backward arcs, i.e. (i, j) is reverse, the flow must satisfy $0 < f_{ij} \leq \kappa(i, j)$.

Ford and Fulkerson presented a solution to the max-flow problem in 1954 [31]. The algorithm starts from an arbitrary feasible flow. In subsequent iterations, the Ford-Fulkerson algorithm identifies an augmenting path, and augments the flow. If the augmenting path is denoted as a set of arcs $\{a_1, a_2, ..., a_k\}$, then the flow augmentation possible is $\delta = \min_{1 \leq i \leq k} \delta(a_i)$, where $\delta(a_i) = \kappa_{a_i} - f_{a_i}$ for forward arcs and $\delta(a_i) = f_{a_i}$ for backward arcs. The flow is adjusted using $f_{a_i} \leftarrow f_{a_i} + \delta$ on forward arcs and on backward arcs using $f_{a_i} \leftarrow f_{a_i} - \delta$. The algorithm iterates until no augmenting path exists, upon which the maximum flow is obtained, as specified by the following theorem:

Theorem 5. *Ford-Fulkerson [31]: Flow* $\mathbf{f}$ *is maximum in graph* $\mathcal{G}$ *if and only if there is no augmenting path in* $\mathcal{G}$ *bearing flow* $\mathbf{f}$.

When there are no lower bounds on capacity, the flow $\mathbf{f}$ defined by $f_{ij} = 0\ \forall (i, j) \in \mathcal{A}$ (the set of arcs in the network) is feasible and can be used to initialize the Ford-Fulkerson algorithm. There are numerous methods for searching for augmenting paths; techniques include shortest path (fewest number of edges) and fattest path (maximum bottleneck capacity along the path) algorithms [32]. Note that the solution to the maximum flow problem (and hence also *MAXFLOW(D,D,L)*) is in general not unique.

To form a Minimum Rejection Algorithm, we first use the Ford-Fulkerson algorithm to identify A. Subsequently we set $D \leftarrow D - A$ and apply FMA to the resultant D. As described in Section 5.1, FMA processes overflowing lines sequentially, adjusting the demand on the line so that it sums to L (thereby identify a line of the rejection matrix). Since we have constructed A so that after modification $D(h, p) = 0$ at any intersection point of overflowing lines h and p, when FMA adjusts one of the overflowing lines it does not affect any other overflowing line. This means that after FMA has been applied, it has generated a Q that satisfies the requirements of Theorem 1. In the process, FMA has developed a schedule S that performs the decomposition $D = D' + D''$, where $D'' = A + Q$. The combined Minimum Rejection Algorithm is specified in Algorithm 2.

Algorithm 2 Minimum Rejection Algorithm

1: Apply the Ford-Fulkerson algorithm to solve $A =$*MAXFLOW(D,D,L)*.
2: Set $D \leftarrow D - A$.
3: Apply FMA to D to generate Q and a schedule S.

6 Simulation Performance

In this section we report the results of simulations of the scheduling approaches performed using OPNET Modeler [33]. We performed simulations on a 16 edge-node star topology network. The links in the network

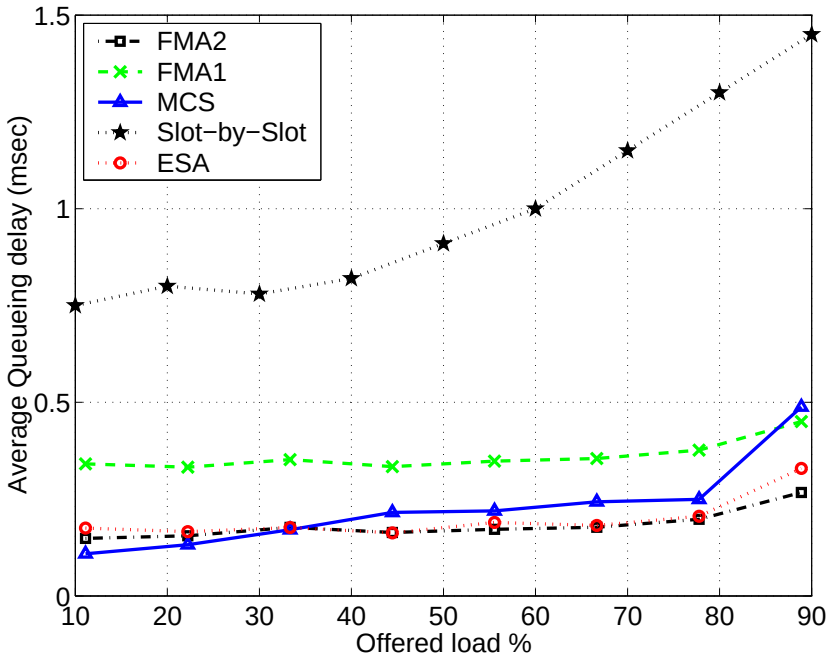

Figure 5. Average queuing delay performance achieved by FMA1, FMA2 , ESA (Equal Share Matching), Slot-by-Slot and MCS under non-uniform, Poisson traffic.

have capacity 10 Gbps and the distance between each edge node and the optical switch is 5 msec. A time slot is of length 10 μsec, and a frame has a fixed length of 1 msec (or 100 slots). Every experiment was run for a duration of 0.5 sec (equal to 500 frame durations) and the results were averaged over 5 repetitions of the simulations. The virtual output queues in the simulations have fixed buffer size (90000 packets). Whenever the buffer is full, arriving packets are dropped.

