Routing and Scheduling Connections in Networks that Support Advance Reservations

Emmanouel (Manos) Varvarigos[1], Vasileios Sourlas[2] and Konstantinos Christodoulopoulos[1]
[1] Department of Computer Engineering and Informatics, University of Patras, Greece, and
Research Academic Computer Technology Institute, Patra, Greece
{*manos,kchristodou*}*@ceid.upatras.gr*
[2] Department of Computer and Communication Engineering, University of Thessaly, Greece
vsourlas@inf.uth.gr

Abstract— **A key problem in networks that support advance reservations is the routing and time scheduling of connections with flexible starting time. 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 so as to minimize the reception time at the destination, or some other QoS requirement. 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. By appropriately pruning the set of candidate paths using path pseudo-domination relationships, we also 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 it is very close to that of the optimal algorithm.**

I. Introduction

Resource reservation is a way to provide Quality of Service (QoS) to end users. For example the RSVP protocol [1], used for Integrated Services (IntServ), is based on resource reservations as a way to meet specific QoS demands. 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 for the resource usage to be in the future. Thus, advance reservation requests contain time- as well as resource-related parameters. Of particular interest to us are the advance reservation requests that have unspecified (flexible) starting time and specified or unspecified duration (UTSD or UTUD respectively). Traditional [2], Multimedia [3], Optical Burst Switched [4] and Grid [5] networks are a few paradigms in which advance reservations have been examined.

We propose a multicost algorithm for routing and scheduling connections with flexible starting time that fall in the category of UTSD and UTUD advance reservation requests. The algorithm is based on the time discretization of the capacity utilization profiles of the links, a data structure introduced in [6] that can be used to keep track of the capacity reservations. The proposed multicost algorithm selects the path to be followed by a connection and the time instant the connection should start so as to minimize the data reception delay, or some other performance criterion, which may depend on the QoS requirements of the user. The algorithm makes its decisions based on network utilization information available at the source at the time of its execution and thus falls under the 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 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 connection 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 here are NP-hard. To obtain polynomial-time algorithms, we define pseudo-domination relationships between paths. Although the set of non-pseudo-dominated paths that we obtain is not guaranteed to contain always the optimum path, our results indicate that if the pseudo-domination relationship is chosen wisely we can obtain performance that is very close to that of the optimal algorithm.

We use simulations to evaluate the performance of the multicost routing and scheduling algorithms 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 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 verified that the performance improvements are more significant when the network propagation delays are small, a typical characteristic of feedback-based algorithms.

Although the algorithm is evaluated in an Optical Burst Switched network this doesn't limit its applicability. We also give examples (section III) of the application of our algorithm in a WDM environment that employs no or full wavelength conversion capabilities.

The remainder of the paper is organized as follows. In Section II we report on previous work. In Section III we present several useful formats for recording the utilization profile of a link. In Section IV we describe the problem under consideration. In Section V we present the multicost routing framework and in Section VI the polynomial algorithms. Section VII presents the performance results. Our conclusions follow in Section VIII.

II. 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 re-introduced issues in advance reservations.

In [7] the authors describe a model for resource reservations in advance and discuss issues that must be resolved in such context. The requirements of the users in a distributed advance reservation environment are discussed in [8]. The proposed design is implemented in the Tenet 2 Protocol suite. An Efficient Reservation Virtual Circuit (ERVC) protocol for high speed networks that uses advance and timed reservations is proposed in [6]. In [9] the authors discuss how to provide advance reservations on top of RSVP.

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 [2] first introduced the concept of advance reservation scheduling and advance-reservation aware routing algorithms. More specifically, [2] proposed several algorithms for advance reservations when the starting times are specified or are flexible, and also discuss the computational complexity of these algorithms. The topic of this paper is similar to [2] but we additionaly examine the network performance and also evaluate our algorithms through extensive full network simulation experiments. Finally, advance reservations as part of a RWA problem in WDM networks is examined in [10].

In OBS networks [4] and [11], an ingress node assembles data destined for the same node and having the same QoS requirements (thus, belonging to the same Forwarding Equivalent Class) into bursts. A burst transmission 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 (usually referred to as the Time Offset - TO) is calculated by taking into account the protocol used (one- or two-way) and the number of hops.

