Computer Networks 52 (2008) 2988–3006

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Routing and scheduling connections in networks that support advance reservations

Emmanouel (Manos) Varvarigos [a,b], Vasileios Sourlas [c], Konstantinos Christodoulopoulos [a,b,*]

[a] Department of Computer Engineering and Informatics, University of Patras, Greece
[b] Research Academic Computer Technology Institute, Patra, Greece
[c] Department of Computer and Communication Engineering, University of Thessaly, Greece

ARTICLE INFO

Article history:
Received 29 October 2007
Received in revised form 1 April 2008
Accepted 18 June 2008
Available online 8 July 2008

Responsible Editor: Prof. A.K. Somani

Keywords:
Multicost routing and scheduling
Advance reservation
Quality of Service
Capacity utilization profiles
Optical Burst Switching

ABSTRACT

A key problem in networks that support advance reservations is the routing and time scheduling of connections with flexible starting time and known data transfer size. In this paper we present a multicost routing and scheduling algorithm for selecting the path to be followed by such a connection and the time the data should start and end transmission at each link so as to minimize the reception time at the destination, or optimize some other performance criterion. The utilization profiles of the network links, the link propagation delays, and the parameters of the connection to be scheduled form the inputs to the algorithm. We initially present a scheme of non-polynomial complexity to compute a set of so-called non-dominated candidate paths, from which the optimal path can be found. We then propose two mechanisms to appropriately prune the set of candidate paths in order to find multicost routing and scheduling algorithms of polynomial complexity. We examine the performance of the algorithms in the special case of an Optical Burst Switched network. Our results indicate that the proposed polynomial-time algorithms have performance that is very close to that of the optimal algorithm. We also study the effects network propagation delays and link-state update policies have on performance.

1. Introduction

Resource reservations is a way to provide Quality of Service (QoS) to end users. For example the resource reservation protocol (RSVP) [2], used for Integrated Services (IntServ), employs resource reservations as a way to meet specific QoS demands [1]. In general, we can distinguish two types of network resource reservations: *immediate reservations*, which are made in a just-in-time manner, and *in advance reservations*, which allow the starting time of the resource usage to be in the future.

Requests for advance reservations contain time- as well as resource-related parameters. According to [10,11], advance reservation requests can be distinguished in several classes. The request is characterized by a source node, a destination node, a bandwidth demand, and one of the following:

- A specified starting time and a specified duration (STSD).
- A specified starting time and an unspecified duration (STUD).
- An unspecified starting time and a specified duration (UTSD).
- An unspecified starting time and an unspecified duration (UTUD).

Of particular interest to us are advance reservation requests that have unspecified (flexible) starting time (UTSD and UTUD).

Since bandwidth is a valuable and sometimes scarce resource, efficient bandwidth management is a key objective of network designers. This is particularly true for

* Corresponding author. Tel.: +30 2610996988.
E-mail addresses: manos@ceid.upatras.gr (Emmanouel (Manos) Varvarigos), vsourlas@inf.uth.gr (V. Sourlas), kchristodou@ceid.upatras.gr (K. Christodoulopoulos).

1389-1286/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.comnet.2008.06.016

multimedia applications where large amounts of content, such as video files, have to be transmitted [3]. Optical Burst Switching (OBS) [4–7] and Grid computing [8] have recently defined two new networking paradigms, where large chunks of data are transferred. OBS was proposed as a way to combine the best attributes of optical circuit switching and optical packet switching. Since bursts may be large, they can easily overload a link if bandwidth reservations are not made for them in advance. Grids introduce new ways to share storage, computing resources, and instruments across geographically separated sites. Typical computations on Grids result in large data transfers among the different sites. This would overload the network unless advance reservations are employed. Moreover, advance reservations in Grids extend the network and include other types of resources, such as computing power, storage, etc.

We propose a multicost algorithm for routing and time scheduling connections with flexible (unspecified) starting time that fall in the UTSD and UTUD categories defined above. The algorithm is based on the time discretization of the capacity utilization profiles of the links, a data structure introduced in [5], which can be used to keep track of the capacity reservations. The algorithm is evaluated through simulations in an Optical Burst Switched network (where a burst can be viewed as a 'small' connection request) but this does not limit its applicability. We also give examples of the application of our algorithms in a Wavelength Division Multiplexing (WDM) environment that employs no or full wavelength conversion capabilities.

The proposed multicost algorithm selects the path to be followed by a connection and the time instant the connection should start transmission so as to minimize the data reception delay, or some other performance criterion. The criterion to be optimized is left flexible until the end of the algorithm and may depend on the user's QoS requirements. The algorithm makes its decisions based on network utilization information available at the source at the time it is executed and thus falls under the general category of feedback-based algorithms. It is worth noting that the proposed algorithm is designed to be employed in a distributed architecture but can be easily extended to function in a centralized manner.

The proposed algorithm consists of two phases: it first computes a set of candidate non-dominated paths, $P_{\text{n-d}}$, for the given source–destination pair. More specifically, we will say, for the purposes of this paper, that a path p_1 dominates another path p_2 for a given request if the propagation delay of p_1 is smaller than that of p_2, and also path p_1 is available for scheduling the connection (at least) at all time intervals at which path p_2 is available. After the set of non-dominated paths has been calculated, the routing and time scheduling decision is made based on the connection's parameters and QoS requirements. For example, if the duration of the connection is known, it is easy to find from the set of non-dominated paths the path resulting in the minimum reception time of the data at the destination.

A serious drawback of the algorithm outlined above is that the number of non-dominated paths may be exponential, and the algorithm is not guaranteed to finish in polynomial-time. This was expected since even versions of the scheduling problem simpler than the one considered in this paper are NP-hard. To obtain polynomial-time algorithms, we use two mechanisms to prune the set of candidate paths. For the first mechanism we define a pseudo-domination relationship between paths. Based on this relationship, a set $P_{\text{n-ps-d}} \subseteq P_{\text{n-d}}$ of non-pseudo-dominated paths is calculated, by killing pseudo-dominated paths, and the path from this set that minimizes the data reception time at the destination is selected. Although this set is not guaranteed to always contain the optimum path, our performance results indicate that by appropriate choosing the pseudo-domination relationship we can obtain performance that is very close to that of the optimal multicost algorithm. The second mechanism is a branch-and-bound alteration to the optimal algorithm, which assumes that we know in advance the criterion to be optimized. If, for example, we want to minimize the data reception time at the destination, we bound the reception time and prune a candidate path if the best-case reception times of the path extensions would be inferior. In this way, we do not loose the optimum solution for the specific optimization function.

We use simulations to evaluate the performance of the optimal multicost routing and scheduling algorithm, the pseudo-domination polynomial-time heuristic variation and the branch-and-bound algorithm for an OBS network. We compare it to that of a typical Dijkstra shortest path algorithm and a Dijkstra shortest path with Collision Avoidance algorithm. Our results show that the proposed multicost algorithm and its polynomial-time variations can lead to significant improvements in the average end-to-end delay experienced by the bursts. The optimal multicost algorithm outperforms all other examined algorithms, but requires the highest number of operations. The number of searched paths and the number of operations required are reduced by employing the branch-and-bound mechanism, without loosing the optimal solution for the specific optimization function. The polynomial-time heuristic variations of the multicost algorithm also have performance close to that of the optimal algorithm, while maintaining the number of operations at low levels. We also verify that the performance improvements obtained by using our proposed schemes are more significant when the network propagation delays are small, a typical characteristic of feedback-based algorithms. Finally, we look at the effect the link-state update mechanism has on the performance of the algorithms examined.

The remainder of the paper is organized as follows. In Section 2 we report on previous work. In Section 3 we present several useful formats for recording the utilization profile of a link and give examples of networks that can use these data structures. In Section 4 we describe the routing and scheduling problem under consideration. In Section 5 we present the multicost routing framework. We describe an algorithm for finding the set of non-dominated paths and present an algorithm for obtaining the best path and the starting time of the reservation. In Section 6 we present two mechanisms to appropriately prune the paths in order to find multicost routing and scheduling algorithms that are of polynomial complexity. Section 7 presents the performance results for the algorithms proposed in Sections 5 and 6 in an Optical Burst Switched network. Our conclusions follow in Section 8.

2. Related work

The topic of advance reservations in traditional and in high-speed networks has been extensively examined. Lately, research efforts in the areas of Optical Burst Switching (OBS) and Grid computing have examined advance reservations in different contexts.

In [12] the authors describe a model for resource reservations in advance and discuss issues that must be resolved in this context. The requirements of the users in a distributed advance reservation environment are discussed in [13]. The proposed design is implemented in the Tenet 2 Protocol suite. In [14] the authors discuss how to provide advance reservations on top of RSVP. The authors in [15] propose an architecture where users can make advance reservations through agents who know the topology and the static link capacities, and thus can select suitable routes to reserve the resources in advance. An Efficient Reservation Virtual Circuit (ERVC) protocol for high-speed networks that uses advance and timed reservations is proposed in [5].

However, the above references mainly address signaling protocols for advance reservations, or they focus on the case where the starting times of the reservations are fixed. The authors in [16] first introduced the concept of advance reservation scheduling and advance-reservation aware routing algorithms, which is the problem that we address in this paper. More specifically, [16] proposed several algorithms for advance reservations when the starting times are specified or are flexible. The computational complexity of these algorithms is also discussed. Advance reservations as part of a Routing and Wavelength Assignment (RWA) problem in WDM networks is examined in [10], where various algorithms for requests with flexible starting times are described. In [11], the architecture and the implementation of a coordinated intra- and inter-domain control plane capable of flexible advance reservations are presented.

An interesting type of networks where our multicost routing and scheduling algorithm are directly applicable are Optical Burst Switched networks. In OBS networks, ingress nodes assemble data packets destined for the same node and having the same QoS requirements into bursts. When a burst is completed, the ingress node sends out a control packet to the egress node to setup the Label Switched Path (LSP) to be followed and reserve bandwidth for the burst to be transmitted.

Signaling protocols proposed so far for OBS networks can be categorized into two main classes: two-way (tell-and-wait) and one-way (tell-and-go) protocols. In tell-and-go protocols, a control packet (called Burst Control Header or BCH) is usually sent out-of-band and leads the burst by a small "Time Offset". The BCH packet contains information on the burst duration and reserves, at each intermediate link, the requested bandwidth for the burst that follows shortly after it. If the reservation at an intermediate node cannot be performed the burst is discarded, assuming there are no Fiber Delay Lines (FDLs) to store it. On the other hand, in tell-and-wait protocols, an end-to-end connection is fully established before the transmission of the burst can start, and ACK/REJ packets are utilized so as to inform the source for the reservation. An example of a tell-and-wait protocol is the Efficient Reservation Virtual Circuit (ERVC) protocol [5], while recent research efforts include the WR-OBS [18] and the Efficient Burst Reservation Protocol (EBRP) [19]. One-way reservation schemes have also received increasing attention. The Ready-to-Go Virtual Circuit (RGVC) protocol [17] is an early attempt for a one-way reservation scheme in optical networks. The Horizon [6], the Just Enough Time (JET) [4], and the Just In Time (JIT) [20] are also one way protocols. A key problem in OBS networks is the design of efficient algorithms for scheduling bursts to the optical channels. Channel scheduling algorithms can be classified into two categories: *with* [21] and *without* void filling [6]. In our work we are mostly interested in algorithms that incorporate void filing.