Comparison between FMA, ESA, MCS, and Slot-by-Slot under Non-uniform Traffic. In the simulations, traffic sources inject traffic at rates up to 10 Gbps into the edge nodes. The arrival distribution of the data packets is Poisson and the size distribution is exponential with mean size of 1000 bits. Then multiple (approximately 100) packets are wrapped into one optical slot. We investigated two cases of destination distributions: (i) a uniform case, where sources send equal amounts of traffic to each destination, and (ii) a non-uniform case, where all destinations receive an equal amount of traffic on average, but each source sends 5 times as much traffic to one destination. The frame-based scheduling algorithms compute the schedule ahead of time based on the predicted traffic of 10 msec (round-trip delay) in future. In the first set of simulations we used the average of the traffic arrivals over the past 10 frame durations to form the prediction of the demand matrix D.

FMA and ESA use the *EXACT* algorithm, which collocates most of the allocations for a particular source-destination pair in an attempt to minimize switch reconfigurations. This concentration has the impact of increasing average waiting time of packets. However this effect is considerably reduced if we distribute similar matchings in two different locations in the frame. In our simulations FMA1 collocates similar matchings (applying *EXACT* in a standard fashion) and FMA2 and ESA separate them into two batches, one placed towards the start of the frame and one towards the end. We compare performance to two previous algorithms: Minimum Cost Search (MCS) [7] and a slot-by-slot scheduling approach based on PIM (Parallel Iterative Matching) [6].

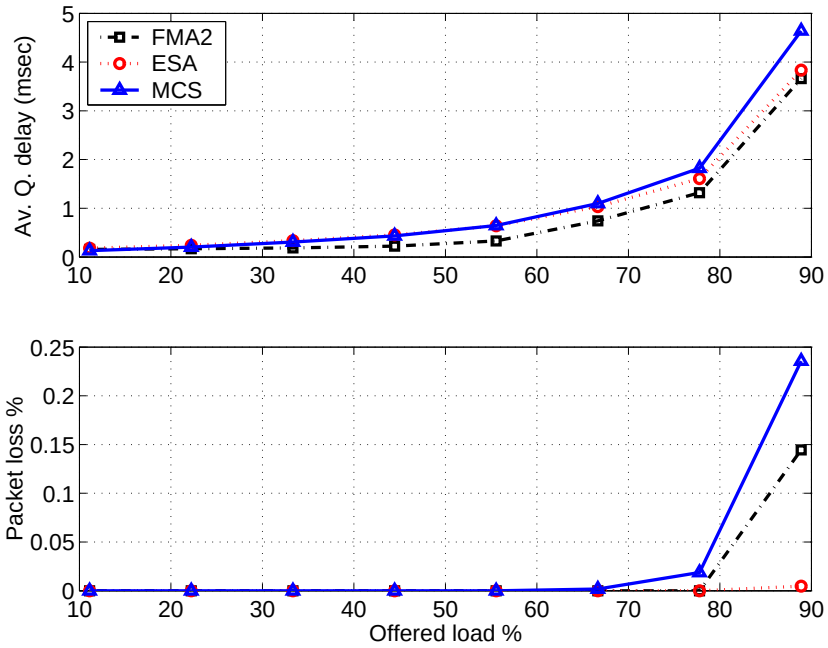

Figure 6. Average queuing delay and packet loss performance for FMA2, ESA and MCS under bursty traffic and non-uniform distribution of the destinations.

Figure 5 shows the queuing delays over a wide range of offered load, from 10% to 90% link capacity under nonuniform traffic (uniform traffic gives similar results). The slot-by-slot algorithm has large average queuing delays, since it is more appropriate for metro and local-area networks [6]. FMA1 generates additional average delay compared to FMA2, which is due to the collocation of matchings. ESA, FMA2 and MCS exhibit similar performance, achieving low average delays under all but the highest load. Under higher loads, the performance of MCS deteriorates due to the additional blocking it induces. On average the percentage of blocking generated by MCS is 0.9%. The matching algorithms (FMA and ESA) generate 0.02% blocking (due to natural blocking in the demand matrices). When the load is high, FMA2 assigns more time slots to the heavier connections, which can use the extra time slots more efficiently. ESA assigns the same number of extra time slots to each connection irrespective of its load. In this scenario only the slot-by-slot scheduling algorithm experiences packet loss (up to 0.31% for loads exceeding 70% of capacity).

Comparison between FMA2, ESA, MCS under Bursty Traffic. We also performed simulations with bursty traffic using on/off traffic sources. Every edge node is equipped with 6 on/off sources. The "on" and "off" periods have Pareto distributions with $\alpha = 1.9$. The mean of the "off" periods is 5 times greater than the mean of the "on" periods. During "on" periods the sources generate packets with an average rate equal to the full link capacity (10 Gbps). The rate distribution is exponential. Figure 6 depicts queuing delays and packet losses for the FMA2, ESA and MCS algorithms. FMA2 demonstrates marginally superior average queuing delay performance compared to the other two algorithms (0.3-0.9 msec less when the load exceeds 50%). Under offered loads greater than 80% of capacity, packet loss occurs as a result of traffic bursts overflowing the network. At 90% load, MCS generates 0.24% loss, FMA2 generates 0.14% loss, and ESA does not generate any packet loss. The loss generated by FMA2 is due to insufficient allocation of additional slots to temporarily low-rate connections that experience a sudden increase in traffic arrivals when they enter an "on" period. ESA allocates extra slots irrespective of demand so eliminates this loss at the cost of additional average delay.

