Incremental Multi-Hop Scheduling Algorithms for All-Optical Broadcast-and-Select Networks with Arbitrary Tuning Latencies

A. Bianco, M. Guido, E. Leonardi
Dipartimento di Elettronica, Politecnico di Torino, Italy
e-mail: {bianco,leonardi}@polito.it

Abstract—We focus on all-optical broadcast, and select slotted WDM networks. Each network user is equipped with one tunable transmitter and one fixed receiver; full connectivity is achieved by tuning transmitters to all different wavelengths available in the optical spectrum. Tuning latencies are considered to be non negligible with respect to the slot time. A centralized network controller allocates slots in a TDM/WDM frame according to requests issued by users. User requests are accommodated in the frame incrementally, as soon as they are received by the network controller.

We propose a novel scheduling algorithm that may route some flows from source to destination through some intermediate nodes, following a multi-hop approach. Since we aim at an incremental solution, we impose a transparency constraint: new user requests may be accepted only without affecting existing allocations, otherwise they are refused.

A heuristic quasi-optimal scheduling algorithm is proposed. Performance results show that significant benefits can be achieved with respect to traditional single-hop approaches.

I. INTRODUCTION

We consider all-optical WDM/TDM broadcast-and-select networks based on a general topology (possibly a star), in which W wavelengths are available for user-to-user packet communications, and we assume that user interfaces are equipped with one full-duplex transceiver; hence, each user can be source and destination of at most one data flow at any given time. By tuning the transmitter, and by dynamically allocating the W available wavelengths, full connectivity is achieved among end-users. The time required to tune transceivers is assumed to be not negligible with respect to the packet transmission time. WDM channels are assumed to be slotted and synchronized; each slot can accommodate on one wavelength the transmission of one packet.

Different approaches were proposed to solve the access problem in all-optical broadcast-and-select networks. Some are based on random access protocols (see [1] and references therein), others are based on a deterministic resolution of possible conflicts via fast slot allocation [2], [3], [4], others compute static WDM/TDM schedules, which allocate transmission resources according to long term user bandwidth needs ([5], [6], [7]). We consider the latter algorithms in a context of slowly-varying traffic patterns, in which scheduling algorithms are executed once in a while (at a much lower rate with respect to individual packet transmissions) to recompute the WDM/TDM schedule in response to changed user requirements; thus, these scheduling protocols react to slowly-varying user bandwidth request. We shall assume to allocate resources on the basis of a traffic request matrix, which describes the users' traffic requirements according to 'long term' bandwidth needs.

Two approaches exist to determine the transmission schedule, i.e., slot assignment in the TDM/WDM frame, named non-incremental and incremental respectively in this paper. Most previous studies [5], [6], [7], [8], [9], [10], [11] refer to the non-incremental (or off-line) approach: algorithms to schedule transmissions are designed assuming an exact knowledge of a stable traffic pattern via a request matrix describing user's transmission requests. In this case the scheduling algorithm is executed by a centralized entity, a network controller, in a non-incremental fashion with respect to packet transmissions and issuing of bandwidth requests, since all the requests are considered and scheduled at the same time. In this context the new scheduling is usually obtained by re-allocating network resources for all user-to-user traffic flows. In a dynamic traffic scenario this means that any change in bandwidth allocation results in the rescheduling of all transmission.

Instead of focusing on non-incremental algorithms, we study incremental (or on-line) scheduling [12], [13], [14]: the traffic pattern is supposed to be time-varying, and each time a variation in some traffic flows is required, the algorithm is executed at the controller to adapt the slot assignments to traffic fluctuations. This scenario naturally leads to scheduling where existing traffic flows are not reallocated. Indeed, to reduce the complexity of the scheduling algorithm and the bandwidth devoted to signalling, we impose to our incremental algorithms a transparency property: a variation in the allocation of slots for a traffic flow does not modify the scheduling of other already established connections.

In this paper, a heuristic algorithm for the identification of a good transparent incremental schedule is proposed. The algorithm is based on the possibility of routing some flows from the source to the destination throughout some intermediate nodes (i.e. in a multi-hop fashion). We show that multi-hop approaches lead to significant performance improvements with respect to single-hop one, especially when tuning latencies are non-negligible.

