

Multihop Packet Scheduling in WDM/TDM Networks with Nonnegligible Transceiver Tuning Times

Marco Ajmone Marsan, *Fellow, IEEE*, Andrea Bianco, *Member, IEEE*, Emilio Leonardi, *Member, IEEE*, Fabio Neri, *Member, IEEE*, and Antonio Nucci, *Student Member, IEEE*

Abstract—This paper addresses the design of packet transmission schedules in photonic slotted wavelength-division multiplexing/time-division multiplexing broadcast-and-select networks with W wavelengths and N nodes. Nodes are equipped with one tunable-wavelength transmitter with nonnegligible tuning times and one fixed-wavelength receiver. A new scheduling algorithm that exploits multihop packet transfer to shorten the duration of scheduling periods is first proposed. A single-hop scheduling algorithm that performs slightly better than previous proposals is then described. A simulation-based analysis of the two algorithms shows that they jointly lead to significant improvements in both throughput and delay with respect to previous single-hop schedules.

Index Terms—Access protocols, communication systems performance, optical communications, photonic switching systems, scheduling, time-division multiplexing, wavelength-division multiplexing.

I. INTRODUCTION

ALL-OPTICAL broadcast-and-select networks that use wavelength-division multiplexing (WDM) as well as time-division multiplexing (TDM) techniques for the provision of integrated services over a packet-mode transport can transparently support several simultaneous very high-speed end-to-end communications, provided that adequate protocols are defined to dynamically share the available network resources.

In this paper, we consider all-optical WDM/TDM broadcast-and-select networks based on a broadcast topology (possibly a star) in which W wavelengths are available for node-to-node packet communications, and we assume that nodes are equipped with *one* full-duplex transceiver; hence, each node can be source and destination of at most one data flow at any given time. By tuning the nodes' transceivers, and by dynamically allocating the W available wavelengths, full connectivity is achieved among end users. WDM transmission

channels are assumed to be slotted and synchronized; each time slot can accommodate the transmission of one packet per wavelength.

The time required to tune the node transceivers is *not* assumed to be negligible with respect to the packet transmission time; in fact, with the presently available optical components, the tuning latency can be quite longer than the packet transmission time [1],[2].

A number of access protocol proposals for all-optical WDM/TDM broadcast-and-select networks appeared in the recent literature, some based on random access approaches (see [3] and references therein), some based on a deterministic scheduling of the transmissions from the different traffic sources [4]–[6].

While random access protocols are generally simple, but do not permit an efficient exploitation of the available bandwidth, protocols based on fast slot allocation algorithms can be rather efficient, but lead to a significant increase in the system complexity and to the waste of part of the network resources, which must be devoted to signaling; in addition, they require some extra processing capability within nodes to manage control information.

Simple and efficient protocols, i.e., protocols that lead to an efficient exploitation of the network resources while limiting the required hardware and software complexity, are however necessary to make all-optical broadcast-and-select networks a credible approach for very high-speed communications. Protocols that try to very rapidly adapt to traffic fluctuations generally require the exchange of a large quantity of control information and may end up being neither simple nor efficient. On the contrary, static WDM/TDM schedules can be simple, but quite inefficient, in the case of bursty traffic. As a compromise, protocols based on slowly varying WDM/TDM transmission schedules, which periodically adapt to slow changes of the traffic pattern, can provide a good tradeoff between simplicity and efficiency.

Moreover, since tuning times can be quite large with respect to transmission times, schedules that require retuning for each packet generally yield very limited throughput. As a consequence, schedules must consider sequences of transmissions that do not require retuning, hence at a fixed wavelength. Additionally, the algorithms that are used for the schedule computation must be as simple as possible, so that their execution can be quite fast.

The problem of finding an optimum transmission schedule for a given traffic pattern in presence of significant tuning laten-

Paper approved by D. P. Taylor, the Editor for Signal Design, Modulation and Detection of the IEEE Communications Society. Manuscript received March 12, 1998; revised October 7, 1999. This paper was presented in part at the IFIP TC-6 Second Working Conference on Optical Network Design and Modeling, Rome, Italy, February 1998, and in part at IEEE GLOBECOM'98, Sydney, Australia, November 1998.

The authors are with the Dipartimento di Elettronica, Politecnico di Torino, I-10129 Turin, Italy (e-mail: ajmone@polito.it; bianco@polito.it; leonardi@polito.it; neri@polito.it; nucci@polito.it).

Publisher Item Identifier S 0090-6778(00)03641-2.

cies was already extensively analyzed [7]–[9]. It can be formulated in the following way.

GIVEN a traffic request matrix R , whose elements r_{sd} are the numbers of packets that must be transmitted from any node s to any node d in a scheduling period,¹ FIND a time/wavelength assignment that guarantees the delivery of the requested traffic, while MIMIMIZING the time necessary to accommodate all transmissions, SUBJECT TO technological constraints.

In [7], this scheduling problem was shown to be NP-hard. Several heuristic approaches for the determination of good schedules were proposed in [8] and [9].

Observe that the scheduling algorithm does not aim at providing a schedule that is modified on the basis of packet-by-packet reservations issued by nodes. Conversely, the traffic request matrix R describes the (static or slowly varying) node traffic requirements based on “sustainable bandwidth” needs, which can be computed with techniques similar to the “equivalent bandwidth” expressions derived for connection acceptance control in asynchronous transfer mode networks [10].

Most of the scheduling algorithm proposals that appeared in the literature aimed at the minimization of the scheduling period duration, assuming that all packets must be transferred over a single-hop route (i.e., packets are directly transferred from their source to their destination with just one transmission). This may not necessarily be the best approach when tuning latencies are long.

In this paper, we present a new heuristic approach for the identification of a good schedule, which yields larger throughput than the schedules resulting from previous proposals. We show that significant throughput gains can be obtained by transmitting selected packet flows over a multihop route, i.e., by routing them through intermediate nodes. Since we aim at throughput maximization, we do not explicitly optimize delays, and we assume that, in the case of multihop transfers, the successive hops that a packet must travel are scheduled in successive scheduling periods. Nevertheless, we shall show that multihop schedules allow significant gains in the throughput-delay characteristics.

Multihop routing implies electronic buffering at intermediate nodes to temporarily store in-transit packets, i.e., packets that have reached an intermediate node and must be forwarded toward their final destination. Since the scheduling period duration is determined so that all transmission requests from all nodes can be satisfied, including all transmissions on multihop paths, the buffering requirements at each intermediate node are equal to the number of packets transmitted in a scheduling period.

Note that even if our algorithm could be adapted to the context of slowly varying traffic request matrices R , we will concentrate on fixed traffic matrices in this paper. The adaptation to a slowly varying traffic pattern can be trivially obtained by recomputing a new time/wavelength assignment either at fixed steps, i.e., every k scheduling periods, or whenever the difference between the “old” and the “new” traffic request matrices

exceeds a predefined threshold. More elaborate and efficient approaches can be devised, but their efficiency/complexity trade-offs require a careful investigation that is beyond the scope of this paper.

The problem of defining an optimal schedule (i.e., a schedule that maximizes the system throughput) when some traffic flows can be transmitted over multihop routes can be formulated as an integer linear programming (ILP) problem, hence shown to be NP-hard. We therefore propose a heuristic approach to the multihop scheduling problem, based on two separate steps. In the first step, the set of traffic flows that are conveniently transmitted over either single-hop or multihop routes is determined; for each multihop flow, the flow routing, i.e., the set of intermediate nodes traversed by the flow, is also defined. As a result of this first step, the single-hop traffic request matrix S is derived from R : each element s_{ij} of S is equal to the sum of the numbers of packets of the traffic flows that, according to the solution found, are to be transmitted over single-hop routes from node i to node j . In the second step, a single-hop scheduling algorithm is run in order to define the time/wavelength schedule for packet transmissions. The algorithm adopted in this paper for such second step represents itself a slight improvement over previously proposed algorithms.