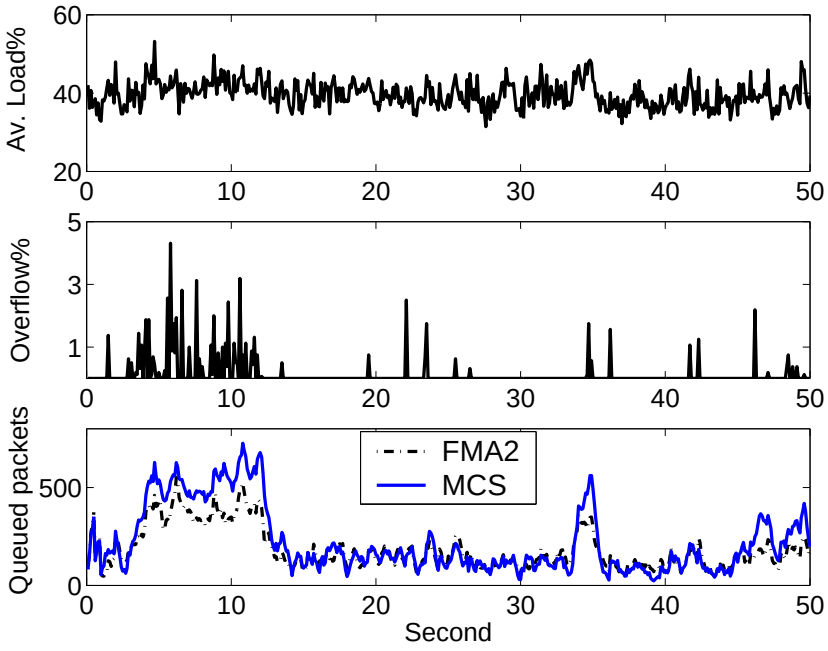

Figure 7. The behaviour of FMA2 and MCS in response to traffic loads derived from Internet traces. The upper panel shows the offered load averaged over all source-destination pairs. The middle panel shows the percentage of overflow traffic. The lower panel shows the overall number of queued packets at the edge nodes.

Comparison between FMA2 and MCS under Real Traffic. We also explored the performance of our algorithms using traffic derived from empirical Internet measurements. We used 50 seconds of packet traces captured from an OC3 link at Colorado State University [34]. The flows were divided into 16 components based on IP source/destination addresses, and each component served as one of the edge nodes. Using auto-regressive flow-based prediction [35], we predicted the traffic demand 1 second ahead (assuming 1 second round-trip and scheduling delay) and applied the scheduling algorithm for the predicted traffic demand matrix. We used a more sophisticated prediction technique for this simulation scenario because of the inadequate performance of the simple linear predictor (moving average method) used in the previous simulations. We considered a frame of length 0.1 seconds (equal to 100 time slots of 1 msec.) and for simplicity assumed that each packet fits one time slot completely. We performed simulations for 50 seconds. The average offered load was around 40%; under this load, MCS and FMA are expected to perform similarly if the traffic is admissible. The derived traffic is such that the demand is inadmissible for a duration of 10 seconds (from 2–12 seconds), because one of the edge nodes is overloaded. Growth in the queue sizes is unavoidable during this period. Figure 7 shows the total number of queued packets at the edge nodes. FMA2 and MCS adapt to the variations of the arrivals in a very similar fashion, but FMA2 has a lower number of queued packets because it does not induce blocking.

Comparison between FMA2 and MRA under Bursty Traffic. We performed simulations with bursty traffic using on/off traffic sources. Every edge node is equipped with 6 on/off sources. The "on" and "off" periods have Pareto distributions with $\alpha = 1.9$. The mean of the "off" periods is 5 times greater than the mean of the "on" periods. During "on" periods the sources generate packets with an average rate up to the full link capacity (10 Gbps). The rate distribution is exponential. The demand matrix has a non-uniform distribution; each destination receives on average the same amount of traffic, but each source sends five