II. PROBLEM STATEMENT AND RELATED WORK

The problem of finding an optimum transmission schedule for a given traffic pattern in presence of non-negligible tuning latencies was already extensively analyzed [5], [6], [7],
It is usually formulated, for the non-incremental case, in the following way:

Given a slot allocation request matrix $R$, whose elements $R_{sd}$ are the numbers of packets that must be transmitted from any source $s$ to any destination $d$ in a frame, find a time/wavelength assignment that guarantees the delivery of the requested traffic, while minimizing the time necessary to accommodate all transmissions (i.e. maximizing the network throughput), subject to tuning latencies constraints.

We refer to this formulation as a variable-frame length problem. This formalization has the advantage of being well-suited for complexity evaluation, since it can be easily made equivalent to an integer linear programming problem. From the application viewpoint, instead, varying the size of the frame duration means varying the user throughput: normally the user specifies its bandwidth requirements (in bits per second), while the entries $R_{sd}$ of the traffic request matrix $R$ specify packets per frame. If the frame duration is changed without a modification of the number of slots per frame allocated to the users, also the bandwidth allocated to users changes. The variable-frame problem formalization is therefore suited only for one-shot transmission of the number of packets specified by $R$ (but this is not of interest in the context of this paper), or when the entries stored in matrix $R$ must be interpreted as relative weights among the different source/destination pairs, and the aim of the scheduling algorithm is to maximize the system throughput while preserving relative weights among connections. Moreover, changing the frame duration in an incremental scenario implies additional complexity and control bandwidth, since the controller must notify all users of the current frame duration after each schedule adjustment, and users must be able to continuously adapt to variations in the frame duration.

We therefore formulate the non-incremental scheduling problem using a fixed-frame length as:

Given a sequence of slot allocation/deallocation request matrices $R(n)$, whose elements $R_{sd}(n)$ specify the numbers of new slots that must be allocated/deallocated from any source $s$ to any destination $d$ given a fixed frame length equal to $F$ slots, find a time/wavelength assignment, that allocate/deallocate in sequence the request matrices satisfying the tuning latencies and transparency constraint (i.e. a time/wavelength assignment that each time avoids reallocation of the already existing allocations) and minimizes the number of requests that can not be accommodated.

In [11] the variable-frame length scheduling problem was shown to be NP-hard. Several heuristic approaches for the determination of good non-incremental schedules were proposed in [5], [6], [7], [10]. All these approaches, being formulated for the non-incremental case, assume a perfectly known and stable traffic pattern. This may be not the case for most application contexts: often the traffic is neither stable nor known a priori. Even if we assume that the traffic can be seen as an aggregation of individual information flows, the aggregation exhibits bandwidth requirements that can be (slowly) changing with time. We assume therefore in this paper that the network controller reacts to variations in the bandwidth requirements of traffic flows by allocating (or deallocating) in the WDM/TDM frame the appropriate number of slots. To reduce the algorithmic complexity and the signalling bandwidth, the operation of slot allocation (and deallocation) for some flows should not have any impact on the scheduling of those traffic flows whose parameters have not been changed. As already mentioned, we call transparency this property of the scheduling algorithm.

III. INCREMENTAL SCHEDULING ALGORITHMS

We consider a network with $N$ users and $W$ wavelengths (for simplicity, $N$ is assumed to be an integer multiple of $W$). In the following, user interfaces are supposed to be equipped with one tunable transmitter and one fixed receiver. All the strategies presented in this paper can however be easily adapted to the dual case in which transmitters are fixed and receivers are tunable.

In the case of tunable transmitters and fixed receivers, the destination address of the traffic flow immediately identifies the wavelength on which data must be transmitted. If $W < N$, more than one receiver share the same wavelength.

Each time the bandwidth requested by a traffic flow changes, the controller reacts by varying the scheduling of the flow, without modifying the slot assignment of other flows.