II. MULTIHOP SCHEDULING PROBLEM

In this section, we first show that the multihop scheduling problem can be formulated as an ILP problem. Then, we present a heuristic algorithm that provides a good suboptimal solution to the multihop scheduling problem.

We consider a network with N nodes, each node being provided with one tunable transmitter and one fixed receiver. We denote by W the number of available wavelengths. The transmitters tuning latency, i.e., the time required to tune from a wavelength to a different one, is taken to be T slots. Thus, two transmissions from the same source at different wavelengths must be separated by at least T empty slots.

Although we consider in this paper the case of transmitter tunability, both the formulation of the problem, and the proposed heuristic, can easily be modified for the case of receiver tunability and extended to a more general context in which both transmitters and receivers are tunable with significant latencies.

A. Problem Formulation

We first provide the formulation of the scheduling problem assuming $W = N$, so that each wavelength leads to a different destination. Later we shall show how to extend the formulation to the case $W < N$.

Case $W = N$: The variables v^{ij} are used to describe the logical topology of the network, i.e., the pattern of logical links in the network, where i and j refer to node indices. A logical link $i \rightarrow j$ exists if packets are transmitted over a single-hop route from node i to node j

$$v^{ij} = \begin{cases} 1, & \text{if a logical link } i \rightarrow j \text{ exists} \\ 0, & \text{otherwise.} \end{cases}$$

¹Note that we will use in the paper both the terms scheduling period and frame with the same meaning.

The binary variables b_{ij}^{sd} describe the routing on the logical topology, where i, j, s , and d all refer to node indices

$$b_{ij}^{sd} = \begin{cases} 1, & \text{if the traffic flow } s \Rightarrow d \text{ is routed through} \\ & \text{link } i \rightarrow j \\ 0, & \text{otherwise.} \end{cases}$$

We assume that the whole traffic flow from node s to node d follows the same route, in order to guarantee the sequential delivery of packets to their destinations. This, together with the fact that traffic flows must travel over existing logical links and that flows must be conserved at each intermediate node, implies that the variables v^{ij} and b_{ij}^{sd} must satisfy the constraints stated below

$$\sum_i b_{id}^{sd} = 1 \quad (1)$$

$$\sum_j b_{sj}^{sd} = 1 \quad (2)$$

$$b_{ij}^{sd} \leq v^{ij} \quad (3)$$

$$\sum_i b_{ij}^{sd} = \sum_k b_{jk}^{sd}, \quad \text{if } j \neq s, j \neq d. \quad (4)$$

The binary variables $t^{ij}(l)$ describe the transmission activity, where superscripts i and j refer to node indices and l refers to time slot labels within a scheduling period

$$t^{ij}(l) = \begin{cases} 1, & \text{if node } i \text{ transmits to node } j \text{ in time slot } l \\ 0, & \text{otherwise.} \end{cases}$$

The variables $t^{ij}(l)$ must satisfy the following constraints, deriving from the fact that nodes can transmit and receive no more than one packet in each time slot, and from the tuning time constraint:

$$\sum_i t^{ij}(l) \leq 1 \quad (5)$$

$$\sum_j t^{ij}(l) \leq 1 \quad (6)$$

$$t^{ij_1}(l) + t^{ij_2}(l+k) \leq 1 \quad \forall k = 1, \dots, T; j_1 \neq j_2. \quad (7)$$

If all traffic requests in R must be satisfied, the variables $t^{ij}(l)$ and b_{jk}^{sd} are related by

$$\sum_l t^{ij}(l) = \sum_s \sum_d r_{sd} b_{ij}^{sd} \quad (8)$$

where the elements of the traffic request matrix R , r_{sd} represent the number of packets that must be transferred from node s to node d during the considered scheduling period.

Finally, we introduce the binary variables $e^i(l)$ to identify an upper bound to the scheduling period duration. Each variable $e^i(l)$ is a boolean indicator of the fact that node i has com-

pleted all its transmissions in the current frame and is ready to start transmission in the next frame, having already tuned to the wavelength to be used at the beginning of the next frame

$$e^i(l) = \begin{cases} 0, & \text{if node } i \text{ is ready to transmit in the next} \\ & \text{frame at slot } l \\ 1, & \text{otherwise.} \end{cases}$$

Using the variables $e^i(l)$ allows the linear expression of the function to be minimized.

The variables $e^i(l)$ must meet the following constraints:

$$e^i(l) \geq e^i(l+1) \quad (9)$$

$$e^i(l) \geq t^{ij}(l) \quad (10)$$

$$e^i(l+T-k+1) = 1, \quad \text{if } \exists j_1 \neq j_2 \text{ and} \\ \exists k, 1 \leq k \leq T : t^{ij_1}(k) + t^{ij_2}(l) > 1. \quad (11)$$

Note that inequalities (9) and (10) force $e^i(l) = 1$ for all the slots from the beginning of the scheduling period to the end of the last transmission by node i . Inequality (11) forces $e^i(l) = 1$ for an additional tuning time if not enough idle slots are available at the end of the scheduling period or at the beginning of the following one. An alternate expression for (11) is

$$e^i(l+T-k+1) \geq t^{ij_2}(l) - 1 + \sum_{j \neq j_2} t^{ij}(k). \quad (12)$$

With the above definitions, the criterion for the optimization of the multihop schedule is the minimization of the transmission period F satisfying

$$F \geq \sum_l e^i(l) \quad \forall i. \quad (13)$$

This minimization requires the solution of an ILP problem and, as a consequence, it is NP-hard.

Case $W < N$: The formulation presented for $W = N$ can be easily extended to consider cases where $W < N$. However, in order to do this, it is necessary to group all destination nodes that receive packets on the same wavelength into a set. Let C_{w_j} denote the set of destinations that receive packets on wavelength w_j .

Also in this case, the binary variables v^{iw_j} describe the logical network topology.² However, now i refers to source node indices, while w_j refers to sets of destination nodes, each set being associated with a different wavelength. A logical link $i \rightarrow C_{w_j}$ exists if packets are transmitted in single-hop fashion from node i to a node belonging to C_{w_j} .

Similarly, in the binary variables $b_{iw_j}^{sd}$ that describe the routing on the logical topology, i, s , and d refer to node indices, while w_j refers to sets of destinations.

Assuming again that the whole traffic flow from node s to node d follows the same route in the logical topology, the variables v^{iw_j} and $b_{iw_j}^{sd}$ must satisfy the following constraints [in-

²Note that we have used w_j as a superscript for the binary variable v instead of C_{w_j} in order to simplify the notation

stead of (4) and (1)], in addition to (3) and (2) (where the subscript j must be modified into w_j):

$$\sum_i b_{iw_j}^{sd} = \sum_k \sum_{j \in \mathcal{C}_{w_j}} b_{jw_k}^{sd}, \quad \text{if } s \notin \mathcal{C}_{w_j}, d \notin \mathcal{C}_{w_j} \quad (14)$$

$$\sum_i b_{iw_j}^{sd} \leq 1, \quad \text{if } d \in \mathcal{C}_{w_j}. \quad (15)$$

Also, in the binary variables $t^{iw_j}(l)$, the superscript i refers to node indices, while the superscript w_j refers to sets of destinations, such that

$$t^{iw_j}(l) = \begin{cases} 1, & \text{if node } i \text{ transmits to some destination} \\ & \text{in } \mathcal{C}_{w_j} \text{ in time slot } l \\ 0, & \text{otherwise.} \end{cases}$$

The variables $t^{iw_j}(l)$ must satisfy, instead of (5) and (7), the following constraints, in addition to (6) (where again the superscript j must be modified into w_j):

$$\sum_i t^{iw_j}(l) \leq 1 \quad \forall w_j = 1, \dots, W \quad (16)$$

$$\begin{aligned} t^{iw_{j_1}}(l) + t^{iw_{j_2}}(l+k) &\leq 1 & \forall k = 1, \dots, T \\ & & \forall i = 1, \dots, N \\ & & \forall w_{j_1}, w_{j_2} = 1, \dots, W \\ & & \text{with } w_{j_1} \neq w_{j_2}. \end{aligned} \quad (17)$$

Relation (8) among the variables $t^{iw_j}(l)$ and $b_{jw_k}^{sd}$ holds also in this case, as well as the definitions of the variables $e^i(l)$, in relations (9)–(12).