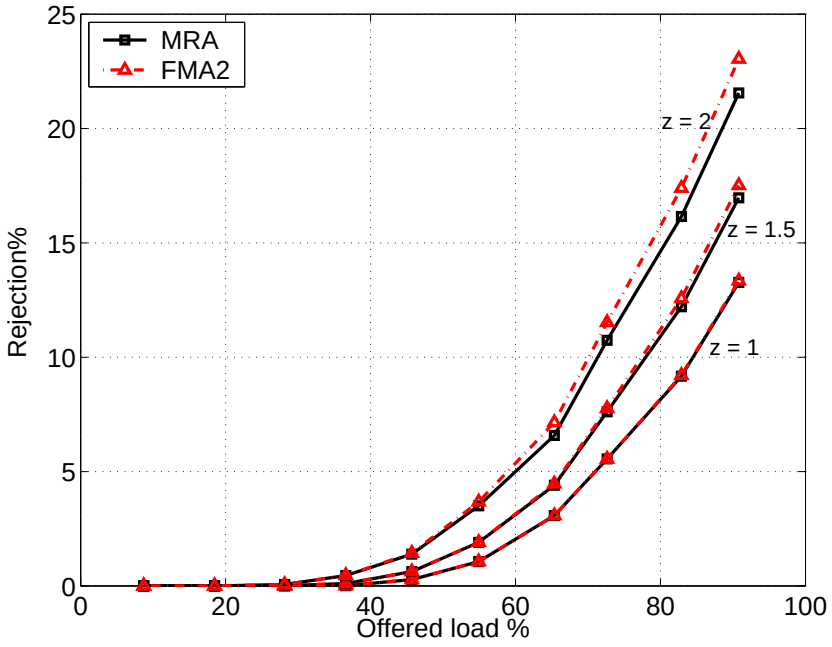

Figure 8. Comparison between the rejection obtained by FMA2 and MRA under varying offered load for different factors of imbalanced load (z). Traffic is bursty (generated by on-off sources) and has uniform distribution, aside from the impact of z.

times as much traffic to one specific destination as compared to the others.

Since the behaviour of MRA and $FMA2$ only differs when there are critical elements in the demand matrix, we investigate scenarios where critical demands are likely to exist. In order to do this, in each frame we choose one arbitrary source i and one arbitrary destination j. Each source generates z times as many packets for destination j compared to other destinations. Similarly source i generates z times as many packets (to all destinations) as any other source. As z increases, the elements of the demand matrix corresponding to these two edge nodes are more likely to be critical connections; the demand element D_{ij} has even higher likelihood of being critical.

Figure 8 compares the percentage of rejected demand achieved by FMA2 and MRA as the offered load changes for various values of z. At high load (greater than 70%) with $z = 2$, there are numerous critical elements and MRA begins to achieve less rejection than FMA2. The discrepancy is still only 2 percent at 90% load. Figure 9 compares the maximum percentage rejection experienced by any demand when scheduling is performed by FMA2 and MRA. As the offered load increases, MRA concentrates rejection on the critical elements; the maximum percentage rejection is thus much (up to 25 percent) higher than that achieved by FMA2, which distributes rejection fairly amongst all competing connections. Figure 10 compares the average end-to-end delay experienced by packets when scheduling is performed using FMA and MRA; the approaches yield similar average delay.

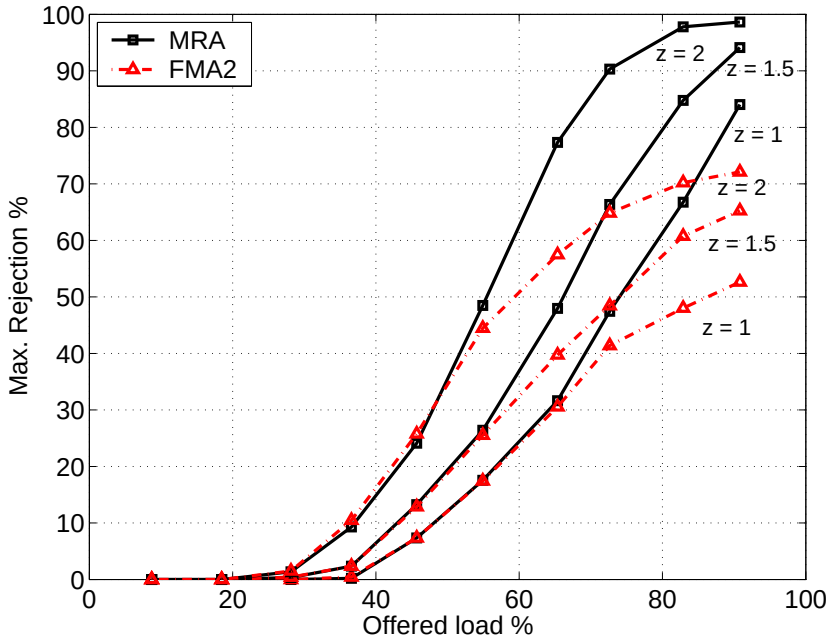

Figure 9. Comparison between the maximum percentage rejection experienced by any demand after scheduling by MRA and FMA2 for different values of z and varying offered load.

7 Conclusion and Future Work

We have formulated the bandwidth allocation problem in the AAPN network as a scheduling problem with the objective of minimizing rejection whilst reducing the number of switch reconfigurations. We proposed a novel scheduling algorithms Fair Matching Algorithm (FMA), that achieve zero rejection for admissible demands and provides (weighted) max-min fair allocation of free capacity. The max-min fairness criterion does not in general achieve minimum rejection when the traffic is not admissible.

We demonstrated that when the demand matrix is inadmissible, the Fair Matching Algorithm minimizes the maximum percentage rejection experienced by any connection. We also proposed a novel algorithm (MRA) that generates a schedule that minimizes the total rejection of demand. Simulations indicate that the discrepancy in total rejection achieved by MRA and FMA is relatively minor, whereas there is a major difference in the fairness of the allocation of rejection. In addition, MRA appears to be less robust to demand prediction errors (when traffic arrivals differ substantially from the demand matrix used for scheduling). Thus it appears that whilst MRA achieves minimum rejection schedules, FMA is a better choice for all-photonic scheduling in practice.