Slots can be identified with a pair $(t, w)$, representing respectively the slot position inside the frame and the wavelength. In general, slot $(t, w)$ can be assigned by the controller to a traffic flow $(s, d)$ from source $s$ to destination $d$ only if the following two conditions are satisfied:

- slot $(t, w)$ is free, i.e., no other user is transmitting on wavelength $w$ at the same time;
- user $s$ is free, i.e., it is neither transmitting on some other wavelength, nor tuning its transmitter.

If both previous conditions are satisfied for slot $t$ on wavelength $w$, we say that slot $(t, w)$ is an $(s, d)$-eligible slot.

While it is possible to formalize the non-incremental line scheduling problem in terms of Integer Linear Programming [6], [7], [13], and, as a consequence, formally define the optimal scheduling policy ([7], [15]) for a given request matrix $R$, any definition of an incremental globally optimal policy (i.e. the policy that minimizes the long term average call blocking probability) is impossible in a real environment, since the future sequence of requests is completely unknown.

Thus a relaxation of the optimality criterion is required. We restrict our investigation to the class of greedy scheduling policies; i.e. those policies for which a new slot request $R_{sd}$ is never rejected if it can be allocated.

A suitable index for the congestion level of the network is represented by the total number of eligible slots $E$, (i.e., $E = \sum_{s=1}^{N} \sum_{d=1}^{N} c_{sd}$, where $c_{sd}$ represents the total number of $(s, d)$-eligible slots. Thus, a reasonable criteria to choose the best allocation of a new request $R_{sd}$ may be a choice that
maximizes the total number of eligible slots $E$ after the allocation. More precisely, we say that an allocation policy is locally optimal if it allocates each new set of requests $R_{sd}$ in such a way that the total number of eligible slots $E$ after allocation is maximized.

Note that when the tuning latency is non-negligible, the class of locally optimal policies may not include any single-hop scheduling policy. Indeed, while multi-hop forwarding of packets lead to an inefficient use of the network bandwidth, since the same packet must be transmitted several times on different time slots, it allows to reduce the number of times that sources are required to tune inside the frame, thus, possibly leading to a more efficient use of transmitters at source nodes.

Unfortunately, multi-hop packet forwarding entails an increase of the delivery delay, since some extra propagation delay and some delay at each intermediate node are required. The store-and-forward delay at intermediate node, (i.e. the time elapsing between the reception and the transmission of each packet) is upper-bounded by the frame length. As a consequence, the delivery delay for packets transmitted on a $k$ hops route is upper bounded by:

$$D_k \leq k(F + d_p)$$

where $d_p$ is the maximum user-to-user propagation delay measured in slot in the network and $F$ is the frame length. To limit the delay penalty, in our heuristic policy we will restrict multi-hop routes to two hops only. We will name our policy two-hop policy in the remainder of the paper.

An example of the potential advantages provided by a two-hop scheduling algorithm is presented, focusing on a network with $N = W = 4$; let us assume that is issued a request of transmitting 1 slot per frame from source node 1 to destination node 2, i.e. $R_{12} = 1$, and that tuning latency is equal to 2 slots. The frame allocation pattern prior to the new request is reported in Tab. I. To maximize the total number of eligible slots $E$, the new request must be allocated on the two-hop route, $1 \rightarrow 3$ and $3 \rightarrow 2$, on slots 5 and 7 respectively. Indeed, the best possible single hop allocation (on slot 7) would cost at least 11 slots in terms of reduction of $E$, as shown in Tab.II, whereas the considered two-hop allocation costs only 9 slots. This is due to the fact that, by allocating on a two-hop route the new request $R_{12}$, no extra tunings are required since connections $1 \rightarrow 3$ and $3 \rightarrow 2$ are already up, while the establishment of a direct connection between stations 1 and 2 would have required an extra tuning at source 1.

Since the algorithmic complexity required to implement a locally optimal incremental scheduling algorithm is very high when the number of nodes, wavelengths, and slots inside the frame exceed some tens, we propose in this paper a simple heuristic scheduling algorithm.