The optimality criterion defined by (13) remains the same.

B. The Heuristic Algorithm

Given the complexity of the multihop scheduling problem, we devised a heuristic algorithm that, given a traffic request matrix R , efficiently schedules packet transmissions on an all-optical WDM/TDM broadcast-and-select network.

For simplicity, our heuristic algorithm is presented in the paper in the case $W = N$; extensions to the case $W < N$ are straightforward.

Recall that the element r_{sd} of the traffic request matrix R represents the number of packets that must be transmitted from source s to destination d in a scheduling period. If we define $L(R)_i$ as the minimum duration of the activity of node i given a traffic request matrix R , then

$$L(R)_i = \max \left(\sum_j r_{ij} + K_i T, \sum_k r_{ki} \right) \quad (18)$$

where K_i is the number of destination nodes to which node i must transmit at least one packet. The length L_{sh} of the single-hop scheduling period (i.e., the period necessary to accommodate all transmission requests in R in single-hop fashion) must satisfy

$$L_{sh} \geq L(R) = \max_i L(R)_i. \quad (19)$$

Since we assume that the successive hops of a multihop transmission occur in successive scheduling periods, we can split in two separate steps the problem of defining a packet schedule that minimizes the scheduling period duration, hence maximizes the throughput, while allowing multihop transmissions.

The first step is aimed at the design of a logical topology through which packets must be routed from sources to destinations; i.e., at the definition of the set of single-hop transmissions, as well as the routing for multihop transmissions.

During the second step, the single-hop scheduling problem is solved starting from a derived traffic request matrix S , whose elements s_{ij} are equal to the numbers of packets that are to be transmitted in single-hop fashion from node i to node j , possibly comprising multihop packet flows, according to the solution found in the first step. A new heuristic single-hop scheduling procedure is proposed, which performs slightly better than previous proposals [8], [9]. It allows the achievement of scheduling period lengths very close to $L(R)$, for each single-hop traffic request matrix and for nonnegligible values of the tuning latency.

1) Logical Topology Design Algorithm: The heuristic algorithm for the logical topology design aims at finding a final single-hop traffic request matrix S such that $L(S)$ is minimized.

The logical multihop topology is built by subsequent rerouting of single-hop packet flows in multihop fashion. Let $S^{(n)}$ represent the single-hop scheduling traffic request matrix at step n , whose elements $s_{ij}^{(n)}$ are equal to the numbers of packets that are to be transmitted in single-hop fashion from node i to node j , possibly comprising multihop packet flows. At step n of the algorithm, one single-hop packet flow in $S^{(n)}$, e.g., the one from i to j ($i \Rightarrow j$) is considered, and an evaluation of the most convenient way to route the flow is attempted. The most convenient route, i.e., the route that leads to a minimization of $L(S^{(n+1)})$, is chosen for the transmission of packets belonging to flow $i \Rightarrow j$. The routes considered for flow $i \Rightarrow j$ are the single-hop route $i \rightarrow j$ and all the two-hop routes through some intermediate node k .

Define, for each source node i , the minimum length of the transmission schedule at step n

$$L_i^{(n)} = \sum_j s_{ij}^{(n)} + K_i T. \quad (20)$$

The logical topology design algorithm comprises the following steps.

- 1) Initialize $S^{(0)}$ to R : all connections are enabled (they take place in single-hop fashion).
- 2) Among all sources of enabled connections, choose node i for which $L_i^{(n)}$ is maximum (if a tie occurs, select one of the tied source nodes at random).
- 3) Among all destinations of enabled connections whose source node is i , find the destination node j for which $s_{ij}^{(n)}$ is minimum (again, if a tie occurs, select one of the tied destination nodes at random).
- 4) Find a *pivot* node k so that $s_{ik}^{(n)} > 0$ and $s_{kj}^{(n)} > 0$, through which the flow $i \Rightarrow j$ could be routed in multihop fashion; if many such nodes exist, choose the one

for which $L(S^{(n)})_k$ is minimum (once more, if a tie occurs, select one of the tied pivot nodes at random).

- 5) If no pivot node has been found, then GOTO step 7). If a pivot node has been found, compute the temporary matrix S^* as follows: first $S^* = S^{(n)}$, then $s_{ij}^* = 0$, $s_{ik}^* = s_{ik}^{(n)} + s_{ij}^{(n)}$, and $s_{kj}^* = s_{kj}^{(n)} + s_{ij}^{(n)}$.
- 6) If $L(S^*) \geq L(S^{(n)})$, then GOTO step 7). If $L(S^*) < L(S^{(n)})$, then the rerouting of connection (i, j) takes place: $S^{(n+1)} = S^*$.
- 7) Connection $i \rightarrow j$ is disabled. If any enabled connection is left, go to Step 2); otherwise $S = S^{(n)}$.

2) *Single-Hop Scheduling Algorithm*: In the second step, the single-hop scheduling algorithm is executed. The new algorithm that we propose tries to minimize the number of times a node needs to tune its transmitter in order to send all the packets to be transmitted during a scheduling period. Although this is quite often a reasonable approach for the identification of the optimal schedule, it does not guarantee that the scheduling period length is always minimized.

At the beginning of the algorithm, the first set of simultaneous transmissions in the scheduling period is selected by executing a maximum weight matching (MWM) algorithm (see [11] and [12]) on the bipartite $(N + W)$ -nodes graph that corresponds to the traffic request matrix S (N vertices of the graph correspond to packet sources, W vertices correspond to packet destinations, edges are defined by the logical topology, and weights associated with edges correspond to the entries of matrix R). After the completion of some of the transmissions in this initial set and the successive tuning delay, one or more transmitters may become available for additional packet transmissions on free wavelengths. At this point, the MWM algorithm is again executed in order to match the remaining transmission requests to the idle wavelengths. The algorithm continues until all requested transmissions have been scheduled. Since the algorithm is based on an incremental application of the MWM algorithm, we shall name it incremental MWM (IMWM).

A more detailed description of the algorithm follows.

For each source-destination pair (i, j) , define two functions: *Source* (i, j) , which returns i , and *Dest* (i, j) , which returns j . Moreover, let \mathcal{A} denote the set of assigned connections, i.e., the set of source-destination pairs (i, j) corresponding to packet flows whose transmission has already been scheduled; \mathcal{A} is initialized to the empty set \emptyset . For each source node i , define the set $\mathcal{D}(i)$ of destinations corresponding to already scheduled transmissions; $\mathcal{D}(i)$ is initialized to \emptyset for each node i . Let t_s denote the system time and initialize $t_s^{(0)} = 0$. At the algorithm start, all sources and destinations are enabled. The algorithm consists of the following steps.

- 1) Use an MWM algorithm over the requests issued by enabled source nodes for transmission toward enabled destinations, in order to find the matching configuration, i.e., the maximal set of concurrently active transmissions; for each pair (i, j) corresponding to transmissions selected by the MWM algorithm, put *Dest* (i, j) in $\mathcal{D}(\text{Source}(i, j))$, put (i, j) in \mathcal{A} , and disable the corresponding source and destination nodes.