8 Appendix

8.1 Proof of Lemma 1

Proof of Lemma 1. We prove this lemma using a similar approach to that adopted in [27]. Suppose that v is weighted max-min fair with the weight vector ω. To arrive at a contradiction, assume that there exists a connection u with no bottleneck link. Then for each link ℓ crossed by u for which $C_\ell = F_\ell$, there must exist

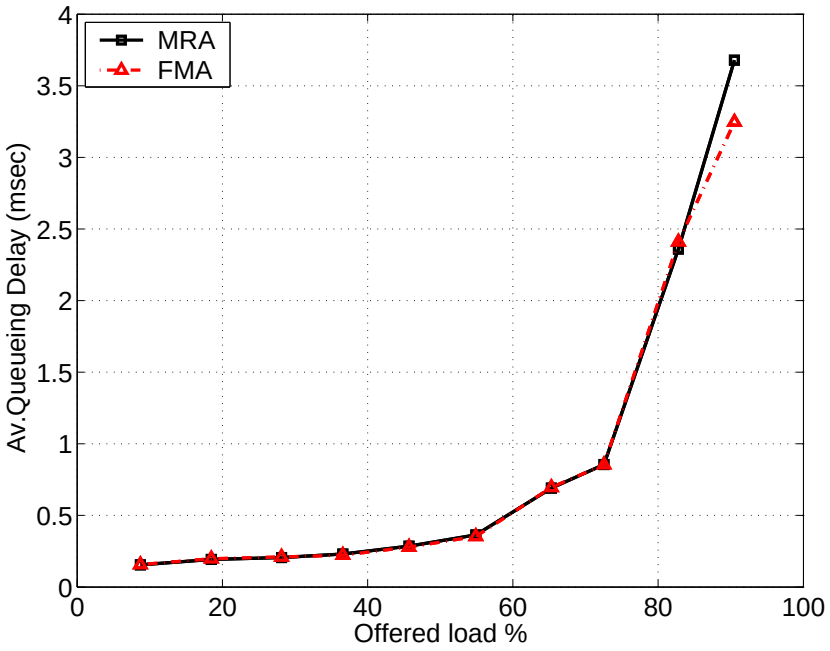

Figure 10. Average queuing delay performance achieved by MRA and FMA2 for varying offered load and $z = 2$.

a connection $x \neq u$ such that $\omega_x > \omega_u$; thus the quantity

$$\delta_\ell = \begin{cases} C_\ell - F_\ell & if \quad F_\ell < C_\ell \\ (\omega_x - \omega_u) \times R_x & if \quad F_\ell = C_\ell \end{cases} \tag{16}$$

is positive. Therefore, by increasing υ_u by the minimum δ_ℓ over all links ℓ crossed by u, while decreasing by the same amount the rates of the connections x of the links ℓ crossed by u with $F_\ell = C_\ell$, we maintain feasibility without decreasing the rate of any connection k with $\omega_k \leq \omega_u$; this contradicts the weighted max-min fairness property of (υ, ω). Note that $\upsilon_x - \min(\delta_\ell)$ is always positive.

Conversely, assume that each connection has a bottleneck link with respect to the feasible set (υ, ω). Then to increase the rate of any connection u while maintaining feasibility, we must decrease the rate of some connection k crossing bottleneck link ℓ of u (because we have $F_\ell = C_\ell$ by the definition of a bottleneck link). Since $\omega_k \leq \omega_u$ for all k crossing ℓ (by the definition of a bottleneck link), the feasible set (υ, ω) satisfies the requirement for weighted max-min fairness. □

8.2 Proof of Theorem 1

Proof. Consider the set of schedules that achieve minimum $N_s(S) = N_s^*$ and label the schedule within this set that achieves minimum rejection S_a. The minimum achievable rejection is no larger than $REJ(S, D, L) = \max(||D||_1 - L, 0)$, where $||D||_1 = \sum_i \sum_j D_{ij}$ (at least one demand element must be satisfied each time-slot). Thus $C(S_a) \leq \max(||D||_1 - L, 0) + gN_s^*$. Now consider schedules that increase the number of switch reconfigurations to $N_s(S) = N_s^* + 1$ and suppose that one of these, S_b, achieves zero rejection, so that $C(S_b) = g(N_s^* + 1)$. The differential in cost $C(S_b) - C(S_a) \geq g - \max(||D||_1 - L, 0)$. If

$g > \max(||D||_1 - L, 0)$, then this difference is strictly positive and any schedule solving *PROBLEM 1* lies within the set of schedules that achieve minimum N_s.

In order to prove that the problem is NP-hard for this range of g, we consider *PROBLEM 2*, which for very large values of τ is reduced to minimizing the schedule length subject to the constraint that N_s is minimum. Gopal et al. prove that this problem, which they refer to as the MINSWT problem, is NP-complete [11].

Suppose we had a deterministic polynomial algorithm called *solve-G(D,L)* that could solve *PROBLEM 1* for the identified range of g for demand matrix D and a frame of length L. We could then define the algorithm Solve-MINSWT (Algorithm 2).