A. Quasi Optimal Incremental Multi-hop Scheduling Algorithm: Two-Hop Policy

In this section, we propose an incremental multi-hop scheduling algorithm that tries to achieve a good trade-off between optimality and computational complexity. The proposed scheduling algorithm limits the possible routes from source $s$ to destination $d$ to single-hop and two-hops paths only, to control the delay penalty introduced by multi-hop forwarding. New requests arrived at the same time at the controller are served in sequence, according to a FIFO discipline.

Suppose that a new request $R_{sd}$ from source $s$ toward destination $d$ has to be served at the controller.

In the first step of the algorithm, if $s$ is already transmitting some packets to $d$, we check whether the new request can be accommodated on slots that are contiguous with those already used by $s$; this would not require any extra tuning at the source. If the test is successful, the allocation of the new request is performed and the algorithm successfully ends.

Otherwise, the amount of requested slots $R_{sd}$ is compared with the tuning latency:

- if $R_{sd} > T$, a two-hop request allocation is first attempted. Then, in case of allocation failure, a single-hop allocation is attempted. In case of further failure the algorithm ends.
- if $R_{sd} \leq T$ a single-hop allocation of the request is first attempted. Then, in case of allocation failure, a two hops allocation is attempted. In case of further failure the algorithm ends.

The single-hop allocation is performed according to the following algorithm: the longest sequence of consecutive $(s, d)$-eligible slots on the destination channel is selected; let $M$ be the length of such sequence.

- if $M \geq R_{sd}$, the allocation is performed on the $R_{sd}$ rightmost slots.
- if $M < R_{sd}$, the allocation procedure fails.

The two-hop slot allocation is attempted according to the following algorithm: all the nodes that are transmitting some packets to destination $d$ and to which source $s$ is transmitting are selected:

- for each selected node $t$, the number of residual available slots $n_{s \rightarrow t}$ and $n_{t \rightarrow d}$ on logical links $i \rightarrow t$ and $t \rightarrow j$ (i.e. eligible slots that can be used for the transport of packets associated to the new request without requiring any extra tuning at the transmitter) is computed;
- nodes $t$ for which $n_{s \rightarrow t} \geq R_{sd}$ and $n_{t \rightarrow d} \geq R_{sd}$ are eligible as intermediate node;
- among the intermediate eligible nodes $t$, if any, the least loaded node (i.e. the one transmitting the minimum number of packets) is chosen, and the request is allocated; otherwise the allocation fails.

IV. Performance Results

In this section, we present a simulation based performance comparison between the two-hop strategy described in the previous section and other scheduling strategies. All numerical results were obtained by stopping simulation runs when
a 5% confidence interval width was reached with 95% confidence level.

We compare our two-hop strategy with two types of scheduling algorithms: the first type, referred as BFS, is a straightforward extension of a single-hop scheduling algorithm already presented in the literature [12], [14]; the second type refers to a novel family of multi-hop scheduling algorithms, where the multi-hop approach is based on a regular logical topology defined a-priori without any knowledge of the traffic pattern. Both algorithms are briefly described below.

In the BFS (Best Fit Search) single-hop policy, when a slot allocation request $R_{sd}$ is issued, first, the network controller checks whether station $s$ is already transmitting on the channel on which destination $d$ is receiving; if the new request can be allocated on slots contiguous with slots already used to transmit packets from $s$ to $d$, thus without requiring any extra tuning, the allocation is successfully performed. Otherwise, a sequential search for all sequences of at least $R_{sd}$ contiguous $(s, d)$-eligible slots is performed. If at least one sequence is found, the minimum-size sequence (ties are randomly broken) is chosen; the first $R_{sd}$ slots in the minimum-size sequence are devoted to the new transmission from $s$ to $d$. If not enough contiguous slots are found, a sequential search is repeated, and any sequence of $(s, d)$-eligible slots is considered, regardless of its length: successive partial allocations are attempted in all the sequences. The request $R_{sd}$ is accepted only if all the $R_{sd}$ slots can be allocated although not contiguously; otherwise, the request is refused. In other words, partial allocation of a request is not considered.