- 2) Update the traffic request matrix S by subtracting all the entries corresponding to transmissions on connections that were just added to \mathcal{A} .
- 3) For each source node i , evaluate $t_i^{(n)}$, the minimum time at which the node can schedule a transmission toward a destination not belonging to $\mathcal{D}(i)$; let $t_s^{(n)}$ be the minimum value of $t_i^{(n)}$ greater than t_s . Update $t_s^{(n+1)} = t_s^{(n)}$. Enable the nodes for which $t_i^{(n)}$ is not greater than $t_s^{(n+1)}$.
- 4) Enable the available destinations in time slot $t_s^{(n+1)}$, i.e., the destinations that are not receiving any packet in time slot $t_s^{(n+1)}$.
- 5) If matrix S is null, the algorithm terminates. Otherwise, consider S_{en} , the submatrix of S referring to enabled source and destination nodes; if S_{en} is not null, go back to step 1). If S_{en} is null, go back to step 3).

III. COMPUTATIONAL COMPLEXITY

In this section, we discuss the algorithmic complexity of the two proposed heuristic algorithms in the case $N = W$.

Let us first examine the computational complexity of the algorithm proposed for the logical topology design.

In the initialization phase, $L(R)_i$ must be evaluated for each node; $O(N^2)$ operations are required. Each step of the algorithm basically requires searching for either the minimum or the maximum of N numbers. The complexity of each step is, as a consequence, $O(N)$. The maximum number of steps before termination is N^2 . The complexity of the whole procedure is thus $O(N^3)$.

The evaluation of the complexity of the IMWM scheduling algorithm is quite difficult, and only an extremely pessimistic (and thus scarcely interesting) upper bound can be easily derived. This is because the complexity of each step depends upon the size of the bipartite graph on which the MWM search is performed. This in turn depends on the number of elements matched in the previous steps, which is unknown in general. An evaluation of the maximum number of steps required before the algorithm termination is also quite difficult, since this too depends upon the number of matches at each step.

The derivation of a quite pessimistic upper bound is instead simple if we assume that each match selects just one traffic flow. Since the execution of the MWM algorithm on an $(N + N)$ -nodes bipartite graph requires at most $N^3 \log N$ operations, and the number of steps is at most N^2 , the computational complexity is upper bounded by $N^5 \log N$.

Similarly, in the case $W < N$, it can be obtained that the computational complexity is upper bounded by $WN^4 \log N$.

IV. NUMERICAL RESULTS

We present numerical results obtained by running the proposed single-hop and multihop scheduling algorithms with an approximate implementation of the MWM algorithm. Observe that the proposed algorithm can be trivially forced to schedule packets with a limited maximum number of hops.

TABLE I
SLOT ALLOCATION WITH SINGLE-HOP SCHEDULING

slot	1	2	3	4	5	...	13	14	15	16	17	...	26	27	28	...	37	38	39	40	
wav. 1	2	2	-	-	-	-	-	-	4	4	-	-	5	3	-	-	-	-	-	-	
wav. 2	5	5	-	-	-	-	-	-	1	1	1	-	-	4	4	-	-	3	-	-	
wav. 3	1	1	1	1	-	-	2	2	2	-	-	-	-	-	-	-	5	-	4	-	
wav. 4	3	3	3	3	3	-	5	5	5	-	-	-	-	-	1	-	-	-	2	2	
wav. 5	4	4	4	4	-	-	-	-	-	3	-	-	2	2	2	-	-	-	-	1	1

TABLE II
SLOT ALLOCATION WITH MULTIHOP SCHEDULING

slot	1	2	3	4	5	6	7	8	9	10	11	12	...	17	18	19	20	21	22	23	24	25	26	27
wav. 1	4	4	4	4	4	4	4	3	3	3	3	3	-	-	-	-	-	-	-	-	-	-	-	-
wav. 2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	4	4	4	4	4	4	1	1	1	1
wav. 3	2	2	2	2	2	2	-	-	-	-	-	-	-	-	-	-	5	5	5	5	5	-	-	-
wav. 4	5	5	5	5	5	5	5	5	5	-	-	-	-	-	-	-	-	-	-	3	3	3	3	3
wav. 5	1	1	1	1	1	1	1	1	1	1	1	1	-	2	2	2	2	2	2	-	-	-	-	-

A. Example of Multihop Scheduling

Let us begin with an example that shows the operations of the algorithms.

Consider a small all-optical WDM/TDM broadcast-and-select network with $N = W = 5$, where nodes and wavelengths are numbered from 1 to 5, each node receives packets at the wavelength labeled with the same number, the tuning time is $T = 10$ slots, and the traffic request matrix R is

$$R = \begin{pmatrix} 0 & 3 & 4 & 1 & 2 \\ 2 & 0 & 3 & 2 & 3 \\ 1 & 1 & 0 & 5 & 1 \\ 2 & 2 & 1 & 0 & 4 \\ 1 & 2 & 1 & 3 & 0 \end{pmatrix}.$$

The requested packet transmissions add up to 44 packets: 3 packets must be transmitted from node 1 to node 2, 4 from node 1 to node 3, and so on.

The scheduling period resulting from the execution of the single-hop scheduling algorithm comprises 50 slots (taking into account 10 slots devoted to the tuning time of the last active source before the next scheduling period), and is partially represented by Table I, where rows correspond to wavelengths, columns correspond to time slots, and numbers in the table represent nodes' indices.

Node 2 transmits at wavelength 1 in the first two slots of the scheduling period, then it tunes to wavelength 3, and within ten slots transmits three packets; then it tunes to wavelength 5 to transmit three more packets, and finally it tunes to wavelength 4 for the transmission of two packets. Ten more slots are then necessary to tune again to wavelength 1 and restart the scheduling period.

By running the logical topology design algorithm to construct a two-hop logical topology (a multihop topology with a maximum number of intermediate nodes equal to 1), we obtain the following routing matrix, where each element x_{ij} represents the

intermediate node in the route from source node i to destination node j :

$$X = \begin{pmatrix} - & 2 & \mathbf{5} & \mathbf{5} & 5 \\ \mathbf{3} & - & 3 & \mathbf{5} & 5 \\ 1 & 1 & - & 4 & \mathbf{1} \\ 1 & 2 & \mathbf{2} & - & \mathbf{1} \\ \mathbf{4} & \mathbf{4} & 3 & 4 & - \end{pmatrix}.$$

From matrix X , we observe that the flow from node 1 to node 3 is routed through node 5, while the flow from node 1 to node 2 is transmitted directly from source to destination, and so on. Entries corresponding to two-hop flows are emphasized in matrix X .

The routing matrix X originates the traffic request matrix S

$$S = \begin{pmatrix} 0 & 4 & 0 & 0 & 12 \\ 0 & 0 & 6 & 0 & 5 \\ 5 & 0 & 0 & 5 & 0 \\ 7 & 5 & 0 & 0 & 0 \\ 0 & 0 & 5 & 9 & 0 \end{pmatrix}.$$

Matrix S results from the two-hop routing of a number of packet flows that were previously transmitted in single-hop fashion. For example, the packet to be transmitted from 3 to 2 now goes first from 3 to 1 and then from 1 to 2; this adds 1 to r_{31} and to r_{12} , and sets $r_{32} = 0$. The number of packets to be transmitted in single-hop fashion is now equal to 63: 19 packets are transmitted in two-hop mode, and their transmissions must be added to the original 44 packets.

Despite the larger number of packets that must be accommodated in one frame, the scheduling period resulting from the execution of the two-hop scheduling algorithm only comprises 37 slots and is partially represented by Table II.