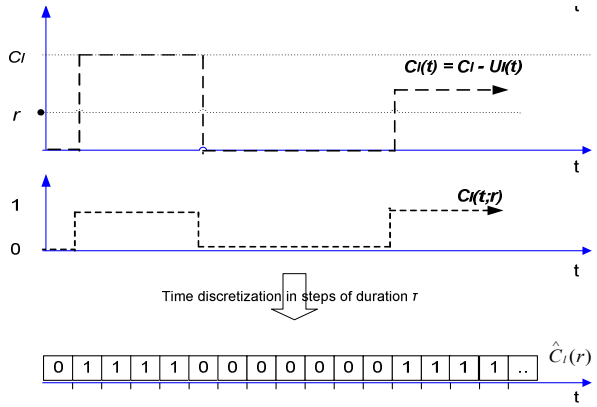

Fig. 1: The capacity availability profile $C_l(t)$, the r-capacity availability profile $C_l(t;r)$, and the binary r-capacity availability vector $\hat{C}_l(r)$ of a given link l of capacity C_l when the rate requested by a connection is r and the discretization step is τ.

The TO can also be chosen so as to provide QoS differentiation, as in [12], where a high loss priority class is given a larger extra offset time in order to make earlier reservation than lower priority classes. The choice of offsets is further examined in [13], where an adaptive scheme is proposed. OBS signaling protocols can be categorized into two main classes: two-way (tell-and-wait) and one-way (tell-and-go) protocols. A typical example of tell-and-wait protocols is the ERVC [6]. On the other hand, RGVC [14], Horizon [4] and Just Enough Time (JET) [11] belong to the tell-and-go class.

Grids [15] introduce new ways to share resources across geographically separated sites by establishing a global resource management architecture. 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) [5] is a framework for advance reservations that treats in a uniform way various types of resources. Algorithms that support advance reservations are discussed in [16] and [17].

We must note here that multiconstrained algorithms used in other works, solve problems in subsets of the solution space considered by multicost algorithms [18]. In the related literature, multiconstrained algorithms have mainly been used for QoS routing problems. In [19] the authors proved that QoS routing with QoS parameters being the bandwidth and the delay is not NP-complete. The general Multiconstrained Path Problem (MCP) is discussed in [20]. 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 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.

III. 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 [6]. 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 $C_l(t)\geq r$, and $C_l(t;r)=0$, otherwise. 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* $\hat{C}_l(r)$, abbreviated CAV, as the vector whose k-th entry is:

$$\left\{\hat{C}_l(r)\right\}_k = \left\{\begin{matrix}1\text{ , if } C_l(t;r)=1, \text{ for all } (k\text{-}1)\tau \le t \le k\tau \\ 0\text{ , otherwise.}\end{matrix}\right\},\ k=1,...,u.$$

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. 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 $\hat{C}_l(t;r)$ 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_w, the capacity of a link l is $C_l=w\cdot C_w$. A connection that wants to reserve k wavelengths, $1\leq k \leq w$, requests rate $r = k\cdot C_w$. If we can find a path of available capacity at least r then the connection can be established (full wavelength conversion). Therefore, $C_l(t;r)$ and $\hat{C}_l(r)$ can be used in this kind of network. If no wavelength converters are available, each link needs to keep track of the utilization of each of its w wavelengths separately. Thus a node has to maintain w binary (since each wavelength can be reserved or not for a given time) utilization profiles for each outgoing link. In this case, the network can be viewed as w "parallel" networks, each having a single wavelength. The case of Optical Burst Switched networks falls in one of the two aforementioned categories.

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

IV. The Routing and Scheduling Problem Under Consideration

The routing and 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 (ingress) node S that serves a connection that requests a certain amount of bandwidth r, for a given duration B, with specific destination (egress) node E. If the duration is not known in advance we have $B=\infty$. We are also given the capacity availability profiles $C_l(t)$ of all links l. We assume that there is an upper bound D on the maximum delay that the connection can tolerate; if this deadline cannot be met, the request should be rejected. Even when there is no limit D, we 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. 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 reception time of the data at its destination. Fig. 2 presents an instance of the problem.

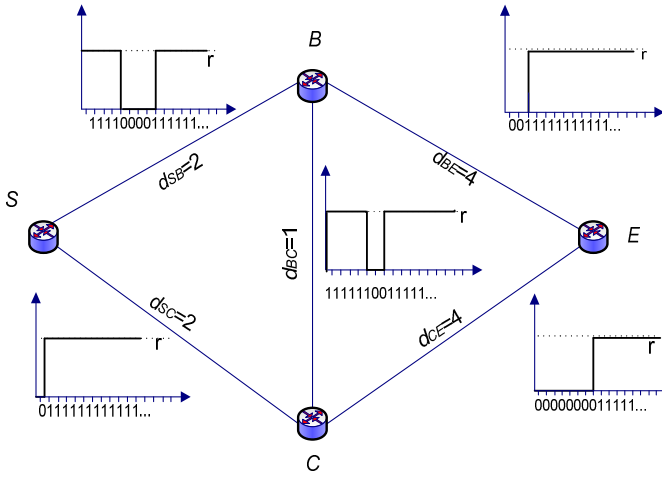

Fig. 2: A connection request for capacity r of duration B with destination node E arrives at node S. Each link is characterized by its propagation delay (in τ-time units) and its binary r-capacity availability vector.