The multi-hop scheduling algorithm, used as a comparison, is based on the definition of a static (i.e. not dynamically adapted to the traffic pattern and load) regular logical topology over the physical topology; packets are routed from

![Table 1](image1.png)

**TABLE I**

Allocation pattern on a frame of 12 slots prior to the new request $R_{12} = 1$ slot.

<table>
<thead>
<tr>
<th>Slots</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source 1</td>
<td>3</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>Source 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Source 3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Source 4</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

![Table 2](image2.png)

**TABLE II**

8 $(i, j)$-eligible slot due to single-hop (SH) and two-hop (2H) allocations.

<table>
<thead>
<tr>
<th>SH</th>
<th>(1, 2)</th>
<th>(1, 3)</th>
<th>(1, 4)</th>
<th>(2, 1)</th>
<th>(2, 3)</th>
<th>(2, 4)</th>
<th>(3, 1)</th>
<th>(3, 2)</th>
<th>(3, 4)</th>
<th>(4, 1)</th>
<th>(4, 2)</th>
<th>(4, 3)</th>
<th>Tot.</th>
</tr>
</thead>
<tbody>
<tr>
<td>2H</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>9</td>
</tr>
</tbody>
</table>

![Graph](image3.png)

**Fig. 1.** Performance results for networks with $N = W$, with $\beta = 1.4$.  

![Graph](image4.png)

For $N = W = 24$, the percentage of slots not allocated decreases as the tuning latency increases. The BFS and Shuffle algorithms outperform the Two-Hop and Manhattun-uni algorithms. For $N = W = 64$, the percentage of slots not allocated decreases even more as the tuning latency increases. The BFS and Shuffle algorithms still outperform the Two-Hop and Manhattun-uni algorithms.
source to destination over the logical topology. To implement logical topologies with a degree greater than one, nodes are required to tune their transmitters according to a fixed preassigned scheduling, since each node is equipped with only tunable transmitter. When a slot allocation request $R_{sd}$ is received by the network controller, it is allocated on the shortest path of the logical topology; since many equivalent shortest paths exist due to the regularity of the topology, allocations are attempted on all the shortest paths with a residual capacity greater or equal to the bandwidth request, starting from the lightly loaded path. We use in our comparison three regular topologies: the mono-directional Manhattan, the bi-directional Manhattan [16] and the Shuffle [17].

We report results for a network with $N = \{24, 64\}$, and $W = \{24, 32, 64\}$ under uniform traffic. Other values of $N$ and $W$ were also considered, but the results obtained in those cases show no qualitative difference from those reported here, and are omitted for the sake of brevity.

The behavior of each connection $(s, d)$ (i.e., the flow of packets from source $s$ to destination $d$) is driven by a two-state Markov chain whose states are labelled ON and OFF. When the chain is in the OFF state, the connection is inactive, i.e., no packets are generated at source $s$ for destination $d$. While in the ON state, a fixed number $P_{ON}$ of packets is generated in each frame at source $i$ with destination $j$. $P_{ON}$ is selected when the chain enters the ON state, and is uniformly distributed between 1 and 16 slots. The average sojourn time in the OFF state has been fixed to 600 frame times, and the average sojourn time in the ON state has been fixed to 400 frame times.

Slot allocations and de-allocations are performed only at frame boundaries, i.e., all bandwidth requests received by the controller during a frame are handled at the end of the frame, in FCFS ordering.

Note that the frame length strongly influences the system performance, since it drives the network load: as previously noted, the bandwidth corresponding to a request of $R_{sd}$ slots per frame is inversely proportional to the frame duration. For a given request matrix $R$, if the frame is very long, the network is very lightly loaded, and all scheduling policies will manage to accommodate all requests. Conversely, if the frame is too short, the network is strongly overloaded and the probability of blocking requests is very large for all strategies.

Similarly, the scheduling difficulty depends on the tuning latency. If we fix the frame length and the number of requested slots, longer tuning latencies make the scheduling more difficult. However, if we choose the frame length proportional to the tuning latency value, the effects of the tuning latency are more complex. Long tuning latencies imply that the network efficiency is mostly limited by tuning, so that wavelengths are not a bottleneck, and scheduling is relatively easy. On the contrary, short tuning latencies make the system bottleneck shift to wavelengths, so that the scheduling problem becomes harder.