It can be observed that node 2 transmits at wavelength 3 in the first six slots of the scheduling period, then it tunes to wavelength 5 and transmits five packets. The packets that it previously had to transmit to nodes 1 and 4 are now transmitted in two-hop fashion through nodes 3 and 5. Additionally, node 2 also has to take care of the transmission of one more packet that it relays from source node 4 to destination node 3.

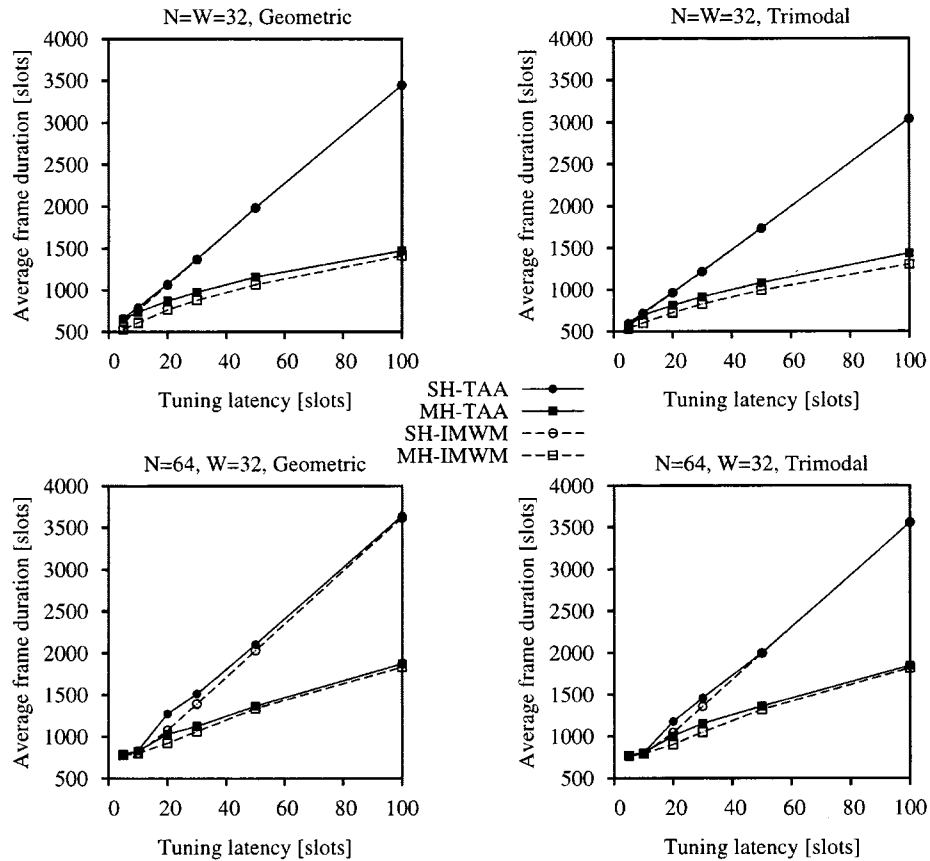


Fig. 1. Average scheduling period duration as a function of the number of slots necessary to tune node transmitters for the single-hop and multihop TAA and single-hop and multihop IMWM algorithms.

B. Scheduling Period Duration

Let us now observe the average performance of the single-hop and multihop scheduling algorithms for larger network configurations. We always assume that the number of wavelengths is $W = 32$, and we take the number of nodes to be either $N = 32$ or $N = 64$. One hundred different traffic matrices R are randomly generated, always setting to zero the elements along the main diagonal (nodes do not transmit to themselves), and fixing the mean of the distributions from which the other elements are drawn to 10 when $N = 32$, and to 5 when $N = 64$ (so that destinations are uniformly selected and the average load on each wavelength channel is constant). The elements of matrix R correspond to individual requests from source nodes and are taken to be either geometrically distributed or uniformly chosen among three values. The three values are 0, 3, and 27 for $N = 32$, and 0, 3, and 12 for $N = 64$. For $W = N = 32$, the distributions of the number of packets transmitted by a node on a given wavelength coincide with the distributions of the elements of matrix R . With $N = 64$, instead, the number of packets that a node has to transmit on a given wavelength corresponds to the superposition of the requests leading to two destination nodes, and is thus either distributed according to the discrete equivalent of an Erlang-2, or chosen among the six values 0, 3, 6, 12, 15, 24, with probability $2/9$ for the values 3, 12, and 15, and $1/9$ for all others.

The performances of the different scheduling algorithms are plotted in Fig. 1 as curves of the mean scheduling period du-

ration (averaged over the 100 randomly generated matrices R) versus the number T of slots necessary to tune the transmitters. In regard to the single-hop scheduling algorithm, we consider both IMWM, newly proposed in this paper, and Tuning Assignment Algorithm (TAA), originally proposed in [9]. In the case of multihop scheduling, we always consider our heuristic approach for the design of the logical topology, but again we consider either IMWM or TAA for the second step of the procedure.

The curves in Fig. 1 show that the gain obtained with the multihop scheduling algorithm can be quite remarkable, especially with long tuning times, both when $N = W$ and $N > W$. Instead, the differences between the two considered single-hop scheduling algorithms do not appear to be significant, even if they slightly favor IMWM over TAA. Marginal differences can be observed between the two considered traffic distributions.

The curves in Fig. 2 report curves of the standard deviation of the scheduling period duration versus T for $N = W = 32$. The standard deviation values are remarkably smaller than the corresponding averages, so that it can be claimed that the scheduling period average lengths are a good indication of the scheduling algorithm performance. Standard deviation values were observed to be similarly small also for the results presented later in this section; thus, for the sake of conciseness, we will not show more standard deviation curves in the remainder of the paper.

The multihop scheduling algorithm can be restricted to a maximum number of hops to limit packet delay. In Fig. 3, we

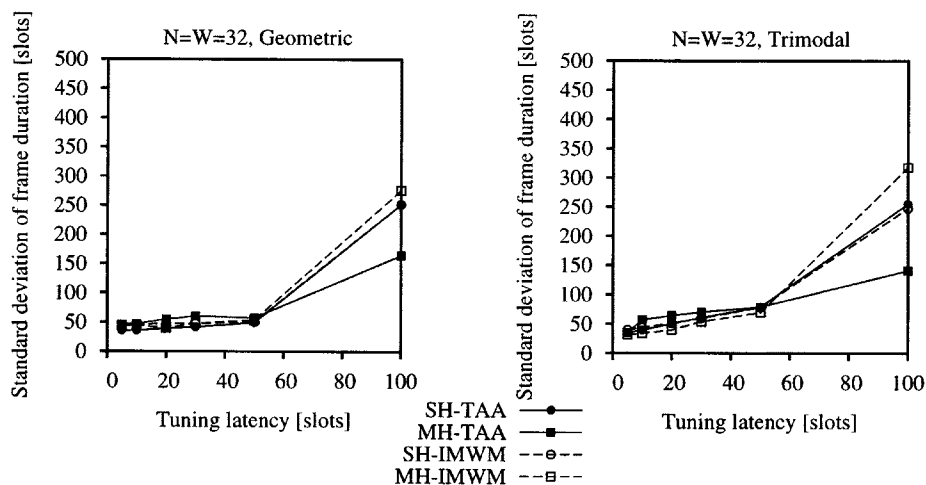


Fig. 2. Standard deviation of the scheduling period duration as a function of the number of slots necessary to tune node transmitters for the single-hop and multihop TAA and single-hop and multihop IMWM algorithms.