Algorithm 3 Solve-MINSWT

```
L = 1;
S = solve-G(D,L);
while REJ(S, D, L) > 0 do
   L = L + 1;
   S = solve-G(D,L);
end while
```

Upon termination of this algorithm, the identified schedule S is guaranteed to have the minimum number of switch reconfigurations (as argued above). Since it is also the minimum length schedule that achieves $REJ(S, D, L) = 0$ it is also a solution to *PROBLEM 2* and hence the MINSWT problem. Algorithm 2 is thus a deterministic polynomial algorithm to solve the MINSWT problem. Therefore, solving *PROBLEM 1* for the considered range of g is as hard as solving MINSWT (and any other problem in NP) and hence is NP-hard. □

8.3 Proof of Theorem 2

Proof. Let $u \in \{(i, j), 1 \leq i, j \leq N\}$ index the source-destination connections specified by the demand matrix. We focus on the properties of the modified demand matrix and associated sets at various iterations of the while loop in Algorithm 1, so we index entities by iteration number and note that this indicates the value of the entity at the *start* of the iteration. For example, $\mathcal{A}_D(h)$ denotes the set of unmodified lines at the start of iteration h of the algorithm.

We prove that FMA achieves weighted max-min fair allocation of the demands. During each iteration h of the while-loop, FMA identifies the line $\gamma \in \mathcal{A}_D(h)$ such that $G_\gamma(h) = \min\{G_\ell(h); \ell \in \mathcal{A}_D(h)\}$. It alters the demands in $a_\gamma(h)$ according to (11) and after this modification, there is no subsequent modification of these demands. Substituting (11) into the definition of the weight, we have $\omega_u = 1 + G_\gamma(h)$ for all $u \in a_\gamma(h)$.

We demonstrate that the adjustment at iteration h leads to γ being a bottleneck link (line) for $u \in a_\gamma(h)$, i.e., after this adjustment it holds that $\omega_z \leq \omega_u$ for $u \in a_\gamma(h)$ and $z \in b_\gamma(h)$. Equivalently, we prove that $\min\{G\}$ is monotonically increasing with respect to the iteration number, i.e., $\min\{G(h)\} \leq \min\{G(h + 1)\}$. The equivalence follows since the ω_z are obtained from adjustments prior to iteration h.

Suppose that line β has minimum G at iteration $h + 1$. Lines γ and β have at most one connection (demand) in common. If there is no common connection, then $G_\beta(h + 1) = G_\beta(h) \geq G_\gamma(h)$. If there is a

common connection k, then:

$$LS_\beta(h+1) = LS_\beta(h) + D_k(\omega_k - 1) \quad (17)$$

$$S_{a_\beta}(h+1) = S_{a_\beta}(h) - D_k \quad (18)$$

and hence

$$\begin{aligned} G_\beta(h+1) &= \frac{L - LS_\beta(h) - D_k(\omega_k - 1)}{S_{a_\beta}(h) - D_k} \\ &= \frac{S_{a_\beta}(h)G_\beta(h) - D_k(\omega_k - 1)}{S_{a_\beta}(h) - D_k} \\ &\geq G_\gamma(h) \end{aligned} \quad (19)$$

where the last inequality follows from substitution based on $G_\beta(h) \geq G_\gamma(h) = \omega_k - 1$.

Thus the application of FMA upon an arbitrary demand matrix D leads to the generation of a bottleneck link for each connection u with weight $\omega_u = \frac{D'_u}{D_u}$. By Lemma 1, this establishes that FMA achieves weighted max-min fair allocation of adjusted demands D'.

□

8.4 Proof of Theorem 3

Proof. The proof is similar to the proof of Theorem 2. In this case, we have $\omega_u = D_u - D'_u$ and $\omega_u = H_\gamma(h)$ for all $u \in a_\gamma(h)$, where γ is the line with minimum H at iteration h. If β is the line with minimum H at iteration $h+1$, then if γ and β share an element k, $LS_\beta(h+1) = LS_\beta(h) + \omega_k$ and $a_\beta(h+1) = a_\beta(h) - 1$. Hence:

$$H_\beta(h+1) = \frac{L - LS_\beta(h) - \omega_k}{|a_\beta(h)| - 1} \quad (20)$$

$$= \frac{|a_\beta(h)|H_\beta(h) - \omega_k}{|a_\beta(h)| - 1} \quad (21)$$

$$\geq H_\gamma(h) \quad (22)$$

where the last inequality follows from substitution $H_\beta(h) \geq H_\gamma(h) = \omega_k$. Hence, ESA leads to a bottleneck link with weight $\omega_u = D_u - D'_u$ and hence, by Lemma 1, achieves max-min fair allocation of capacity. □

8.5 Proof of Theorem 4

Proof. For the *MAXFLOW(D,X,L)* problem, define the following capacity bounds:

$$\mu_h(X) = \min(\sum_{p \in O_c} X_{hp}, r_h(D) - L) \quad (23)$$

$$\mu_p(X) = \min(\sum_{h \in O_r} X_{hp}, c_p(D) - L) \quad (24)$$

The optimization in the *MAXFLOW* problem generates a matrix Y such that:

$$|Y| = \sum_h \sum_p Y_{hp} = \min(\sum_p \mu_p(X), \sum_h \mu_h(X)). \quad (25)$$

This is an application of the max-flow min-cut theorem [36] (see Figure 4).