In OBS networks, a burst transmission request can be viewed as a connection that requires an advance reservation with an unspecified starting time and a specified duration (UTSD). The starting time of the transmission (Time Offset) is calculated by taking into account the protocol used (one- or two-way) and the number of hops. The time offset can also be chosen so as to provide QoS differentiation, as in [22], where a high loss priority class is given a larger extra offset time in order to make earlier wavelength reservation than lower priority classes. The choice of the offsets is further examined in [23], where an adaptive scheme that depends on the utilization of the links and the burst losses in the core is proposed. In [24] a dynamic Wavelength Routed OBS architecture is introduced, in which centralized control is employed to provide resource reservation efficiency, low delay, and QoS differentiation.

Grids introduce new ways to share resources across geographically separated sites by establishing a global resource management architecture. The efficiency of a Grid system depends on the development of sophisticated resource management schemes capable of allocating resources to users based on their QoS requirements. Advance reservations in this context involve the ability of the scheduler to guarantee the availability (of any kind) of resources at a particular time in the future. The Globus Architecture for Reservation and Allocation (GARA) [9] is a framework for advance reservations that treats in a uniform way various types of resources, such as communication, computation, and storage resources. Although GARA has gained popularity in the Grid community, its limitations in coping with current application requirements and technologies led to the proposal of the Grid Quality of Service Management (G-QoSm) framework [25]. Resource Brokers and algorithms that support advance reservations are discussed in [26–28].

We must note here that the multicost algorithmic approach proposed in the present paper is rather different than the multiconstrained algorithmic approaches used in other works for different problems. Multiconstrained algorithms solve problems in subsets of the solution space considered by multicost algorithms [29]. In the related literature, multiconstrained algorithms have mainly been used for QoS routing problems. In [30] the authors proved that QoS routing with QoS parameters being the bandwidth and the delay is not NP-complete. The authors in [31] considered the problem of finding multiple shortest

paths satisfying the constraints. Heuristic algorithms for the *restricted shortest path* (RSP) problem with parameters being the delay and the cost, proven to be NP-complete, have been examined in [32,33]. The general Multiconstrained Path Problem (MCP) is discussed in [34–37], where heuristic algorithms are also proposed. To the best of our knowledge, the present work is the first time a multicost algorithm is used to address a scheduling problem. To this end, temporal information in the form of quantized utilization profiles, as described in the next section, is included in the multicost formulation, and appropriate operations such as addition and domination are defined in this context.

3. Link utilization profiles

In a network that employs advance reservations, each node needs to keep a record of the capacity reserved on its outgoing links, as a function of time, in order to perform channel scheduling and reservation. Assuming that each connection reserves a constant amount of bandwidth for a given time duration, the *utilization profile* $U_l(t)$ of a link l is a stepwise function with discontinuities at the points where reservations begin or end, and is updated dynamically with the admission of each new reservation. We define the *capacity availability profile* of link l of capacity C_l as $C_l(t) = C_l - U_l(t)$. Since, a source trying to route a connection of rate r is only interested in time periods at which $C_l(t) > r$, we also define the *r-capacity–availability profile* $C_l(t;r)$ of link l as the binary function $C_l(t;r) = 1$ if $C_l(t) \geqslant r$ and $C_l(t;r) = 0$, otherwise.

Since the duration of the connections that pass through a link and the link propagation delays are arbitrary, the time intervals during which the requested bandwidth r is available and the times at which these intervals start and finish are also arbitrary. In order to obtain from the r-capacity availability profile $C_l(t;r)$ of link l a data structure that is easier to handle in an algorithm, we discretize it in time steps (or slots) of duration τ to obtain the *binary r-capacity availability vector* $\widehat{C}_l(r)$, abbreviated CAV, as the vector whose k-th entry is

$$\left\{\widehat{C}_l(r)\right\}_k = \left\{\begin{array}{ll} 1, & \text{if } C_l(t;r) = 1, \text{for all} (k-1)\tau \leqslant t \leqslant k\tau \\ 0, & \text{otherwise.} \end{array}\right\},$$
$$k = 1, \ldots, u.$$

In Fig. 1 we illustrate the link utilization profiles.

The discretization of the time axis in steps of duration τ results in some loss of information (unless all connection durations and network propagation delays are multiples of τ). If D is the maximum allowable delay for a connection we wish to route, we are only interested in values of $C_l(t;r)$ before time D from the present time, or, equivalently, we can assume that the binary r-capacity availability vector $\widehat{C}_l(t;r)$ has $u = \lceil D/\tau \rceil$ entries. In any case, the choice of the discretization step τ provides a tradeoff between the required accuracy – efficiency, since it determines the size of the binary utilization vectors and the processing overhead.

The data structures defined above can be useful in a number of network settings. For example, in an optical WDM network with full wavelength conversion and $w$wavelengths per link, each of capacity C, the capacity of a link l is $C_l = w \cdot C$. A connection that wants to reserve k wavelengths, $1 \leqslant k \leqslant w$, has requested rate $r = k \cdot C$. If we can find a path of available capacity at least r then the connection can be established (since we assume full wavelength conversion capabilities). Therefore, the r-capacity availability profile $C_l(t;r)$ and the *r-capacity availability vector* $\widehat{C}_l(r)$ can be used in this kind of network with the difference that $C_l(t;r)$ then takes w discrete values. If no wavelength converters are available, each link needs to keep track of the utilization profile of each of its w wavelengths separately. Thus a node has to maintain w binary (a wavelength can be reserved or not for a given time) utilization profiles for each outgoing link. In that case, where wavelength conversion is not available, the network can be viewed as w "parallel" networks, each having a single wavelength. The profiles $C_l(t;\lambda)$ and $\widehat{C}_l(\lambda)$ can then be used for each wavelength λ (equivalently, for each parallel network) and the dependence on r is no longer present. The case of Optical Burst Switched networks falls in one of the two aforementioned categories.

The algorithms that we will propose can be used in any network that supports advance reservations and for which we can convert its link utilization profiles to a binary vector such as $\widehat{C}_l(r)$. They can also be used in the case where the network can be viewed as consisting of w parallel networks (like the WDM network with no wavelength conversion, described above), each characterized by a set of binary link utilization profiles. In that case we have to run the algorithm for obtaining the set of non-dominated paths (Section 5.1) w times (once for each parallel network) and then choose the solution that satisfies our requirements among all these sets.

In order to simplify notation, in the remainder of the paper and when no confusion can arise, we will denote the profiles $C_l(t;r)$ and $\widehat{C}_l(r)$ of a link l by $C(t)$ and $\widehat{C}$, respectively, suppressing the dependence on l and r.

4. The routing and scheduling problem under consideration

The routing and time scheduling problem in a network that supports advance reservations is defined as follows. We are given a network with links$l$ of known propagation delays d_l, and a source node S that wishes to reserve a certain amount of bandwidth r, for a given duration B, to serve a connection to a specific destination (egress) node E. If the duration is not known in advance we have $B = \infty$. We are also given the utilization profiles $U_l(t)$, or equivalently the capacity–availability profiles $C_l(t)$ of all links l. We also assume that there is an upper bound D on the maximum delay the connection can tolerate, which corresponds to the latest time by which the last bit of the data to be transferred by the connection must have arrived at the destination; if this deadline parameter cannot be met, the request should be rejected. Even when there is no limit D on the maximum allowable delay, we can still assume that the dimension u of the capacity availability vectors is finite, corresponding to the latest time (relative to the present

time) for which reservations have been made on the network links plus the connection duration. Given this information, we want to find a feasible path to route the connection and the time at which the connection should start so as to optimize some performance criterion, such as the number of hops, or the propagation delay, or the data reception time at the destination. Fig. 2 presents an instance of the problem.

After choosing the best available path, a tell-and-go or a tell-and-wait scheme is used to establish the connection and reserve the requested capacity. In either case the connection may fail since the utilization profile at some intermediate link may have changed by the time the setup packet arrives at that link. This is a problem that cannot be avoided by any routing and scheduling algorithm in a network that has nonzero propagation delays. However,

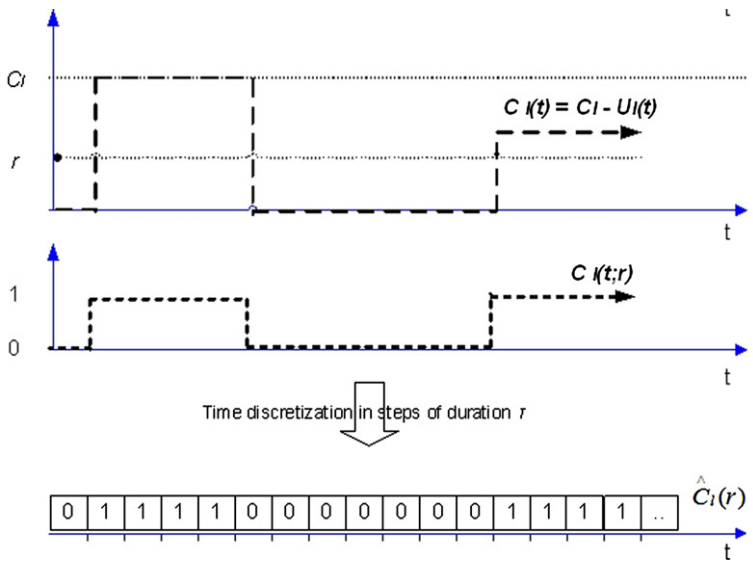


Fig. 1. The capacity availability profile $C_l(t)$, the r-capacity availability profile $C_l(t;r)$, and the binary r-capacity availability vector $\hat{C}_l(r)$ of a given link l of capacity C_l when the rate requested by the connection is r and the discretization step is τ.

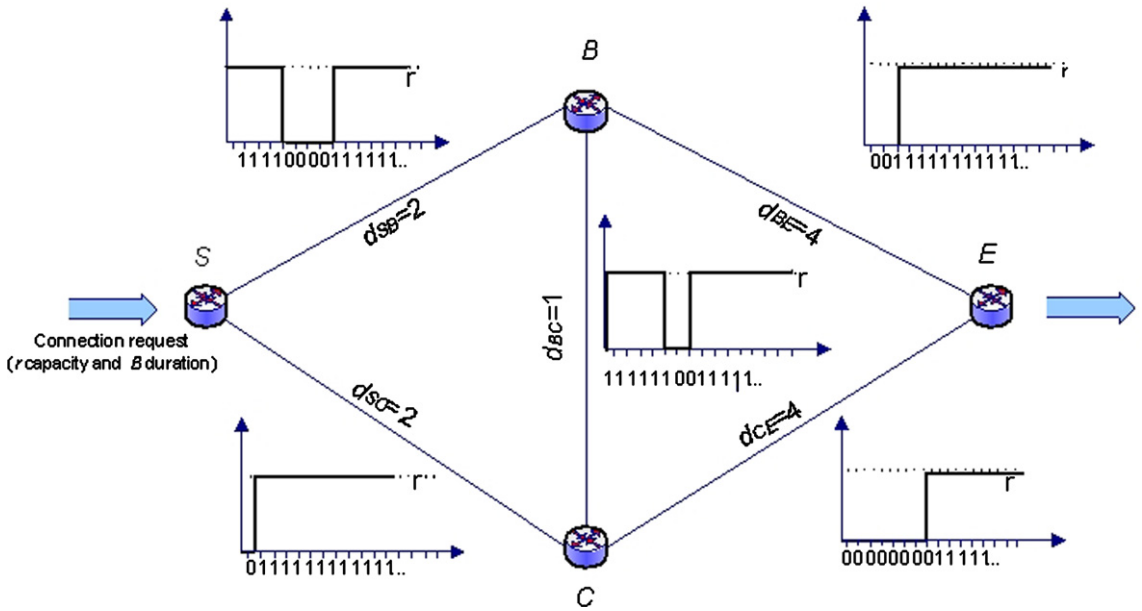


Fig. 2. A connection requesting capacity r for duration B arrives at node S with destination node E. Each link is characterized by its propagation delay (in τ-time units in the figure) and its binary r-capacity availability vector.

the performance results will show that even when our proposed feedback-based algorithms use somewhat outdated information at the source, the connection blocking probability will be substantially smaller than when using an algorithm that does not consider link utilization information.