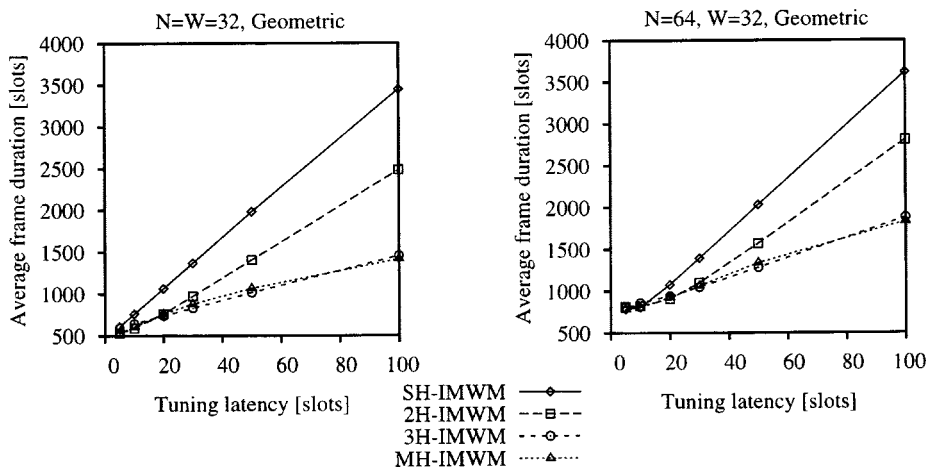


Fig. 3. Average scheduling period duration as a function of the number of slots necessary to tune node transmitters for the single-hop, 2-hop, 3-hop, and multihop IMWM algorithms.

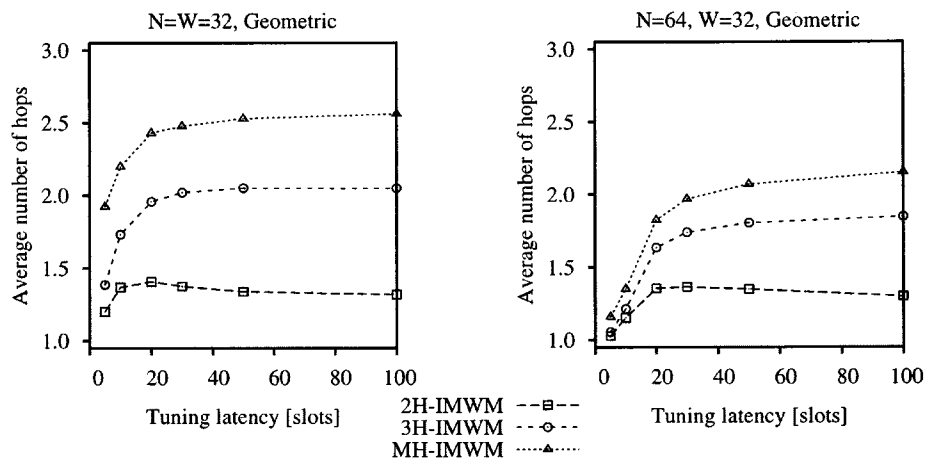


Fig. 4. Average number of hops as a function of the number of slots necessary to tune node transmitters for the 2-hop, 3-hop, and multihop IMWM algorithms.

show curves of the average scheduling period duration versus the transmitter tuning time in slots, for the case of a maximum number of hops equal to 1 (single-hop), 2, 3, or unlimited (multihop). We can observe that improvements are significant from 1 to 2 hops and from 2 to 3 hops, but there is no evident

gain when no restriction on the maximum number of hops is imposed. Actually, the multihop schedule for some tuning delays is even longer than the 3-hop schedule. This is due to the fact that when the number of hops grows, the resulting traffic request matrix S is more difficult to schedule in single-hop

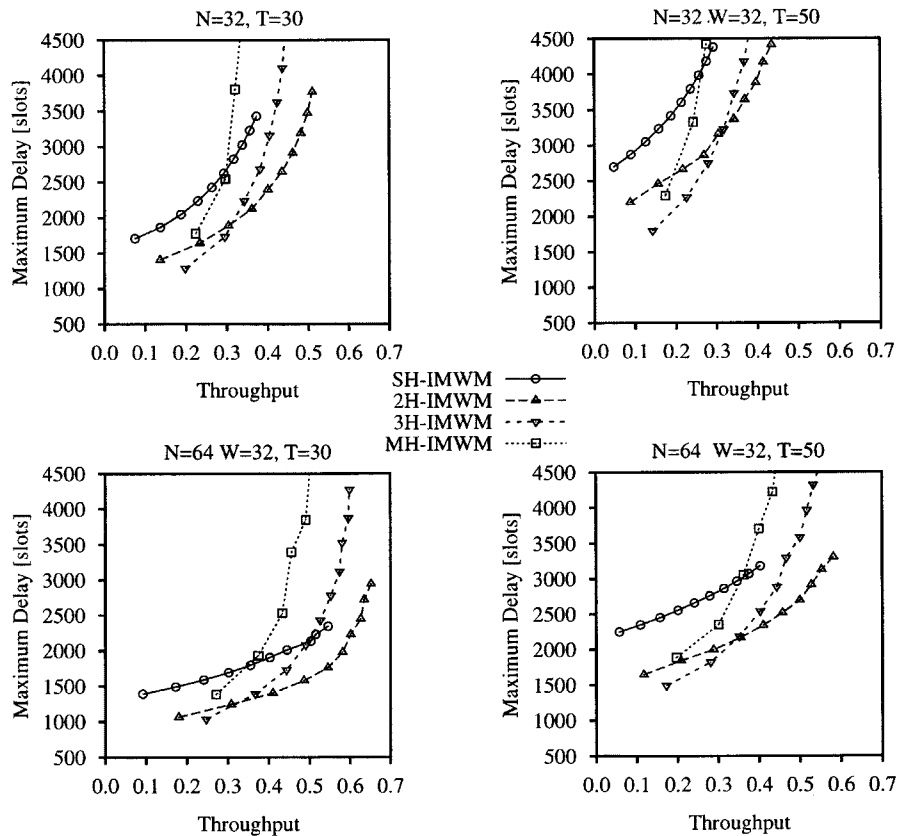


Fig. 5. Maximum delay versus throughput for the single-hop, 2-hop, 3-hop, and multihop IMWM algorithms.

mode. In Fig. 4, we report the average number of hops in the same scenario. While only a small fraction of traffic follows a multihop path if the maximum number of hops is limited to 2, more than half traffic flows become multihop in 3-hop schedules, and most traffic flows follow a multihop path if no constraint is imposed on the number of hops. When the number of wavelengths is equal to half the number of nodes, the amount of traffic following a multihop path is reduced.

C. Packet Delays

The fact that multihop schedules produce shorter scheduling periods with respect to single-hop schedules guarantees that a higher throughput can be obtained. However, it is not clear whether these throughput gains also lead to a reduction of the packet delay, because of the fact that multihop packets must be transmitted several times in successive scheduling periods. Moreover, the fact that gains are large only for long tuning times might suggest, as an alternate approach for the improvement of the system efficiency, the scheduling of a larger quantity of packets (for example scheduling in one scheduling period all the packets corresponding to k original scheduling periods). While this surely has a negative impact on delay, it also leads to throughput improvements for the single-hop scheduling due to the reduced number of tuning actions.

In order to investigate these effects, we plot in Fig. 5 the curves of the maximum packet delay, and in Fig. 6 the curves of the average packet delay, versus throughput, for the cases $T = 30$, and $T = 50$. The four curves again refer to the single-hop,

2-hop, 3-hop, and multihop schedules. The first point in each curve refers to the average over 30 traffic request matrices; subsequent points correspond to the averages computed over the same 30 matrices multiplied by a scale factor ($k = 2, 3, 4, \dots$), so as to schedule a larger amount of packets in the same scheduling period. The maximum delay is computed as the scheduling period duration times the maximum number of hops incremented by one (this is because in the worst case, a transmission in h hops involves a packet generated at the beginning of frame i and transmitted at the end of frame $i + h$). The average delay is computed as the scheduling period duration times the average number of hops experienced by packets.