Consider an arbitrary rejection matrix D^w and set B =*MAXFLOW*(D, D^w, L). Then we can write $D^w = B + Q$ where Q is a non-negative matrix. Now consider the conditions necessary for D^w to achieve minimum rejection. First, $D^w_{hp} = 0$ if $h \notin O_r$ and $p \notin O_c$ (any non-zero values constitute unnecessary rejection).

Without loss of generality, suppose that $\sum_p \mu_p(D^w) < \sum_h \mu_h(D^w)$. Then for each row $h \in O_r$, $\sum_p B_{hp} = \mu_p(D^w)$. This implies that row h either achieves its required rejection solely from B (i.e., $r_h(B) = r_h(D) - L$), or that $B_{hp} = D^w_{hp}$ for all $p \in O_c$. In the latter case, D^w must contain additional rejection (positive entries) on row h at the *non-critical connections*. If D^w is to achieve minimum total rejection, $r_h(Q) = r_h(D) - L - r_h(B)$.

Now consider the columns of D^w. After the generation of B, the rejection on column p is $c_p(B)$. Then for minimum rejection we require that $c_p(Q) = c_p(D) - L - c_p(B)$. Note that if $r_h(B)$ does not satisfy the rejection requirements of row h, then $Q_{hp} = 0$. Thus, no positive elements of Q contribute to required rejection on both a row h and a column p.

Based on this discussion, if D^w achieves minimum rejection, we can express its rejection $|D^w|$ as:

$$\begin{aligned} |D^w| &= \sum_h \sum_p (B + Q) \\ &= |B| + \sum_{h \in O_r} (r_h(D) - L - r_h(B)) \\ &\qquad + \sum_{p \in O_c} (c_p(D) - L - c_p(B)) \\ &= \sum_{h \in O_r} (r_h(D) - L) + \sum_{p \in O_c} (c_p(D) - L) - |B| \end{aligned} \tag{26}$$

Therefore, in order for D^w to achieve minimum rejection, $|B|$ must be maximized (the first two terms are functions solely of D and L). Compare the solutions B = *MAXFLOW*(D, D^w, L) and A = *MAXFLOW*(D, D, L). Since $D^w_{hp} \leq D_{hp}$ for any (h, p), the constraints in the second problem are looser, which implies that $|A| \geq |B|$, irrespective of the particular values in D^w. Note that A is also a solution to *MAXFLOW*(D, A, L).

Hence if we ensure that $D^w_{hp} \geq A_{hp}$ for all (h, p), we derive $|B| = |A|$, which implies that $|B|$ attains its maximum value (and hence $|D^w|$ is the minimum rejection). We can thus construct a rejection matrix that achieves minimum rejection by solving A = *MAXFLOW*(D, D, L), and setting $D'' = A + Q$, where Q satisfies the constraints specified in the theorem. If a schedule S decomposes the demand into an allocated matrix D' and this rejection matrix D'', then it achieves minimum rejection. □

References

[1] L. Xu, H.G. Perros, and G. Rouskas, "Techniques for optical packet switching and optical burst switching," *IEEE Comm. Mag.*, vol. 39, no. 1, pp. 136–142, Jan. 2001.

[2] R. Ramaswami and K.N. Sivarajan, "Routing and wavelength assignment in all-optical networks," *IEEE/ACM Trans. Networking*, vol. 3, no. 5, pp. 489–500, Oct. 1995.

[3] G.V. Bochmann, M.J. Coates, T. Hall, L.G. Mason, R. Vickers, and O. Yang, "The agile all-photonic network: An architectural outline," in *Proc. Queens' Biennial Symp. Comms.*, Kingston, Canada, June 2004.

[4] L.G. Mason, A. Vinokurov, N. Zhao, and D. Plant, "Topological design and dimensioning of agile all photonic networks," *Computer Networks*, vol. 50, no. 2, pp. 268–287, Feb. 2006.

[5] I. Keslassy, M. Kodialam, T.V. Lakshman, and D. Stiliadis, "Scheduling schemes for delay graphs with applications to optical packet networks," in *Proc. IEEE Work. High Perf. Switch. and Routing*, Phoenix, AZ, Apr. 2003.

[6] X. Liu, N. Saberi, M.J. Coates, and L.G. Mason, "A comparison between time-slot scheduling approaches for all-photonic networks," in *Int. Conf. on Inf., Comm. and Sig. Proc (ICICS)*, Bangkok, Thailand, Dec. 2005.

[7] N. Saberi and M.J. Coates, "Bandwidth reservation in optical WDM/TDM star networks," in *Proc. Queens' Biennial Symp. Comms.*, Kingston, Canada, June 2004.

[8] A. Ganz and Y. Gao, "A time-wavelength assignment algorithm for a WDM star network," in *Proc. IEEE Infocom*, Florence, Italy, 1992.

[9] A.Ganz and Y.Gao, "Efficient algorithms for SS/TDMA scheduling," *IEEE Trans. Comm.*, vol. 40, pp. 1367–1374, August 1992.

[10] C. A. Pomalaza-Raez, "A note on efficient SS/TDMA assignment algorithms," *IEEE Trans. Comm.*, vol. 36, pp. 1078–1082, 1988.