4.1. Binary capacity availability vector of a path

Assuming routing decisions are made at the sources (source routing), the binary capacity availability vectors of all network links should be gathered continuously at all nodes, over a network control plane that must exist, in order to be used for routing and scheduling by the source nodes.

In the example presented in Fig. 2, consider a path p_{SBE}, where S is the source and E is the destination node, and let $\widehat{C}_{SB} = (c_{1,SB}, c_{2,SB}, \ldots, c_{u,SB})$ and $\widehat{C}_{BE} = (c_{1,BE}, c_{2,BE}, \ldots, c_{u,BE})$ be the binary capacity availability vectors of links l_{SB} and l_{BE}, respectively. In order to compute the capacity availability vector (CAV) of path p_{SBE}, we first gather the CAVs of the links that comprise it to the source node S. The CAV $\widehat{C}_{SB}$ is maintained at node S, so only $\widehat{C}_{BE}$ has to be transmitted from node B to node S. Upon its arrival at S, $\widehat{C}_{BE}$ is left shifted by d_{SB} bits (propagation delay of link l_{SB} in τ-time units) to purge utilization information that corresponds to time periods that have already expired, obtaining in this way $\widehat{C}_{BE}(S)$. More formally, $\widehat{C}_{BE}(S) = \mathrm{LSH}_{d_{SB}}(\widehat{C}_{BE})$ where $\mathrm{LSH}_{d_{SB}}$ is the left-shifting operation. We then shift the resulting CAV by a further d_{SB} positions to the left to take into account the propagation delay that any data sent from Swill suffer in order to reach node B (assuming the propagation delay of each link is the same in both directions). We finally execute a bitwise Boolean AND operation, denoted by "&", among the CAVs in order to compute the binary utilization vector of path p_{SBE}. Fig. 3 illustrates the procedure for finding $\widehat{C}_{SBE}$ from the CAVs $\widehat{C}_{SB}$, $\widehat{C}_{BE}$ of links l_{SB} and l_{BE} and the propagation delay d_{SB} of link l_{SB}.

More formally, the CAV of a path can be obtained from the CAVs of the links that comprise it using the associative operator $\oplus$ between binary vectors defined as follows:

$$\widehat{C}_{SBE} = \widehat{C}_{SB} \oplus \widehat{C}_{BE} = \widehat{C}_{SB} \& \mathrm{LSH}_{d_{SB}}(\widehat{C}_{BE}(S)) = \widehat{C}_{SB} \& \mathrm{LSH}_{2d_{SB}}(\widehat{C}_{BE}). \quad (1)$$

Note that if node S transmits data at the τ-time intervals indicated by 1's in the binary capacity availability vector $\widehat{C}_{SBE}$ of path p_{SBE}, the data are guaranteed (if no reservations are performed in the meantime at the intermediate nodes) to find available capacity when they arrive at all intermediate links of the path. This procedure can be extended so as to compute paths with more than two hops. Each time we extend a path by adding a new link we should left-shift its CAV by a number of positions (τ-time units) equal to twice the delay of the previous path (previously added links); shifting by one propagation delay purges outdated information and shifting by one more propagation delay accounts for the forward propagation delay.

5. Multicost routing and scheduling connections

In what follows we present a multicost routing and scheduling algorithm for networks that use timed and in advance reservations. In multicost routing [29], each link l is assigned a vector V_l of cost parameters, as opposed to the scalar cost parameter assigned in single-cost routing. In our initial formulation, the cost parameters of a link l include the propagation delay d_l of the link and its binary capacity availability vector $\widehat{C}_l$, that is,

$$V_l = (d_l, \widehat{C}_l) = (d_l, c_{1,l}, c_{2,l}, \ldots, c_{u,l}),$$

but they may also include other parameters of interest (such as the number of hops, or the capacity availability profile instead of the binary capacity availability vector, or some other parameters). A cost vector can then be defined for a path $p$consisting of links 1, 2,…,k, based on the cost vectors of its links, according to

$$V(p) = \odot_{l \in p} V_l \stackrel{\text{def}}{=} \left(\sum_{l \in p} d_l, \oplus_{l \in p} \widehat{C}_l \right), \quad (2)$$

where $\oplus$ is the associative operator defined in Eq. (1).

We will say that a path p_1 dominates another path p_2 for a given connection and a given source–destination pair if the propagation delay of p_1 is smaller than that of p_2, and also

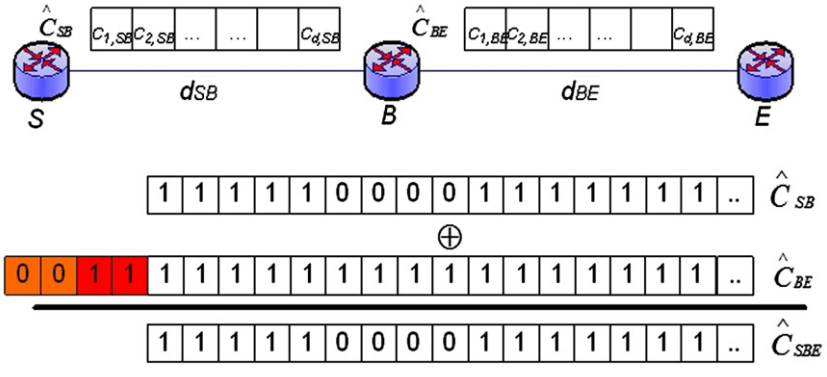


Fig. 3. Calculation of the path capacity availability vector $\widehat{C}_{SBE}$. $\widehat{C}_{BE}$ is shifted by $2 \cdot d_{SB} = 4$ τ-time units ($d_{SB} = 2$ in this example), before the Boolean AND operation is applied.

path p_1 is available for scheduling the connection (at least) at all time intervals at which path p_2 is available. Formally:

p_1 dominates p_2 (notation : $p_1 > p_2$) **iff**

$$\sum_{l\in p_1} d_l < \sum_{l\in p_2} d_l \quad \text{and} \quad \oplus_{l\in p_1} \widehat{C}_l \geqslant \oplus_{l\in p_2} \widehat{C}_l, \tag{3}$$

where the vector inequality "$\geqslant$" should be interpreted component wise. The set of non-dominated paths $P_{\text{n-d}}$ for a given connection and source–destination pair is then defined as the set of paths with the property that no path in $P_{\text{n-d}}$ dominates another path in $P_{\text{n-d}}$.

The routing and scheduling algorithm we propose consists of two phases: given a source–destination pair (S and E, respectively), the set $P_{\text{n-d}}$ of non-dominated paths between them is calculated first, and then an optimization function $f(V(p))$ is applied to the cost vector of each path $p \in P_{\text{n-d}}$ to select the optimal one. Following the routing decision, a tell-and-wait or a tell-and-go protocol is finally used to establish the connection and reserve the requested capacity.

5.1. Algorithm for computing the set $P_{n\text{-}d}$ of non-dominated paths

In this section we present an algorithm for obtaining the set $P_{\text{n-d}}$ of non-dominated paths from a given source node S to a given destination node E. The algorithm computes the non-dominated paths from the source to all network nodes (including, of course, E) and can be viewed as a generalization of Dijkstra's algorithm that only considers scalar link costs. The basic difference with Dijkstra's algorithm is that instead of a single path, a set of non-dominated paths between the origin and each node is obtained. Thus a node for which one path has already been found is not finalized (as in the simple Dijkstra case) since we can find more "non-dominated" paths to that node later.

The algorithm to obtain the set of non-dominated paths between an origin node and all other nodes of the network is now formally described for the case where the link cost vectors consist of one additive parameter (delay or number of hops), and the binary capacity availability vector of the link.

We denote by V_l the cost vector of link l. Each path is represented by a label that includes the cost vector associated with it and the first hop to the source using that path. The source that serves the connection is taken to be node S.

We let W_i be the set of labels of the paths from node S to a node n_i, and $W = \cup_{n_i \neq S} W_i$ be the set of all labels. Initially, every node has a single label corresponding to the link (if any) that connects it directly to the origin node. In each subsequent step, the algorithm marks labels (equivalently paths) from the set W as final. We let $W^f \subseteq W$ be the subset of all final labels for all the nodes, and $W_i^f \subseteq W_i$ be the set of final labels for node n_i. We also let T be the set of nodes with at least one final label. The algorithm can now be described as follows:

Step 0 (Initialization): $W = \{V_{p_1}, V_{p_2}, \ldots, V_{p_N}\}, W^f = \{\}, T = \{\}$, *where* V_{p_i} *is the label of the path* p_i *(if any) leading directly from node S to node* n_i.

Step 1 (Choosing the optimum label): The label of path p whose cost vector minimizes the additive component is chosen. In case of a tie, we look at the second component, which is the binary capacity availability vector, and a dominant one is chosen. If V_{p_l} *is the cost vector of the chosen label and* n_i *is the node to which it leads, then the following updates are performed:*

$$W_i^f = W_i^f \cup \{V_{p_i}\}, \quad W^f = W^f \cup \{V_{p_i}\}, \quad T = T \cup \{n_i\}.$$

Step 2 (Obtaining the new labels): The neighbors of node n_i, *which may or may not belong to the set T, are now considered and are given new labels (except for the origin node and the node specified as the previous node in the label). The new label for the path* p_j *leading to the neighbor* n_j *of node* n_i *by extending the path* p_i *through the link* $l = (n_i, n_j)$ *is then computed as follows. The new cost vector is updated according to* $V'_{p_j} = V_{p_i} \odot V_l$ *where* V_l *is the label of link* $l = (n_i, n_j)$, *and "*$\odot$*" represents the operation defined in Eq. (2).*

Step 3 (Discarding dominated paths): Each neighbor considered in step 2 compares its new label with its previous labels using the domination relation of Eq. (3). Let n_j *be one of the neighbors of node* n_i, V'_{p_j} *the new label obtained from step 2 and* W_j *be the set of labels for this node. The new label has to be compared with the labels* $V_{p_j} \in W_j$ *(both final and non-final). If any cost vector in* W_j *dominates* V'_{p_j}, *then* V'_{p_j} *is discarded and* W_j *does not change. If the new cost vector* V'_{p_j} *is not dominated by any of the vectors in* W_j, *then* V'_{p_j} *is added to the set* W_j *and W, so that* $W_j = W_j \cup \{V'_{p_j}\}$ *and* $W = W \cup \{V'_{p_j}\}$. *If the new vector dominates one of the vectors in* W_j, *then* W_j *and W are updated by eliminating the dominated vectors and adding the new vector* V'_{p_j}. *Note that it is not possible for the new vector to dominate an existing vector and be dominated by another one at the same time.*

Step 4 (Termination): If after an iteration the set W^f *is equal to W, the algorithm is completed. Otherwise (when there are still some labels to be chosen), we go back to Step 1.*

The set $P_{\text{n-d}}$ of non-dominated paths from the given source S to the given destination node E is the final set W_E^f.

The pseudo-code of the algorithm described in this section is given in Appendix A.

A serious drawback of the algorithm described above is that the number of non-dominated paths may be exponential, and the algorithm is not guaranteed to finish in polynomial-time. This was expected since versions of the scheduling problem even simpler than the one considered in this paper are NP-hard. In Section 6 we propose polynomial-time heuristic algorithms that have performance close to that of the optimal algorithm.