To put in evidence the differences among the considered strategies, it is therefore important to select with care the frame duration, that must be somehow tied to the traffic load and tuning latency. It must be noted that, given a request matrix $R = [R_{sd}]$, the minimum frame duration $F_{\text{min}}$ required to accommodate all requests is bounded by the following inequality:

$$F_{\text{min}} \geq F^* = \max_\sigma \left( \max_i \left( \sum_j R_{sd} + K_s T \right), \max_s \sum_{d \in D_w} R_{sd} \right)$$

where $T$ is the tuning latency, i.e., the number of idle slots between two transmissions of the same source on different wavelengths, $K_s$ is the number of wavelengths on which source $s$ has packets to transmit (hence $\sum_{d} R_{sd} + K_s T$ is the minimum activity period of user $s$, including tuning actions), and $D_w$ is the set of destinations receiving on wavelength $w$ (thus, $\sum_s \sum_{d \in D_w} R_{sd}$ is the minimum activity period on wavelength $w$).

In our case, the request matrix is not deterministically known. However, on average, only 40% of connections are active, due to the ratio between the average duration in frames of the ON and the OFF states. Under this assumption, the lower bound (1) is evaluated to obtain $F^*$, and the frame duration $F$ is chosen to be $F = \beta F^*$.

This approach makes the frame duration $F$ vary with the tuning latency. Given the system setup in terms of offered load and tuning time constraints, we select a frame duration which is reasonably far from the minimum, i.e., for which an optimal allocation strategy should be very close to a complete accommodation of all information flows. This also means that performance curves in the sequel are plotted with different frame sizes for different values of $T$, and that performance indices cannot be compared in their absolute values: while the different strategies are correctly compared for a fixed value of $T$, the values taken by the same strategy at different values of $T$ must be compared with care.

Note that this does not mean that we consider networks in which the frame length is adaptive; this is contrary to the spirit of the scheduling strategies that we are considering (see the definition of the fixed frame problem in Section II). Rather, we wish to compare the effectiveness of different scheduling algorithms for variable tuning latencies, but we want to eliminate the effect of the growing scheduling difficulty for growing tuning latencies.

The performance index we examine is the percentage of slots not allocated as a function of the tuning latency. In Fig. 1 we examine a network with $N = W = 24$ (on the left) and with $N = W = 64$ (on the right). Solid lines identify our two-hop scheduling algorithm, dotted lines the single-hop BFS policy, dashed lines the multi-hop approach based on regular topologies.

It can be observed that the multi-hop approach based on regular topologies provides significant better performance only for very high tuning latencies and for a small number
of nodes. Since in a real system the number of nodes should be around some hundredth and also tuning latencies should be of the order of few tens time slots, this approach seems to be the less promising. Conversely, our two-hop algorithms provides comparable or better performance with respect to the BFS algorithm. These observations hold also when using different values for the parameter set, i.e., by varying the ON-OFF periods characterizing the request arrival or by varying the frame size. For this reason, we won’t present results for the multi-hop regular topology approach in the sequel.

It may be argued that the the two-hop approach may be of smaller benefit with respect to the BFS strategy when the number of wavelengths is smaller than the number of nodes, which is of course a more realistic scenario in a real environment. Fig. 2, referring to a network where \( N = 64, W = 32 \), shows instead that the performance gain is still important.

\[ \text{Fig. 2. Performance results for networks with } N < W, \text{ with } \beta = 2.0. \]

V. CONCLUSIONS

We studied incremental transparent scheduling policies for for all-optical broadcast and select TDM/WDM networks. Transceivers tuning latencies are considered not negligible with respect to the slot time and play an important role in the definition of the scheduling algorithm. Transmission request are scheduled in a fixed length frame by allocating slots in the TDM/WDM frame to source-destination pairs.

We proposed a novel scheduling algorithm that may route some flows from source to destination through some intermediate nodes, following a two-hop approach. Performance results show that significant benefits can be achieved with respect to traditional single-hop approaches or with respect to other multi-hop scheduling algorithm.

REFERENCES