We can observe that in the considered cases, both the single-hop maximum and average packet delays can be greatly reduced with a 2-hop approach over the entire throughput range. The same is not true for schedules with a larger numbers of hops that produce quite large gains for low throughput values, but become ineffective at higher throughputs. This means that the gains of the multihop scheduling algorithm are maintained, independently of the quantity of traffic to be scheduled, provided that the number of hops is kept low. Multihop schedules, if the number of hops is kept low, lead to significant improvements of the network bandwidth exploitation under urgent strict delay requirements. This result can be explained by observing that the growth in the maximum packet delay is normally proportional to the maximum allowed number of hops, so that imposing a constraint can be quite beneficial. The average packet delay instead depends on the average number of hops, which remains rather close to 1 with 2-hop schedules,

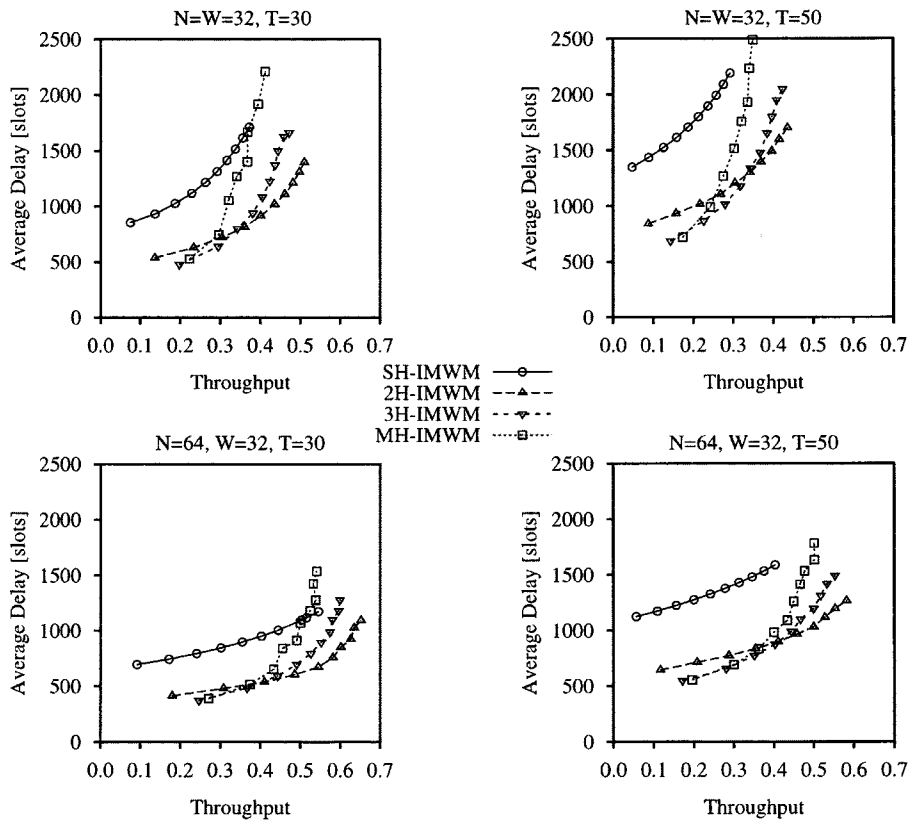


Fig. 6. Average delay versus throughput for the single-hop, 2-hop, 3-hop, and multihop IMWM algorithms.

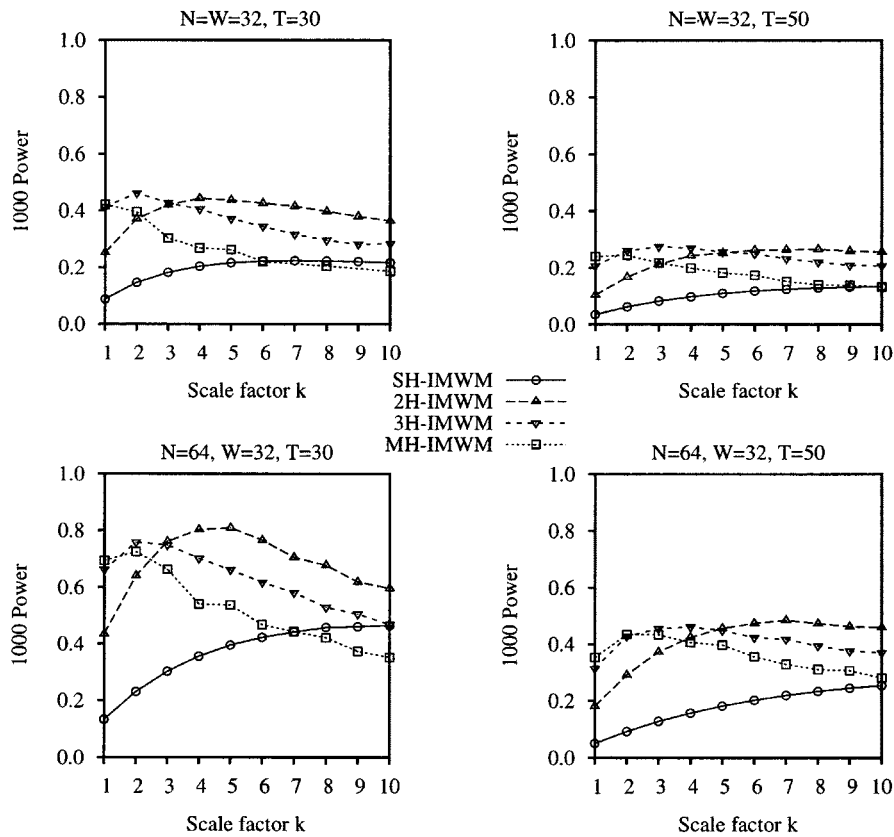


Fig. 7. Power \times 1000 as a function of the scale factor k for the single-hop, 2-hop, 3-hop, and multihop IMWM algorithms.

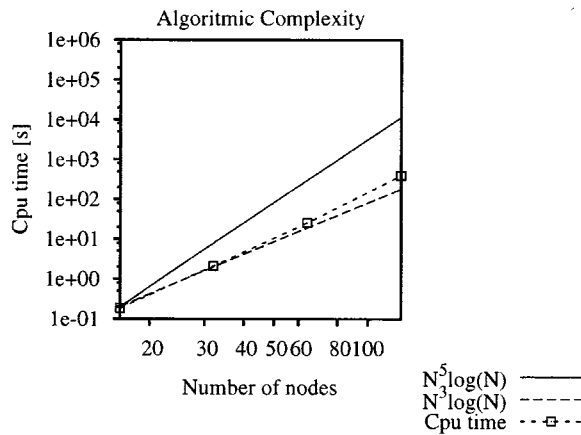


Fig. 8. CPU time for different problem sizes.

when only the critical packet flows follow 2-hop routes; this leads to advantages in throughput with minor losses in delay.

In order to better visualize these results, we plot in Fig. 7 the curves of the power (throughput over average packet delay) times 1000 versus the requested traffic, for the cases $T = 30$, and $T = 50$. The four curves again refer to the single-hop, 2-hop, 3-hop, and multihop IMWM algorithms. It can be clearly seen that limiting the number of hops to 2 or 3 provides the best performance; unbounded multihop algorithms provide performance only slightly better or comparable to those of single-hop schedules.