5.2. Choosing the optimal paths to schedule the connection

In the previous paragraph we obtained the set $P_{\text{n-d}}$ of non-dominated paths from the given source node S to the given destination node E. Each path in the set $P_{\text{n-d}} = \{p_1, p_2, \ldots, p_k\}$ of non-dominated paths comes with

a cost vector, consisting of its propagation delay in τ-time units and its CAV (Fig. 4). In the second phase of the algorithm we apply an optimization function $f(V(p))$ to the cost vector of each path $p \in P_{\text{n-d}}$ to select the optimal path. This optimization function depends on the duration and QoS requirements of the connection and on the reservation protocol (tell-and-go or tell-and-wait) to be used for establishing the connection. For example, if a tell-and-wait protocol is used, which requires a pre-transmission delay equal to an end-to-end roundtrip delay T_{RTT} (measured in τ-slots), the optimization function should not depend on the first T_{RTT} entries of the path's CAV.

In general, the optimization function $f(V(p))$ applied to the cost vector of a path to compute the final (scalar) path cost has to be monotonic in each of the cost components. For example, it is natural to assume that it is increasing with respect to delay, decreasing with respect to capacity, decreasing with increased capacity availability (as expressed by the CAV), etc. Each component of the path cost vector is characterized by the way it is obtained from the links' cost vector components (e.g., addition for the delay d_l, or $\oplus$ for the capacity availability vector $\widehat{C}_l$ of link l) and also by the optimization criterion that has to be applied to it (e.g., minimization, in the case of the delay, which in all practical cases has to be minimized, or maximization in the case of capacity, etc.).

Assuming that we want to serve a connection request of duration b (in τ-time units) from source to destination using a tell-and-go connection establishment scheme, the following process is used to select the best path from the k non-dominated paths that have been found between them:

Step I: *For the capacity availability vector $\widehat{C}_p$ of each non-dominated path p we calculate the first position $R_p(b)$ after which $\widehat{C}_p$ has b consecutive ones. In other words, $R_p(b)$ is the earliest time at which the data of a connection with duration b can start transmission on path p.*

Step II: *We can use one the following alternative algorithms to select the path to route the connection:*

Step IIa: *Minimum reception time algorithmSelect the path p that minimizes the reception time, defined as the time the last bit of the connection reaches the destination:*

$$d_p + R_p(b) + b,$$

where d_p is the propagation delay of path p in τ-slots. If the minimum reception time is larger than the maximum allowable delay D (Section 3), then the connection is not served (rejected).

Step IIb: *Minimum propagation delay algorithmAmong the paths p with reception time $d_p + R_p(b) + b$ choose the one that has the smallest propagation delay d_p.*

Step IIc: *Minimum hop algorithmBy extending the algorithm given in Section 5.1 in a straight-forward way, we can include in the cost vector of a path, in addition to its propagation delay and CAV, the number of hops of the path. In this case, we can choose among the paths p that have reception time $d_p + R_p(b) + b$ the one that uses the minimum number of hops h_p.*

Step III: *Updating the CAV of the chosen pathHaving chosen the path to schedule the connection, the next step is to update the CAVs by converting the ones in the appropriate positions of the CAV to zeros (meaning capacity is no longer available at the corresponding time periods).*

Step IV: *Sending the link-state updates to the other nodes*

Since ingress nodes need to use link-state information to compute the non-dominated paths, link-state update packets should be sent to the other source nodes whenever the CAV of a link changes (see Section 5.3).

For the purpose of being specific, the optimization function we use in the rest of this paper is the one that selects the path that leads to the minimum reception time of the data of the connection at the destination, that is, we use Step IIa.

The procedure described above assumes a tell-and-go connection establishment protocol. If a tell-and-wait protocol is used instead, we simply have to redefine $R_p(b)$ as the first position after $2 \cdot d_p$ (the round trip propagation delay of path p) after which $\widehat{C}_p$ has b consecutive ones.

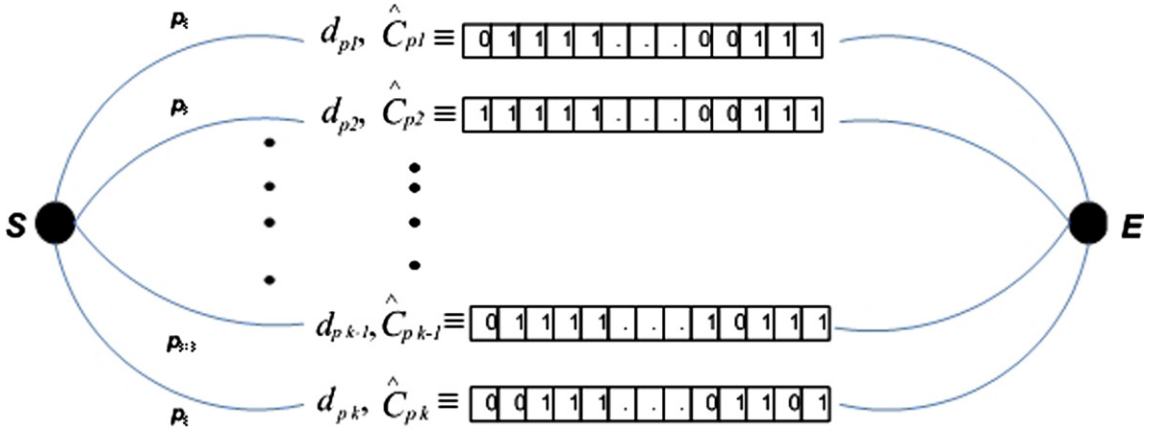


Fig. 4. The set of the non-dominated paths between the source node S and the destination node E.

The pseudo-code of the algorithm described in this section is given in Appendix B.

5.3. Link-state update mechanisms

We distinguish two cases regarding the way the link-state updates are made, to be referred to as *update-upon-reservation* and *update-upon-selection*. We assume that the chosen path is $p = \{n_0, n_1, \ldots, n_h\}$, where $n_0 \equiv S$ and $n_h \equiv E$.

In the *update-upon-reservation* approach the updates are performed during the setup process. An intermediate node $n_i \in p$ that receives a setup packet checks if it can schedule the connection on the requested link $l = (n_i, n_{i+1})$ for the desired time interval. If this is feasible, the reservation information is communicated from node n_i to all source nodes to whom it may be useful. More precisely, an update control packet is sent from node n_i to every node n for which we have

$$2 \cdot d_{n_i,n} \leqslant R_p(b) + b, \tag{4}$$

where $d_{n_i,n}$ is the shortest path delay from node n_i to node n (in τ-slots), $R_p(b)$ is the time after which the connection is scheduled to start at the source S (defined in Step I of Section 5.2) and b is the connection duration in time slots.

In the *update-upon-selection* approach, the link update packets are transmitted by the source S as soon as the selection of the path to be followed has been made. More precisely, S sends an update message to node n to inform it that the links (n_i, n_{i+1}), $i = 0, \ldots, h-1$, will be reserved for the time intervals $[d^p_{S,n_i} + R_p(b), d^p_{S,n_i} + R_p(b) + b]$, $i = 0, \ldots, h-1$, respectively, if

$$2 \cdot d_{S,n} \leqslant d^p_{S,n_i} + R_p(b) + b, \tag{5}$$

where d^p_{S,n_i} is the delay to reach intermediate node n_i over the chosen path p in τ-slots:

$$d^p_{S,n_i} = \sum_{k=0}^{i-1} d_{n_k,n_{k+1}}.$$

Source S does not send a separate message to node $n$for every link (n_i, n_{i+1}) that satisfies Eq. (5), but collects the corresponding updates and transmits them in a single packet to keep the control overhead low.

As mentioned earlier, the connection setup process may fail if the utilization profile at an intermediate link $l = (n_i, n_{i+1})$ has changed by the time the setup packet arrives at n_i. In case of a rejection and assuming the *update-upon-selection* technique is used, node n_i has to inform other nodes about the cancellation of the reservations over the remaining path. Moreover, a REJ packet is forwarded upstream on the path to cancel the reservations made and inform other nodes for this cancellation (using the dual of the *update-upon-reservation*process).

5.4. Synchronization of link utilization profiles

The proposed multicost algorithm is sensitive to synchronization errors between the link profiles, in which case the data arriving at a node may find the capacity reserved for them occupied by a different session. Such errors may be due to inaccuracies with which link lengths are known, the quantization of time (Section 3), and uncertainties in the corresponding propagation delays due to temperature variations. Synchronization has been a key issue in various kinds of networks, such as TDM, packet-based communication, etc., and various time standards have been proposed for such networks. With GPS, time accuracy on the order of half microsecond from UTC is easily achievable. For the proposed algorithm, apart from using synchronization techniques between the switches, guardbands can be also employed. For example if a connection request is of size b slots, the proposed algorithm could be used to search for the scheduling of a connection with $b + g$ size, where g is the quardband in slots. The value of g can then be chosen according to the expected lack of synchronization and timing uncertainties, with the tradeoff of wasting network capacity when using large values of g. However, even with $g = 1$ slot, the synchronization between the nodes can deviate up to $\tau/2$ (where τ is the time slot duration), which is acceptable for τ in the order of tens of μs.

6. Polynomial-time algorithms

6.1. Polynomial-time algorithm for computing the set of non-pseudo-dominated paths

The basic idea in the polynomial-time algorithms we will propose in this section is the following. We define a pseudo-domination relationship $>_{ps}$ between paths, which has weaker requirement than the domination relationship $>$ defined in Eq. (3), in the sense that two paths may not dominate each other with respect to the $>$ relationship, but one of them may dominate the other with respect to the $>_{ps}$ relationship. This pseudo-domination relationship can be used in step 3 of the multicost routing algorithm of Section 5.1 to prune (kill) paths, yielding a set $P_{\text{n-ps-d}} \subseteq P_{\text{n-d}}$ of non-pseudo-dominated paths that has considerably smaller (polynomial) cardinality than $P_{\text{n-d}}$. One of the algorithms given in Section 5.2 can then be applied to the paths in $P_{\text{n-ps-d}}$. The chosen path is not guaranteed to be the optimal one over all paths, but it is often a good path, as our performance results indicate, provided that the pseudo-domination relationship $>_{ps}$ is defined wisely, so as not to eliminate good paths.

To be more specific, we define two new metrics for a link l. The first metric is called the *slot availability weight*

$$w_l = \text{weight}(\widehat{C}_l),$$

where the weight( ) of a binary vector represents the total number of 1's in the vector.

The second metric is the *b-consecutive slot availability* of a link l, denoted by $L_l(b, \widehat{C}_l)$, defined as the total number of runs of consecutive 1's in $\widehat{C}_l$ that have length equal to the connection duration b.

For example, if the CAV of a link$l$ is the vector

$$\widehat{C}_l = (00111101001100011100011),$$

we have $w_l = 12$, $L_l(3, \widehat{C}_l) = 3$, $L_l(2, \widehat{C}_l) = 7$.

In the following we present two polynomial-time heuristic variations of the optimal multicost algorithm of Sec-

tion 5 that use different pseudo-domination relationships to prune the set of candidate paths, based on the *slot availability weight* and the *b-consecutive slot availability* metrics. Note that the path capacity availability vectors are still maintained by the algorithms and combined with the associative operator $\oplus$ given in Eq. (1), but the domination relation used to prune the paths is changed.

1. *Availability weighting algorithm*
   The algorithm is similar to the one in Section 5.1, but the pseudo-domination relation used to prune paths is defined as

$$p_1 \text{ pseudo-dominates } p_2 (p_1 >_{ps} p_2) \textbf{iff}$$
$$\sum_{l\in p_1} d_l < \sum_{l\in p_2} d_l \text{ and weight}(\oplus_{l\in p_2} \widehat{C}_l) \leqslant \text{ weight } (\oplus_{l\in p_1} \widehat{C}_l) \quad (6)$$

   A set of pseudo-non-dominated paths can be found for each source–destination pair. The procedure of Section 5.2 is then used to find the path that minimizes the reception time of the data at the destination, or some other minimization criterion.