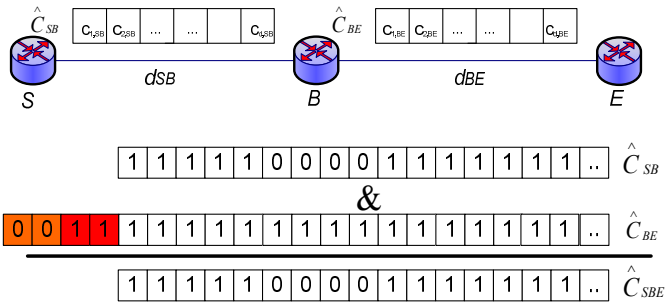

Fig. 3: Calculation of the path capacity availability vector $\hat{C}_{SBE}$. $\hat{C}_{BE}$ is shifted by $2\cdot d_{SB}=4$ τ-time units ($d_{SB}=2$ in this example), before the Boolean AND operation is applied.

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 algorithm in a network that has nonzero propagation delays. However, even when the feedback-based algorithm we propose uses 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.

A. Binary Capacity Availability Vector of a path

Assuming the routing decision is made at the source *S*, the binary capacity availability vectors of all network links should be gathered continuously at all nodes, over a network control plane that must exist. To calculate the capacity availability vector (CAV) of a path we have to combine the CAVs of the links that comprise it. In the example presented in Fig. 3, consider a path p_{SBE}, where *S* is the source and *E* is the destination node, and let $\hat{C}_{SB}$ and $\hat{C}_{BE}$ be the binary capacity availability vectors of links l_{SB} and l_{BE}, respectively. In order to compute the CAV of path p_{SBE}, we first gather the CAVs of the links that comprise it to the source node *S*. The CAV $\hat{C}_{SB}$ is maintained at node *S*, so only $\hat{C}_{BE}$ has to be transmitted from *B* to *S*. Upon its arrival at *S*, $\hat{C}_{BE}$ is left shifted by d_{SB} bits (propagation delay of link *SB* in τ-time units), denoted by $\mathrm{LSH}_{d_{SB}}$, to purge utilization information that corresponds to time period that has already expired to transfer the utilization information from *B* to *S*, obtaining in this way $\hat{C}_{BE}(S)$. We then left shift the resulting CAV by a further d_{SB} positions, to take into account the propagation delay from *S* to *B* (assuming the propagation delay of each link is the same in both directions). We finally execute a bit-wise Boolean AND operation, denoted by "**&**", among the CAVs in order to compute the binary utilization vector of the path p_{SBE}.

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:

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

Note that if *S* transmits data at the intervals indicated by 1's in $\hat{C}_{SBE}$, the data are guaranteed (if no reservations are performed in the meantime at the intermediate links) to find available capacity when they arrive at all intermediate links. This procedure can be extended so as to compute paths with more than two hops.

V. 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, 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 $\hat{C}_l$, that is,

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

but they may also include other parameters of interest (such as the number of hops, the capacity availability profile, etc). A cost vector [18] can then be defined for a path *p* consisting of links 1,2,…,*k,* based on the cost vectors of its links, as

$$V(p) = \bigodot_{l=1}^{k} V_l \overset{def}{=} \left(\sum_{l=1}^{k} d_l, \bigoplus_{l=1}^{k} \hat{C}_l \right), \quad (2)$$

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

$$p_1 \text{ dominates } p_2 \text{ (notation: } p_1 > p_2) \textbf{ iff}$$
$$\sum_{l \in p_1} d_l < \sum_{l \in p_2} d_l \text{ and } \bigoplus_{l \in p_1} \hat{C}_l \geq \bigoplus_{l \in p_2} \hat{C}_l, \quad (3)$$

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

The routing and scheduling algorithm we propose consists of two phases: given a source-destination pair (*S* and *E*, respectively), the set $P_{n\text{-}d}$ 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 $p \in P_{n\text{-}d}$ 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.

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

The algorithm that computes the non-dominated paths from a given source to all network nodes (including *E*) 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 Dijkstra case) since we can find more "non-dominated" paths to that node later.

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 $W = \bigcup_{n_i \neq S} W_i$ 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, and $n_i \neq S$.

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 (Section V.A). If V_{p_i} is the cost vector of the chosen label and n_i is the node to which it leads, then the following updates are performed:

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

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, $V_{p_j}' = V_{p_i} \odot V_l$ where V_l is the label of link l, 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 vector V_{p_j}' is not dominated by any of the vectors in W_j , then V_{p_j}' is added to the sets 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 V_{p_j}'. Note that it is not possible for the new vector to dominate an existing vector and be dominated by another 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 .