[11] I. S. Gopal and C. K. Wong, "Minimizing the number of switchings in an SS/TDMA system," *IEEE Trans. Comm.*, vol. 33, pp. 1497–1501, June 1985.

[12] B. Towles and W. J. Dally, "Guaranteed scheduling for switches with configuration overhead," *IEEE/ACM Trans. Networking*, vol. 11, pp. 835–847, October 2003.

[13] T. Gonzalez and S. Sahni, "Open shop scheduling to minimize finish time," *J. ACM*, vol. 23, pp. 665–679, Oct. 1976.

[14] P. Crescenzi, X. Deng, and C. H. Papadimitriou, "On approximating a scheduling problem," *J. Combinatorial Optimization*, vol. 5, pp. 287–297, 2001.

[15] G. Bongiovanni, D. Coppersmith, and C.K. Wong, "An optimal time slot assignment algorithm for an SS/TDMA system with variable number of transponders," *IEEE Trans. Comm.*, vol. 29, pp. 721–726, Oct. 1981.

[16] I.S. Gopal, G. Bongiovanni, M. A. Bonuccelli, D. T. Tang, and C. K. Wang, "An optimal switching algorithm for multibeam satellite systems with variable bandwidth beams," *IEEE Trans. Comm.*, vol. 30, pp. 2475–2481, Nov. 1982.

[17] T. Inukai, "An efficient SS/TDMA time slot assignment algorithm," *IEEE Trans. Comm.*, vol. 27, pp. 1449–1455, May 1979.

[18] R. L. Graham, "Bounds on multiprocessing timing anomalies," *SIAM J. Applied Mathematics*, vol. 17, pp. 416–429, March 1969.

[19] R. L. Graham, E. L. Lawler, J. K. Lenstra, and K. Rinnooy Kan, "Optimization and approximation in deterministic scheduling: A survey," *Ann. Disc. Math.*, pp. 287–326, 1979.

[20] D. McLaughlin, S. Sardesai, and P. Dasgupta, "Preemptive scheduling for distributed systems," in *Proc.11th Int. Conf. Parallel and Distributed Computing Systems*, Chicago, Illinois USA, Sept. 1998.

[21] K. Jansen and M.I. Sviridenko, "Polynomial time approximation schemes for the multiprocessor open and flow shop scheduling problem," in *27th International Colloquium on Automata, Languages and Programming*, Geneva, Switzerland, July 2000, pp. 878–889.

[22] K. Jansen, R. Solis-Oba, and M.I. Sviridenko, "A linear time approximation scheme for the job shop scheduling problem," in *Proc. Third Inter. Workshop Approximation Algorithms for Combinatorial Optimization Problems*, Aug. 1999, pp. 177–188.

[23] C.H. Papadimitriou and K. Steiglitz, *Combinatorial optimization: algorithms and complexity*, Prentice-Hall, 1982.

[24] S. Micali and V. V. Vazirani, "An $O(\sqrt{\|V\|}\|E\|)$ algorithm for finding maximum matching in general graphs," in *Proc. IEEE Symp. on Found. Comp. Sci.*, Syracuse, NY, 1980, pp. 17–27.

[25] R.M. Karp, "Reducibility among combinatorial problems," in *Proc. Complexity of computer computations*, R.E. Miller and J.W. Thatcher, Eds., New York, NY, 1972, pp. 85–103, Plenum Press.

[26] Ed. D. Hochbaum, *Approximation Algorithms for NP-Hard Problems*, PWS Publishing Company, Boston, MA, 1996.

[27] D. Bertsekas and R. Gallager, *Data Networks*, Prentice Hall, Englewood Cliffs, NJ, 1992.

[28] P. Marbach, "Priority service and max-min fairness," *IEEE/ACM Trans. Networking*, pp. 733–746, Oct. 2003.

[29] A. Charny, D. Clark, and R. Jain, "Congestion control with explicit rate indication," in *Proc. ICC*, Seattle, WA, Jun. 1995.

[30] M.A. Marsan, A. Bianco, E. Leonardi, F. Neri, and A. Nucci, "Simple on-line scheduling algorithms for all-optical broadcast-and select networks," *IEEE European Trans. Telecom.*, vol. 11, no. 1, pp. 109–116, Jan. 2000.

[31] L. R. Ford, Jr., and D. R. Fulkerson, "Maximal flow through a network," *Canadian. J. Math.*, pp. 399–404, 1956.

[32] J. Edmonds and R. M. Karp, "Theoretical improvements in algorithmic efficiency for network flow problems," *J. Assoc. Comput. Mach.*, pp. 248–264, 1972.

[33] "OPNET modeler 10.5," http://www.opnet.com.

[34] "Passive measurement and analysis (PMA) project," http://pma.nlanr.net/Traces/Traces/daily.

[35] T. Ahmed and M.J. Coates, "Predicting flow vectors," Tech. Rep., McGill University, Montreal, Canada, Sept. 2005, available at http://www.tsp.ece.mcgill.ca/Networks/publications.html.

[36] P. Elias, A. Feinstein, and C. E. Shannon, "Note on maximum flow through a network," *IRE Transactions on Information Theory IT-2*, pp. 117–119, 1956.