2. *b-consecutive slot availability algorithm*
   The algorithm is similar to the one in Section 5.1, but the pseudo-domination relation used to prune paths is defined as

$$p_1 \text{ pseudo-dominates } p_2 (p_1 >_{ps} p_2) \textbf{ iff}$$
$$\sum_{l\in p_1} d_l < \sum_{l\in p_2} d_l \text{ and } L(b, \oplus_{l\in p_2} \widehat{C}_l) \leqslant L(b, \oplus_{l\in p_1} \widehat{C}_l) \quad (7)$$

These pseudo-domination relationships transform the cost vector of a link into a cost vector with 2 costs. The first cost is the delay of the link which is an additive float cost, while the second cost is the availability weight (or the b-consecutive slot availability) of the link which is a concave bounded integer. The upper bound of the integer second cost is the size of the link vector u. A cost vector with these 2 costs results in a polynomial-time problem as proven in [30] and also used in [34]. More specifically, for a given value of the integer cost, there can be only one non-pseudo-dominated path between a source–destination pair, the one with the smallest delay. Since the integer cost can take values at the most equal to the dimension u of the link vector, this is the upper limit on the number of non-pseudo-dominated paths per source–destination pair, which is polynomial in u and clearly do not depend on the network size. Moreover, if the hop count is included in the cost vector (and the pseudo-domination relationships above are appropriately modified), an upper bound on the number of computed paths per node is $u \cdot (N-1)$, where N is the number of network nodes (also an upper bound on the hop count of acyclic paths), which is again polynomial in u. Assuming the *general* case where the size of the problem is proportional to u (link utilization profiles are part of the problem and generally require $O(u)$ bits to record), this corresponds to a polynomial-time algorithm. If the link utilization profiles can be recorded in less than $o(u)$ bits (that is, if they are sparse and encodings that use less than $o(u)$ bits can be found), then the algorithm even though polynomial in u, may not be polynomial in the size of the problem.

6.2. Branch-and-bound: Discard paths by considering the optimization function

In this section we describe a way to reduce the search space of possible paths using a branch-and-bound version of the optimal algorithm. It is important to note that the optimal algorithm of Section 5, as well as the polynomial-time heuristics of Section 6.1, do not assume a specific optimization function for choosing the optimal path, until the very end; any optimization function (which, importantly, can be different for different connections, depending on their QoS requirements, or other factors) can be used in the second phase of the algorithm, after the set of non-dominated (or non-pseudo-dominated) paths have been found. In contrast, the branch-and-bound mechanism described in the current section assumes that the optimization function to be applied is known a priori, and improves the execution time and complexity of the algorithm by using the current best value of the optimization function to kill paths that, if expanded, would lead to worse cost. For the sake of being specific we will assume that the optimization function is the one minimizing the data reception time at the destination.

We modify the algorithm of Section 5.1 for computing the set of non-dominated paths $P_{\text{n-d}}$. We start by bounding the data reception time at the destination by computing the reception time over the Dijkstra's shortest path $p_{S,E}$ from source node S to destination node E, which is given by $B^* = d_{p_{S,E}} + R_{p_{S,E}}(b) + b$ (as in step IIa of Section 5.2).

We can now discard a path p_i that ends at node n_i (even if $n_i \neq E$), if by extending path p_i using the Dijkstra's shortest path from n_i to E (with delay $d_{n_i,E}$) and assuming this path is also free of reservations ("best-case"), the earliest reception time is larger than B^*. Formally:

$$\text{Discard } p_i \text{ if } d_{p_i} + d_{n_i,E} + R_{p_i}(b) + b \geqslant B^*$$

If we find a path p_i ending at destination node $E$with reception time smaller than the current value of B^*, we replace B^* with this new value. Since the optimization criterion we use is the one minimizing the data reception time at the destination, when the algorithm finishes, the optimum path will be the one that caused the last update of B^*.

By initially bounding the total delay by $B^* = d_{p_{S,E}} + R_{p_{S,E}}(b) + b$, the paths that we examine are those with $R(b)$ smaller than $R_{p_{S,E}}(b)$, since their delays are at least as large as the shortest path delay. Since $R(b)$ is integer and less than u, the number of non-dominated paths between the given source–destination pair can be at the most u, which does not depend on the network size. Therefore, with the branch-and-bound extension we have a polynomial-time algorithm, for this specific optimization function. Note that for the specific optimization function we do not lose optimality, as might happen with the heuristic variations of Section 6.1. We lose, however, in generality and flexibility, since the optimal multicost algorithm of Section 5, and the polynomial-time heuristics of Section 6.1, can be used with any kind and number of costs and any optimization function, which can be different for different connections and can also change with time. The proposed branch-and-bound mechanism can only be used for these two specific

costs, delay and link utilization vector, and the specific optimization function for which we can compute a "best-case" reception time using linear relations between the costs of the links. Finally, note that the branch-and-bound mechanism, can be also incorporated in the heuristic variations described in Section 6.1, assuming that the optimization function to be used is given a priori.

7. Performance evaluation results

In order to evaluate the performance of the proposed optimal multicost routing and time scheduling algorithm, the polynomial-time heuristic variations and the branch-and-bound polynomial-time algorithm, we have conducted full network simulation experiments. We did not evaluate the case where the branch-and-bound mechanism is incorporated in the heuristic versions of the multicost algorithm, even though it is expected that further time improvements would be obtained in that case, since we wanted to examine the effect of each mechanism separately.

The experiments were performed assuming an Optical Burst Switched network. More specifically, we assume that bursts arrive at each edge node according to a Poisson process with rate λ requests per second and their destinations are uniformly distributed over all remaining nodes of the network. The burst sizes are exponentially distributed with mean $\bar{l}$ bits, corresponding to mean burst duration equal to $\bar{B} = \bar{l}/C$. Bursts correspond to connection requests with unspecified starting times that have to be routed and scheduled by the proposed algorithms. We have used a Boolean vector of size u to implement the utilization profiles of the links, as presented in Section 3. Each element of the vector corresponds to a time step of duration τ and takes the zero value if the link is reserved and 1 if the link is available at the corresponding time slot. As time elapses, we shift the vector accordingly, so that its entries to correspond to future time intervals as related to the current time. Initially, the cost vectors of all links contain only 1's, since the links are fully available at the beginning. Update information is communicated to candidate nodes according to Section 5.3, and the delay of the update information is taken into account in the simulation experiments. Upon receiving an update message, a node calculates the entries that correspond to the specific reservation and updates the utilization profile of the specific link by putting 0's to the corresponding entries.

We have extended the ns-2 platform [38] and tested the following routing algorithms:

(i) *The Dijkstra shortest path algorithm* (*Dij*), where bursts are routed on the shortest propagation delay path computed at the beginning of the simulation. The algorithm takes into account reservations made at the first hop to avoid contention at the source (ingress) node, but does not take into account reservations made at subsequent nodes.

(ii) *The optimal multicost algorithm* (*OM*) described in Section 6 that chooses the path that minimizes the burst reception time at the destination over all non-dominated paths. The cost vector of the algorithm contains two cost parameters: (a) the link delay d_l, (b) the binary capacity availability vector $\widehat{C}_l$, which has $u = D/\tau$ entries. The domination relation $>$ of Eq. (3) is used for calculating the non-dominated paths. When the optimal multicost algorithm uses the branch-and-bound mechanism (Section 6.2) we identify it by using the notation OM-BB. Note that OM and OM-BB algorithms return the same optimum path solutions. OM and OM-BB algorithms differ only in the number of computed paths and thus the number of operations performed.

(iii) *The availability weighting heuristic multicost algorithm* (*AWHM*), where the pseudo-domination relation $>_{ps}$ of Eq. (6) is used for calculating the set of non-pseudo-dominated paths based on the weights of their CAVs. The branch-and-bound extension was not evaluated.

(iv) *The b-consecutive slot availability heuristic multicost algorithm* (*CSAHM*), where the pseudo-domination relation of Eq. (7) is used for calculating the set of candidate paths. The branch-and-bound extension was not evaluated.

(v) *The Dijkstra shortest path algorithm with contention avoidance* (*Dij/CA*), where the shortest paths are computed at the beginning of the simulation for every source–destination pair. Upon a burst transmission request the source/ingress node combines (using the $\oplus$ operator) the utilization profiles of the links on the shortest path and schedules the start time for the transmission of the burst so as to avoid contention at subsequent nodes.

The optimal multicost (OM) algorithm, and its AWHM and CSAHM heuristic variations first compute the set of non-dominated paths or pseudo-non-dominated paths, respectively, from the source to the given destination, and then choose the path that minimizes the reception time of the burst at the destination. The Dijkstra (Dij) and the Dijkstra/CA (Dij/CA) algorithms always select the shortest path to transmit a burst, and the difference between these two algorithms lies in the computation of the time offset (*TO*) after which the burst starts transmission: the Dij algorithm computes the *TO* by using reservation information only for the first link on the shortest path, while the Dij/CA algorithm takes into account reservations made at all the links on the shortest path.

The routing and time scheduling algorithms presented compute the path to be followed and the time offset after which the source should start transmitting the burst. In order to set up the path and reserve the appropriate resources we use a one-way reservation scheme that employs timed reservations and retransmissions in the OBS domain, similar to the scheme proposed in [40]. We assumed that dropped bursts are retransmitted because (a) this is what would happen in a real network, where the loss of data cannot be afforded (without retrials in the OBS network layer, a higher layer protocol, such as TCP, would be commissioned with the retransmission of the dropped data) and (b) this is a more reliable model for evaluating the performance of the proposed routing algorithms (if retrials were not incorporated, the network

would tend to drop bursts that travel more hops, treating such bursts unfairly, and the throughput results would not be representative of actual performance). The ingress node stores the burst in a limited-size input buffer until it is successfully transmitted. In our simulations the size of the ingress buffer was set equal to 256 Mbytes per node. For the traffic loads simulated we never observed a case of a burst being dropped due to ingress buffer overflow, and all bursts managed to reach their destinations either in their first attempt or after several retrials.

We experimented with two network topologies: a 5×5 mesh with wraparounds (Fig. 5a) and the NSF network topology (Fig. 5b). In the mesh topology, the nodes were arranged along a two-dimensional grid topology, with neighboring nodes placed at a distance of 50 km, 100 km or 200 km from each other. In the NSFnet topology, in addition to the actual link physical lengths (the link lengths depicted in Fig. 5b, adopted from [39]), we have also experimented with link lengths that are a fraction (10%, 30%, or 60%) of these lengths. All links were assumed to be bi-directional with propagation delays proportional to their lengths (5 μs/km). We have assumed that each link has a single wavelength with bandwidth $C = 1$ Gb/s. The setup (BCH) packet processing delay was set to 0.02 μs. Unless otherwise stated, we have used the *update-upon-reservation* technique to convey reservation information.

We used the average end-to-end delay experienced by a burst as the main metric for assessing algorithmic performance. Additional performance metrics measured in our simulations were the average number of retrials required for a successful transmission, the average number of computed paths per request, and the average number of operations for serving each request The average number of operations is defined as the number of bitwise comparisons, additions, and AND operations required for computing the binary CAV cost components, plus the number of integer operations required for computing the integer-valued delay and hop count cost components.