Finally, in Fig. 8 we plot the maximum CPU times required to run the IMWM algorithms for different values of N for the case $W = N = 32$, with 30 random traffic request matrices. We report for comparison the curves $\alpha N^5 \log N$ and $\beta N^3 \log N$, where α and β are chosen in order to match the CPU time for $N = 16$; the first curve is related to the pessimistic upper bound computed in Section III, while the second curve refers to the complexity of the MWM algorithm. It can be seen that the complexity of the logical topology design plus the single-hop scheduling IMWM, measured in terms of CPU times, increases with the number of nodes much less than the pessimistic upper bound computed in this paper, and similarly to the complexity of the MWM algorithm alone.

V. CONCLUSIONS

In this paper, we considered all-optical WDM/TDM broadcast-and-select networks with N nodes and W slotted and synchronized WDM transmission channels. Network nodes have one fixed receiver and one tunable transmitter with nonnegligible tuning times.

For these systems, we explored multihop scheduling of WDM/TDM packet transmissions. We first formally defined the multihop scheduling problem as an ILP problem. To solve the problem with acceptable complexity, we devised a heuristic approach which runs through two steps, called the logical topology design and the single-hop scheduling. An algorithm which provides a suboptimal solution to the logical topology design problem was proposed. We further introduced an algorithm for the single-hop scheduling problem which provides minor improvements over previously proposed algorithms.

Our results show that the adoption of multihop scheduling algorithms provides important throughput advantages over single-hop scheduling. We also showed that limiting the maximum number of intermediate nodes to 1 or 2 leads to a good compromise in terms of throughput versus delay, with respect to single-hop and unbounded multihop scheduling algorithms.

REFERENCES

- [1] A. Sneh and K. M. Johnson, "High-speed tunable liquid crystal optical filter for WDM systems," in *Proc. IEEE/LEOS'94, Summer Topical Mtg. on Optical Networks and their Enabling Technologies*, Lake Tahoe, NV, July 1994.
- [2] D. A. Smith *et al.*, "Evolution of acousto-optic wavelength routing switch," *J. Lightwave Technol.*, vol. 14, pp. 1005–1119, June 1996.
- [3] B. Mukherjee, "WDM based local lightwave networks—Part I: Single-hop systems," *IEEE Network Mag.*, vol. 6, pp. 12–27, May 1992.
- [4] J. C. Lu and L. Kleinrock, "A wavelength division multiple access protocol for high-speed local area networks with passive star topology," *Perform. Eval. J.*, vol. 16, pp. 223–239, Nov. 1992.
- [5] D. Guo, Y. Yemini, and Z. Zhang, "Scalable high speed protocols for WDM star networks," in *Proc. IEEE INFOCOM'94*, Toronto, ON, Canada, June 1994.
- [6] F. Jia, B. Mukherjee, J. Iness, and S. Ojha, "Variable length message scheduling algorithms for a WDM based local lightwave network," in *Proc. IEEE INFOCOM'94*, Toronto, ON, Canada, June 1994.
- [7] I. S. Gopal and C. K. Wong, "Minimizing the number of switchings in a SS/TDMA system," *IEEE Trans. Commun.*, vol. COM-33, pp. 497–501, June 1985.
- [8] A. Ganz and Y. Gao, "A time-wavelength assignment algorithm for a WDM star network," in *Proc. IEEE INFOCOM'92*, Florence, Italy, May 1992.
- [9] M. S. Borella and B. Mukherjee, "Efficient scheduling of nonuniform packet traffic in a WDM/TDM local lightwave network with arbitrary transceiver tuning latencies," *IEEE J. Select. Areas Commun.*, vol. 14, pp. 923–934, June 1996.
- [10] R. Guerin, H. Ahmadi, and M. Naghshineh, "Equivalent capacity and its application to bandwidth allocation in high-speed networks," *IEEE J. Select. Areas Commun.*, vol. 9, pp. 968–981, 1991.
- [11] N. McKeown, V. Anantharam, and J. Walrand, "Achieving 100% throughput in an input-queued switch," in *Proc. IEEE INFOCOM'96*, San Francisco, CA, Mar. 1996.
- [12] R. E. Tarjan, *Data Structures and Network Algorithms*. Philadelphia, PA: Society for Industrial and Applied Mathematics, Nov. 1983.



Marco Ajmone Marsan (S'76–M'78–SM'86–F'99) was born in Turin, Italy, in 1951. He received the Dr. Ing. degree from Politecnico di Torino, Turin, Italy, and the M.S. degree from the University of California at Los Angeles, both in electrical engineering.

From November 1975 to October 1987, he was at the Electronics Department of Politecnico di Torino, first as a Researcher, then as an Associate Professor. From November 1987 to October 1990, he was a Full Professor at the Computer Science Department of the University of Milan, Milan, Italy. During the summers of 1980 and 1981, he was with the Research in Distributed Processing Group, Computer Science Department, UCLA. During the summer of 1998, he was an Erskine Fellow at the Computer Science Department of the University of Canterbury in New Zealand. He has co-authored over 200 journal and conference papers in the areas of Communications and Computer Science, as well as the two books *Performance Models of Multiprocessor Systems* (Cambridge, MA: MIT Press, 1986) and *Modeling with Generalized Stochastic Petri Nets* (New York: Wiley, 1985). His current interests are in the fields of performance evaluation of communication networks and their protocols.

Prof. Ajmone Marsan received the Best Paper Award at the Third International Conference on Distributed Computing Systems in 1982.



Andrea Bianco (M'99) was born in Turin, Italy, in 1962. He received the Dr.Ing. degree in electronics engineering in 1986 and the Ph.D. degree in telecommunications engineering in 1993, both from Politecnico di Torino, Turin, Italy.

Since 1994, he has been an Assistant Professor at the Electronics Department at the Politecnico di Torino, first in the Dipartimento di Sistemi di Produzione, later in the Dipartimento di Elettronica. In 1993, he visited Hewlett-Packard Labs in Palo Alto, CA. His current research interests are in

the fields of protocols for all-optical networks and switch architectures for high-speed networks.



Emilio Leonardi (M'00) was born in Cosenza, Italy, in 1967. He received the Dr.Ing. degree in electronics engineering in 1991 and the Ph.D. degree in telecommunications engineering in 1995, both from the Politecnico di Torino, Turin, Italy.

In 1995, he spent one year at the Computer Science Department of the University of California, Los Angeles, where he was involved in the Supercomputer-SuperNet (SSN) project aimed at the design of a high-capacity optical network architecture. During the summer of 1999, he joined the High-Speed Research

Group of Lucent Technologies, in Holmdel, NJ, where he worked on the design of scheduling algorithms for high-capacity switch architectures. He is currently an Assistant Professor at the Electronics Department of Politecnico di Torino. His research interests are in the fields of all-optical networks, switching architectures, and queuing theory.



Fabio Neri (M'99) was born in Novara, Italy, in 1958. He received the Dr.Ing. and Ph.D. degrees in electrical engineering from Politecnico di Torino, Turin, Italy, in 1981 and 1987, respectively.

He is currently an Associate Professor at the Electronics Department of Politecnico di Torino, Turin, Italy. From 1991 to 1992, he was with the Information Engineering Department at University of Parma, Parma, Italy, as an Associate Professor. From 1982 to 1983, he was a Visiting Scholar at George Washington University in Washington, DC. In the summer

of 1995, he was Visiting Researcher at the Computer Science Department of the University of California in Los Angeles. In the summer of 1998, he visited Bell Laboratories/Lucent Technologies in Holmdel, NJ. He has coauthored over 50 papers published in international journals and presented in leading international conferences. His research interests are in the fields of performance evaluation of communication networks, high-speed and all-optical networks, discrete event simulation, and queuing theory.



Antonio Nucci (S'00) was born in Lecce, Italy, in 1974. He received the Dr.Ing. degree in electronics engineering from Politecnico di Torino in July 1998. He is currently working toward the Ph.D. degree in telecommunications engineering at the Politecnico di Torino, Turin, Italy. His research interests include all-optical networks and optimization methods.