7.1. Results for the 5×5 mesh with wraparounds network

For the experiments in this section the time step τ was set equal to 0.01 ms and the size u of the capacity availability vectors was equal to 8000 (corresponding to a maximum delay $D = u \cdot \tau = 80$ ms). The distance between adjacent nodes of the wraparound mesh was chosen to be 50 km (corresponding to a single link propagation delay of 0.25 ms=$25 \cdot \tau$, and an average end-to-end propagation delay measured to be 1.3 ms). The mean burst size $\overline{I}$ was set equal to 300 kB, corresponding to mean burst duration $\overline{B} = 2.4$ ms $= 240 \cdot \tau$, while the burst arrival rate λ varied between 5 bursts/s and 125 bursts/s per source node.

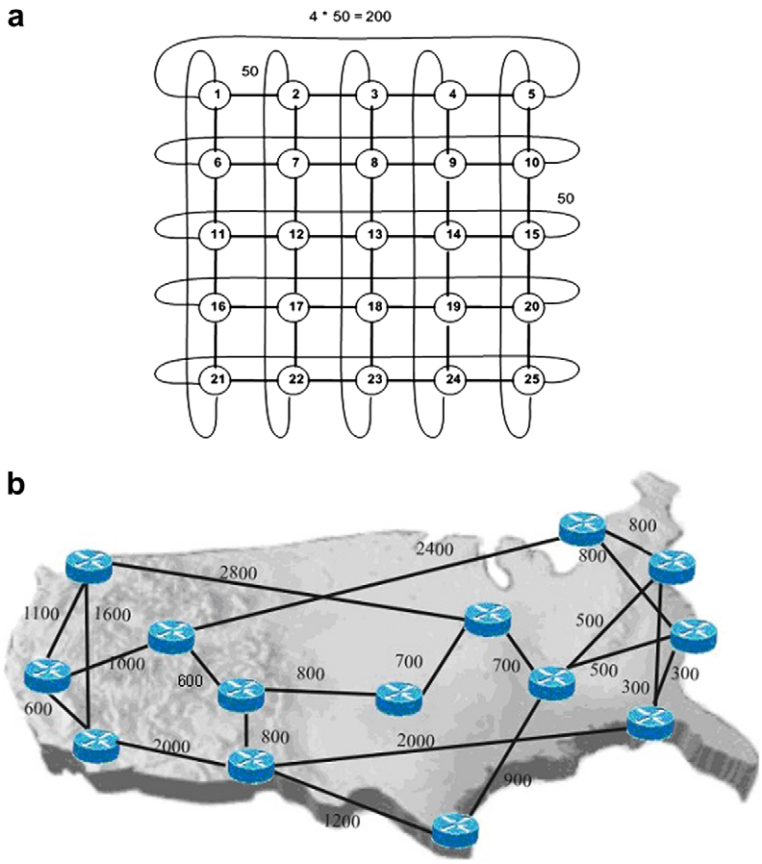


Fig. 5. (a) 5×5 mesh with wraparounds network and (b) the 14 nodes NSF network (link distances in km – distances correspond to actual network size).

Fig. 6a illustrates the average end-to-end delay experienced by the bursts using each of the algorithms examined. We observe that the optimal multicost (OM) algorithm outperforms significantly the Dijkstra (Dij) and the Dijk-

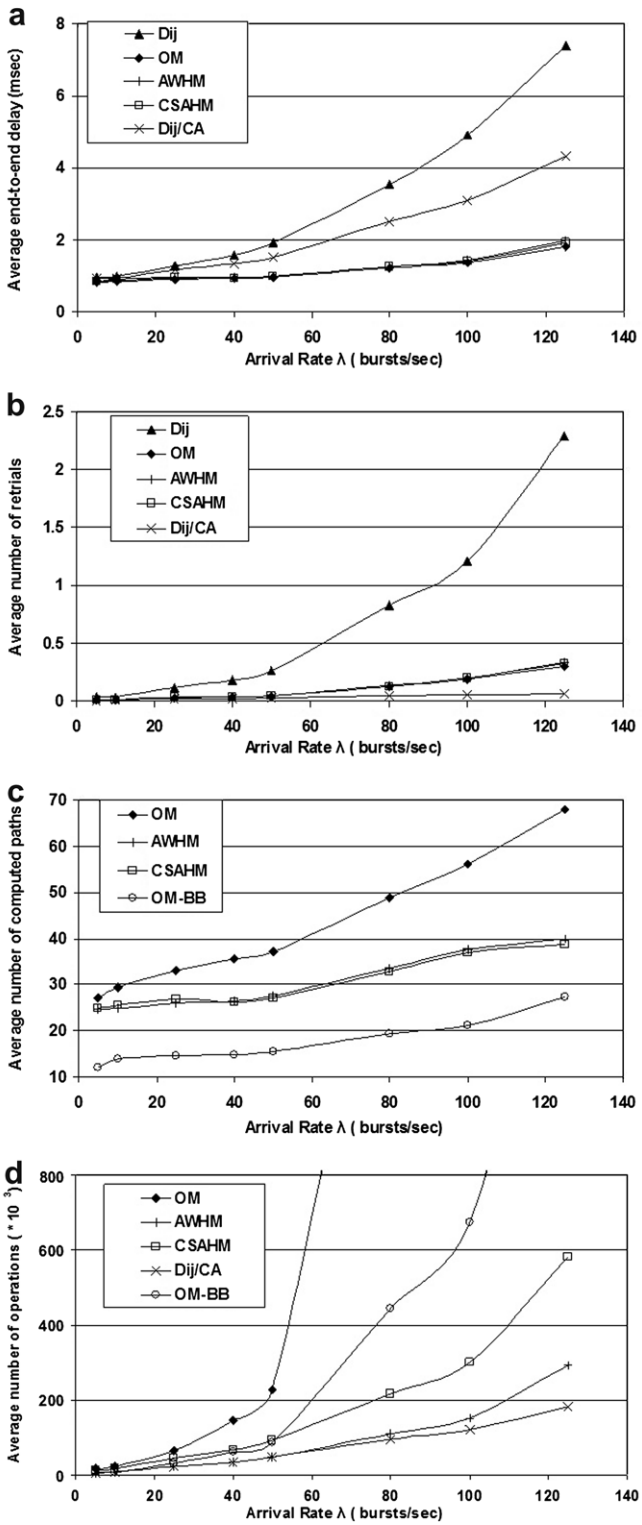


Fig. 6. (a) Average end-to-end delay per burst, (b) average number of retrials for a successful transmission, (c) average number of computed paths, and (d) average number of operations, for mean burst size $\bar{l} = 300$ kB and various burst arrival rates λ.

stra/CA (Dij/CA) algorithms, while the performance difference between the OM algorithm and the AWHM and CSAHM heuristic algorithms is very small.

Fig. 6b shows that the Dij/CA algorithm requires the smallest average number of retrials for a successful transmission. The OM, AWHM and CSAHM algorithms have slightly worse performance. A retrial is required when multiple bursts simultaneously try to reserve capacity on the same link, in which case only one succeeds, while the other(s) fail and have to retry. The Dij/CA algorithm is less affected by these issues, since for every source–destination pair a single path (the shortest path) is tried, and the algorithm avoids contentions over that path. The simple Dij algorithm has the worst performance in terms of the number of retrials, since it only considers link utilization information for the first hop.

Fig. 6c illustrates the average number of computed paths per successful transmission. Note that for this metric we also consider the retrials performed until we have a successful transmission. We have excluded from this graph the performance of the Dij and the Dij/CA algorithms, since in these algorithms path computation is only performed once. However, we have graphed the performance of the optimal multicost algorithm with the branch-and-bound mechanism (OM-BB). As explained previously, there is no difference between the OM and OM-BB algorithms in the metrics presented in Fig. 6a and b, because both OM and OM-BB algorithms select the same optimal paths (Section 6.2). From Fig. 6c we can verify that the branch-and-bound mechanism reduces the search space and thus results in a smaller number of computed paths, for the given optimization function. As expected, by comparing OM and AWHM and CSAHM algorithms we can observe that the use of the pseudo-domination relation for pruning candidate paths reduces the number of computed paths, as discussed in Section 6.1.

Regarding the average number of operations, illustrated in Fig. 6d, the OM algorithm exhibits the worst performance, as expected. The number of operations is reduced by using the branch-and-bound mechanism (OM-BB) due to the smaller number of searched paths (Fig. 6c). However, the improvement is not significant due to the tedious comparisons (domination relation) between the link cost vectors, which are not avoided in the OM-BB. Thus, although the OM-BB algorithm computes fewer paths than the heuristic algorithms (Fig. 6c), it requires a higher number of operations (Fig. 6d). The number of operations is maintained at low levels when the AWHM and CSAHM algorithms are used, due to the use of the pseudo-domination relations that not only reduces the search space but also avoids the comparisons of the CAV vectors. Note that the number of operations required by the CSAHM algorithm is larger than that of the AWHM algorithm, since obtaining the b-consecutive slot availability value requires one complete scan of the CAV vector. The number of operations of the AWHM algorithm is similar to that of the Dij/CA algorithm for small values of the burst arrival rate λ, and becomes worse than that as λ increases.

We have also measured the performance of the algorithms when the burst arrival rate λ is kept constant and the mean burst size $\bar{l}$ varies. The conclusions are similar to those presented above. We observed that the performance depends more strongly on the mean burst size $\bar{l}$ than on λ. This is because when the burst sizes increase the capacity fragmentation problems become more important.

For the rest of the this paper we will focus only on the performance of the AWHM algorithm since the delay performance of the CSAHM algorithm was always very similar to that of the AWHM algorithm, while its complexity was always worse.

7.1.1. Effect of time discretization step τ

The time discretization process and the size u of the binary capacity availability vector have a significant effect on the performance and the complexity of the algorithms. Fig. 7a and b shows the average end-to-end delay and the average number of operations, respectively, for different values of the time discretization step τ, for burst arrival rate $\lambda = 80$ bursts/s and mean burst size $\bar{l} = 300$ kB. For fair comparison, we have kept constant the product $D = u \cdot \tau = 80$ ms, so as not to change the maximum allowable delay. From Fig. 7a we can observe that the time discretization process affects the average end-to-end delay of all the algorithms. Since a burst is released to the network at discrete time instances and the reservation is performed for an integer number of τ-time units, performance deteriorates as τ increases. On the other hand, the average number of operations is inversely proportional toτ (Fig. 7b), so the complexity of the algorithms improves as τ increases.

7.1.2. Effect of propagation delays

The network propagation delays play a significant role on the link-state update mechanism. In the proposed multicost algorithm and its heuristic variations the nodes keep track of the link states (the CAVs) and use them to compute the set of non-dominated or non-pseudo-dominated paths, respectively, while in the Dijkstra algorithm with Contention Avoidance (Dij/CA) the link states are used to compute the time offset after which a burst should start transmission at the source. Cleary, information regarding a link reservation cannot be useful if it reaches a node after path selection. Therefore, we expect network propagation delays to affect the performance of all algorithms examined, except for the Dijkstra algorithm that only uses information on the first hop.

Fig. 8a shows the average end-to-end delay percentage (%) improvement obtained by using the AWHM algorithm as opposed to the Dij/CA algorithm in a 5×5 mesh with wraparounds, with adjacent node distances of 50, 100 or 200 km. As expected, the performance improvements obtained are more important when network propagation delays are small, while they almost disappear for large propagation delays. This comes at the expense of increased complexity (Fig. 8b), since when the network propagation delays are small the state information maintained at ingress nodes includes information for a large number of reservations. Therefore, the processing overhead and the complexity of the proposed algorithms (except for the Dijkstra algorithm) increase as network propagation delays decrease.

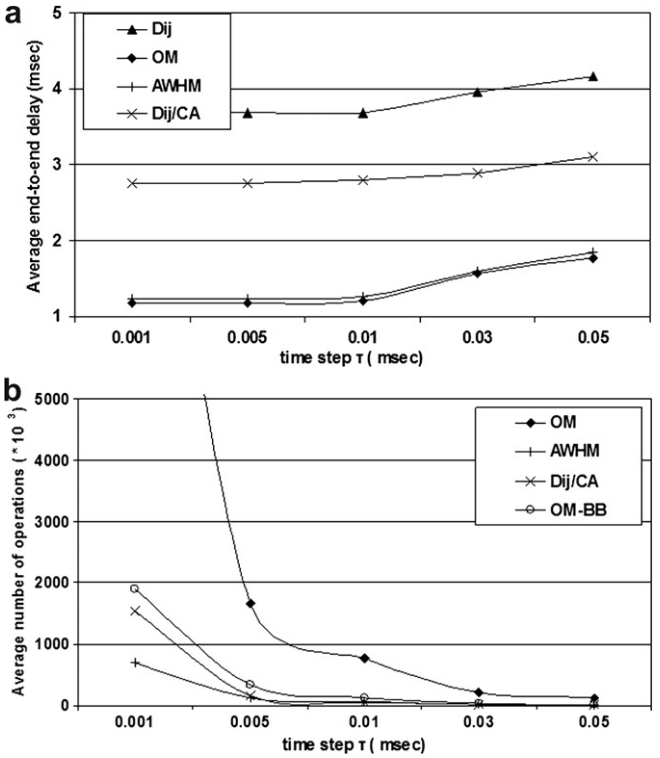


Fig. 7. (a) Average end-to-end delay and (b) average number of operations per burst, for $\lambda = 80$ burst/s, $\bar{I} = 300$ kB and various values for the time discretization step τ. The product of τ with u was kept constant.

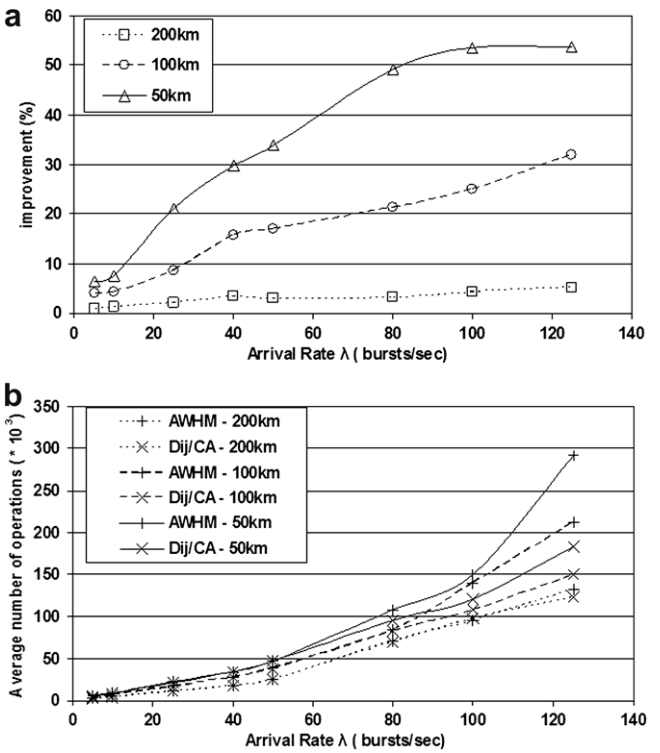


Fig. 8. (a) Average end-to-end delay improvement obtained by comparing the performance of the algorithm (iii) with algorithm (v), and (b) average number of operations per burst, for $\bar{I} = 300$ kB and various values for λ. The distance between adjacent nodes was set to 50 km, 100 km or 200 km.

7.2. NSF network

In this section we present performance results obtained for the NSFnet network topology. Fig. 9a shows the average end-to-end delay percentage improvement obtained by using the AWHM algorithm as opposed to the Dij/CA algorithm. We have assumed mean burst size $\bar{I} = 300$ kB and varying bust arrival rate λ and link propagation delays. More specifically, we have experimented with link lengths that are fractions (10%, 30% or 60%) of the original NSFnet distances (shown in Fig. 5b). These results indicate that the performance improvements obtained with the AWHM algorithm are substantial for the NSFnet when the link lengths are 10% and 30% of the actual lengths (corresponding to average network propagation delays of 1.3 ms and 3.9 ms, respectively) and diminish as the propagation delays increase. Similar to the 5×5 mesh network, when the propagation delays increase the complexity of the algorithms decreases (Fig. 9b), since the useful update information that reaches ingress routers is less.

7.3. Link-state update strategies

Fig. 10a shows the percentage improvement in the average end-to-end delay obtained by using the AWHM algorithm as opposed to the Dij/CA algorithm. Specifically, we have graphed the performance improvement for the cases that the AWHM algorithm uses the *update-upon-reservation* or the *update-upon-selection* technique (Section 5.3) in the 5×5 mesh with wraparounds network when the distance between adjacent nodes was set to 50, 100 or 200 km. We observe that the *update-upon-selection* is slightly better than the *update-upon-reservation* with respect to the average end to end delay metric (Fig. 10a), while it also results in considerably smaller average number of exchanged messages (Fig. 10b). When comparing these two techniques for different network propagation delays, we observe that the *update-upon-selection* technique outperforms the *update-upon-reservation* technique for large propagation delays, while the performance difference between these two techniques diminishes for small propagation delays. As expected, the average number of update messages exchanged decreases when the network propagation delays increase, since update messages are sent only to nodes that can use this information (see Eqs. (4) and (5)). Note that the average number of update messages exchanged when the *updated-upon-selection* technique is used is always less than 24 messages (namely, the number of nodes in the network, excluding the source), while in the case of the *update-upon-reservation* technique the upper bound is $24 \cdot h_{\max}$, where $h_{\max}$ is the maximum number of hops on the paths, since the update is performed at every intermediate node down the selected path. Similar conclusions were obtained for the case of the NSFnet topology (the results are omited for brevity).

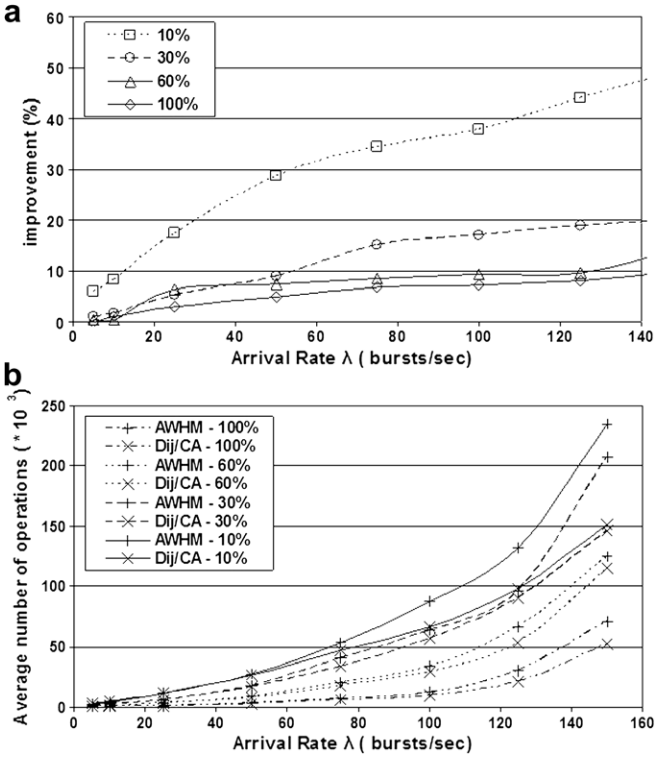


Fig. 9. (a) Average end-to-end delay improvement obtained by comparing the performance of the AWHM algorithm with Dij/CA algorithm, and (b) average number of operations per burst, for $\bar{I} = 300$ kB and various values of λ. The lengths of the links correspond to 10%, 30%, 60% or 100% of the actual distances of the NSF network.

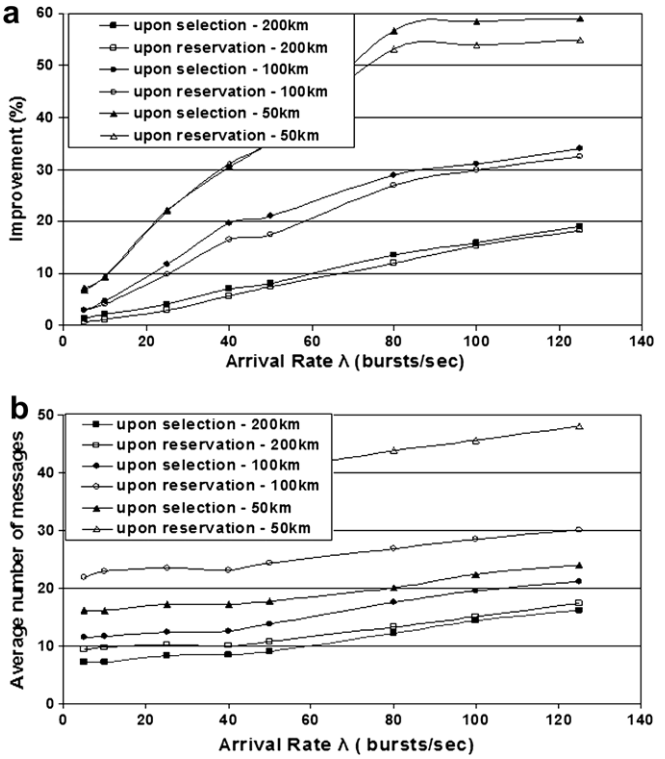


Fig. 10. (a) Average end-to-end delay improvement obtained by comparing AWHM algorithm using the *update-upon-reservation* or the *update-upon-selection* technique as opposed to Dij/CA algorithm, and (b) average number of update messages exchanged for the AWHM algorithm in the 5×5 Mesh topology, $\bar{I} = 300$ kB, and various values for burst arrival rates and propagation delays.

8. Conclusions

We presented several multicost algorithms for routing and time scheduling connections in networks that support advance reservations. We initially presented an optimal scheme of non-polynomial complexity that is based on the concept of non-dominated paths. We then proposed two heuristic algorithms of polynomial complexity, by defining corresponding appropriate pseudo-domination relationships that are used to prune the search space. We also proposed a branch-and-bound mechanism that reduces the search space, for the case where the function to be optimized is known and is the same for all connections. The performance of the proposed multicost algorithms was evaluated in an Optical Burst Switched network environment. Our simulation results showed that the proposed multicost algorithms significantly outperform other algorithms, such as the Dijkstra and the Dijkstra/CA algorithm with respect to the average end-to-end delay experienced by a burst. The optimal multicost algorithm is not polynomial and requires a large number of operations. The number of operations depends on the time discretization step τ, and can be decreased by increasing τ, at a relatively small penalty in terms of end-to-end delay. The proposed polynomial-time AW heuristic multicost algorithm yields delay performance that is very close to that of the optimal multicost algorithm, while maintaining the number of operations at low levels. The branch-and-bound mechanism was shown to reduce the number of searched paths, without sacrificing optimality, but is less flexible, since it assumes that the objective function to be optimized is the same for all connections. The performance improvements obtained by the proposed multicost algorithms are more pronounced for small network propagation delays, in which case more up-to-date link utilization information is used in making routing and scheduling decisions.

Acknowledgements

This work has been supported by the European Commission through the IST PHOSPHORUS project (www.ist-phosphorus.eu). K. Christodoulopoulos was supported by GSRT through PENED project 03EΔ207, funded 75% by the European Commission and 25% by the Greek State and the private sector.

Appendix A

In this Appendix we provide pseudo-code for the algorithm that finds the set of non-dominated paths between an origin node and all other nodes of the network.

A.1. Notation

The network is defined as a directed graph $G = (N,L)$: N are the nodes, and L are the links.

The connection request is of size b in slots and has source node $S \in N$ and destination node $E \in N$
V_l is the cost vector of link $l \in L$. Each path is represented by a label that includes the cost vector V_p associated with it and the first hop to the source using that path.
W_i is the set of labels of the paths from node S to a node $n_i \in N$, and $W = \cup_{n_i \neq S} W_i$ is the set of all labels.
$W^f \subseteq W$ is the subset of all final labels for all the nodes, and $W_i^f \subseteq W_i$ is the set of final labels for node n_i. T is the set of nodes with at least one final label.

Compute_the_Set_of_Non_Dominated_Paths (G, S, E, V_l of all link)
  Initialization (G, S,V_l)
  while $W \neq W^f$
    n_i = Choose_the_optimum_label (W, W^f, T)
    Obtain_the_new_labels_and_discard_dominated_paths (G, n_i, W,V_l)
  end
  return ($P_{n-d} \equiv W_E^f$)
end

Initialization (G, S, V_l of all links)
  $W^f = \{\}$, $T = \{\}$
  for all links $l \in L$ that start from $S \in N$
    $W = W \cup V_l$
  end
end

Choose_the_optimum_label (W, W^f, T)
  find path $p_i \in W$ with minimum delay (delay information is included in V_{p_i})
  $n_i \in N$ = ending node of path p_i
  $W^f = W^f \cup \{V_{p_i}\}, T = T \cup \{n_i\}$
  return (n_i)
end

Obtain_the_new_labels_and_discard_dominated_paths(G, n_i, W, V_l)
1: for all $n_j \in N$ neighbors of n_i (connected through link $l = (n_i,n_j) \in L$)
    $V'_{p_j} = V_{p_i} \odot V_l$ ($\odot$ is defined in Eq. (2))
    for all $V_{p_j} \in W$ (V_{p_j} are paths ending at n_j)
      if $V_{p_j} > V'_{p_j}$ (">" is defined in Eq. (3))
        goto 1 (check the next neighboring node)
      else if $V'_{p_j} > V_{p_j}$
        $W = W - \{V_{p_j}\}$
      end
    end
    $W = W \cup \{V_{p_j}\}$
  end
end

Appendix B

In this Appendix we give pseudo-code for the algorithm that selects the best path from the set of non-dominated paths.

Choose_the_optimal_path (b,$P_{\text{n-d}}$)
  min_arrival=∞
  for all $p \in P_{\text{n-d}}$
    calculate $R_p(b)$ (find the first placement that $\widehat{C}_p$ has b consecutive ones)
    temp_arrival=$R_p(b) + b + d_p$,
    if temp_arrival <min_arrival
      temp_path=p
      temp_transmission=$R_p(b)$
      min_arrival=temp_arrival
    end
  end
  return (temp_path, temp_transmission)
end

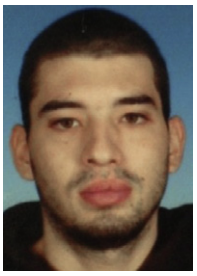
References

[1] S. Shenker, C. Partridge, R. Guerin, Specification of guaranteed quality of service, RFC 2212, September 1997.
[2] R. Braden, L. Zhang, S. Berson, S. Herzog, S. Jamin, Resource reservation protocol (RSVP), RFC 2205, September 1997.
[3] K. Nahrstedt, R. Steinmetz, Resource management in networked multimedia systems, Computer 28 (1995) 52–63.
[4] M. Yoo, C. Qiao, A high speed protocol for bursty traffic in optical networks, SPIE's All-Optical Communication Systems: Architecture, Control and Protocol Issues 3230 (1997) 79–90. Nov.
[5] E.A. Varvarigos, V. Sharma, An efficient reservation connection control protocol for gigabit networks, Computer Networks and ISDN Systems 30 (12) (1998) 1135–1156.
[6] J. Turner, Terabit burst switching, Journal of High Speed Networks 8 (1999) 3–16.
[7] C. Qiao, M. Yoo, Optical burst switching (OBS)—a new paradigm for an optical Internet, Journal of High Speed Networks 8 (1999) 69–84.
[8] I. Foster, C. Kesselman, The Grid 2: Blueprint for a New Computing Infrastructure, Morgan Kaufmann, 2003.
[9] I. Foster, C. Kesselman, C. Lee, R. Lindell, K. Nahrstedt, A. Roy, A distributed resource management architecture that supports advance reservations and co-allocation, in: International Workshop on Quality of Service, 1999.
[10] J. Zheng, H.T. Mouftah, Routing and wavelength assignment for advance reservation in wavelength-routed WDM optical networks, in: International Conference on Communications (ICC), 2002.
[11] E. He, X. Wang, V. Vishwanath, J. Leigh, AR-PIN/PDC: flexible advance reservation of intradomain and interdomain lightpaths, in: GLOBECOM 2006, July 2006.
[12] L.C. Wolf, L. Delgrossi, R. Steinmetz, S. Schaller, Issues of reserving resources in advance, Lecture Notes in Computer Science 1018 (1995).
[13] D. Ferrari, A. Gupta, G. Ventre, Distributed advance reservation of real-time connections, Multimedia Systems 5 (3) (1997).
[14] A. Schill, F. Breiter, S. Kuhn, Design and evaluation of an advance reservation protocol on top of RSVP, in: 4th IFIP International Conference Broadband Communications, 1998, pp. 23–40.
[15] O. Schelén, S. Pink, Sharing resources through advance reservation agents, Journal of High Speed Networks 7 (3–4) (1998). Special issue on Multimedia Networking.
[16] R. Guerin, A. Orda, Networks with advance reservations: the routing perspective, in: INFOCOM, 2000.
[17] E.A. Varvarigos, V. Sharma, The ready-to-go virtual circuit protocol: a loss-free protocol for multigigabit networks using FIFO buffers, Transactions on Networking 5 (October) (1997) 705–718.
[18] M. Dueser, P. Bayvel, Analysis of a dynamically wavelength-routed optical burst switched network architecture, Journal of Lightwave Technology 20 (April) (2002) 574–585.
[19] K. Christodoulopoulos, K. Vlachos, K. Yiannopoulos, E.A. Varvarigos, Relaxing delayed reservations: an approach for quality of service differentiation in optical burst switching networks, in: Broadnets, 2006.
[20] I. Baldine, G.N. Rouskas, H.G. Perros, D. Stevenson, JumpStart: a just-in-time signaling architecture for WDM burst-switched networks, Communications Magazine 40 (2) (2002) 2–89.
[21] J. Xu, C. Qiao, J. Li, G. Xu, Efficient channel scheduling algorithms in optical burst switched networks, in: INFOCOMM, vol. 3, 2003, pp. 2268–2278.

[22] M. Yoo, C. Qiao, S. Dixit, QoS performance of optical burst switching in IP-over-WDM networks, Journal on Selected Areas in Communication 18 (October) (2000) 2062–2071.
[23] T. Coutelen, H. Elbiaze, B. Jaumard, An efficient adaptive offset mechanism to reduce burst losses in OBS networks, in: GLOBECOM 2005.
[24] A. Zapata, P. Bayvel, Dynamic wavelength-routed optical burst-switched networks: scalability analysis and comparison with static wavelength-routed optical networks, in: OFC, 2003, pp. 212–213.
[25] R. Ali, K. Amin, G. Laszewski, O. Rana, D. Walker, M. Hategan, N. Zaluzec, Analysis and provision of QoS for distributed grid applications, Journal of Grid Computing (June) (2004) 163–182.
[26] W. Smith, I. Foster, V. Taylor, Scheduling with advanced reservations, in: IPDPS'00, 2000, pp. 127–132.
[27] J. Xing, C. Wu, M. Tao, L. Wu, H. Zhang, Flexible advance reservation for grid computing, in: GCC 2004, 2004, pp. 241–248.
[28] C. Castilo, G. Rouskas, H. Khaled, On the design of online scheduling algorithms for advance reservations and QoS in grids, in: International Parallel and Distributed Processing Symposium (IPDPS), 2007.
[29] F.J.P. Gutierrez, E.A. Varvarigos, S. Vassiliadis, Multicost routing in max–min fair networks, in: 38th Allerton Conference on Communicating, Control and Computing, October 2000.
[30] Z. Wang, J. Crowcroft, Quality-of-service routing for supporting multimedia applications, Journal on Selected Areas in Communications 14 (7) (1996) 1228–1234.
[31] G. Liu, K. Ramakrishnan, A*Prune: An algorithm for finding *K* shortest paths subject to multiple constraints, in: INFOCOM, vol. 2, 2001, pp. 743–749.
[32] D.S. Reeves, H.F. Salama, A distributed algorithm for delay constrained unicast routing, Transactions on Networking 8 (April) (2000) 239–250.
[33] A. Juttner, B. Szviatovszki, I. Mecs, Z. Rajko, Lagrange relaxation based method for the QoS routing problem, in: INFOCOM, vol. 2, April 2001, pp. 859–868.
[34] S. Chen, K. Nahrstedt, On finding multi-constrained paths, in: ICC 98, vol. 2, 1998, pp. 874–879.
[35] F.A. Kuipers, T. Korkmaz, M. Krunz, P. Van Mieghem, An overview of constraint-based path selection algorithms for QoS routing, Communication Magazine 40 (December) (2002) 50–55.
[36] X. Yuan, Heuristic algorithms for multiconstrained quality-of-service routing, Transactions on Networking 10 (April) (2002) 244–256.
[37] P. Van Mieghem, F. Kuipers, Concepts of exact QoS routing algorithms, Transactions on Networking 12 (5) (2004).
[38] The Network Simulator – ns-2: <http://www.isi.udu/nsnam/ns>.
[39] NSF network: <http://moat.nlanr.net/INFRA/NSFNET.html>.
[40] A. Maach, G.V. Bochmann, H. Mouftah, Robust optical burst switching, Networks (2004) 447–452.

Emmanouel (Manos) Varvarigos received a Diploma in Electrical and Computer Engineering from the National Technical University of Athens in 1988, and the M.S. and Ph.D. degrees in Electrical Engineering and Computer Science from the Massachusetts Institute of Technology in 1990 and 1992, respectively. He has held faculty positions at the University of California, Santa Barbara (1992–1998, as an Assistant and later an Associate Professor) and Delft University of Technology, the Netherlands (1998–2000, as an Associate Professor). In 2000 he became a Professor at the department of Computer Engineering and Informatics at the University of Patras, Greece, where he heads the Communication Networks Lab. He is also the Director of the Network Technologies Sector (NTS) at the Research Academic Computer Technology Institute (RA-CTI). His research activities are in the areas of high-speed networks, protocols, network architectures, distributed computation and grid computing.

Vasileios Sourlas received the diploma degree from the Computer Engineering and Informatics Department, University of Patras, Greece, in 2004 and the M.Sc. degree in Computer Science and Engineering of the Computer Engineering and Informatics Department in 2006. He is currently working toward the Ph.D. degree at the Department of Computer and Communication Engineering of University of Thessaly. His current research interests include routing and scheduling protocols in optical networks.



Konstantinos Christodoulopoulos received the Diploma of Electrical and Computer Engineering from the National Technical University of Athens, Greece, in 2002 and the M.Sc. degree in Advanced Computing from Imperial College London, UK, in 2004. He is currently working toward the Ph.D. degree at the Computer Engineering and Informatics Department of the University of Patras, Greece. His research interests are in the areas of protocols and algorithms for optical networks and grid computing.