B. Choosing the optimal path 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 $P_{\text{n-d}}$ comes with a cost vector V, consisting of its propagation delay in τ-time units and its CAV. In the second phase of the algorithm we apply an optimization function $f(V(p))$ to the cost vector of each path $p \in P_{n\text{-}d}$ 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 a tell-and-wait) to be used for establishing the connection. In general, the optimization function f 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 (expressed by the CAV), 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 scheme, the following process is used to select the best path from the non-dominated paths that have been found:

Step I: Finding the earliest transmission time
For the capacity availability vector $\hat{C}_p$ of each non-dominated path p we calculate the first position $R_p(b)$ after which $\hat{C}_p$ 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: Minimum reception time algorithm
Select the path p that minimizes the reception time, defined as the time the last bit of the connection reaches the destination:

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

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

Step III: Updating the CAVs
Having chosen the path to schedule the connection, the next step is to update the CAVs of the chosen path by converting the ones in the appropriate positions of the CAV to zeros. This is done by the reservation protocol. Since ingress nodes need to use link state information to compute the non-dominated paths, link-state update packets should also be sent to the other nodes. Due to space limitation we won't describe such mechanisms in this paper. Approaches such [21], which extends SNMP to cope with time related link availability profiles, are applicable.

If we wanted to use a tell-and-wait protocol, instead, we simply have to redefine $R_p(b)$ as the first position after $2 \cdot d_p$ (the round trip delay of p) after which $\hat{C}_p$ has b consecutive ones.

VI. Polynomial Time Algorithms

A serious drawback of the algorithm described in Section V is that the number of non-dominated paths may be exponential, and the algorithm is not guaranteed to finish in polynomial time. To obtain a polynomial time algorithm we define a pseudo-domination relationship $>_{\text{ps}}$ between paths, which has weaker requirement than the domination relationship $>$ defined in Eq. (3). This pseudo-domination relationship can be used in step 3 of the multicost routing algorithm of Section V.A to prune (kill) paths, yielding a set $P_{n\text{-}ps\text{-}d} \subseteq P_{n\text{-}d}$ of non-pseudo-dominated paths that has considerably smaller (polynomial) cardinality than $P_{n\text{-}d}$. Then, the algorithm given in Section V.B can be applied to the paths in $P_{n\text{-}ps\text{-}d}$. 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 $>_{\text{ps}}$ is defined wisely.

We define two new metrics for a link l. The first metric is called the *slot availability weight* of the link, denoted by weight($\hat{C}_l$), which is the total number of 1's in the vector. The second metric is the *b-consecutive slot availability* of a link l, denoted by $L(b,\hat{C}_l)$, which is the total number of runs of consecutive 1's in $\hat{C}_l$ that have length equal to the connection duration b.

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

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

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

Based on these metrics, we present two polynomial-time heuristic variations of the optimal multicost algorithm that use two different pseudo-domination relationships to prune the set of candidate paths.

1) Availability Weighting algorithm

The pseudo-domination relationship used to prune paths is

$$p_1 \text{ pseudo-dominates } p_2 \ (p_1 >_{ps} p_2) \textbf{ iff}$$
$$\sum_{l \in p_1} d_l < \sum_{l \in p_2} d_l \text{ and } weight(\bigoplus_{l \in p_2} \hat{C}_l) \le weight(\bigoplus_{l \in p_1} \hat{C}_l) \quad (4)$$

2) b-consecutive slot availability algorithm

The pseudo-domination relationship used to prune paths is

$$p_1 \text{ pseudo-dominates } p_2 \ (p_1 >_{ps} p_2) \textbf{ iff}$$
$$\sum_{l \in p_1} d_l < \sum_{l \in p_2} d_l \text{ and } L(b, \bigoplus_{l \in p_2} \hat{C}_l) \le L(b, \bigoplus_{l \in p_1} \hat{C}_l) \quad (5)$$

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-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 [19]. 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. 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.

VII. Performance Evaluation Results

In order to evaluate the performance of the proposed optimal multicost routing and scheduling algorithm and its polynomial-time heuristic variations, we have conducted full network simulation experiments. The experiments were performed assuming an Optical Burst Switched network. More specifically, we assume that bursts arrive at each node according to a Poisson process of rate λ requests/sec and their destinations are uniformly distributed over all remaining nodes. The burst sizes are assumed to follow the exponential distribution with mean $\overline{I}$ bits, corresponding to mean duration $\overline{B} = \overline{I}/C$. Each burst corresponds to a connection request with unspecified starting time that has to be routed and scheduled by the proposed algorithms. We have extended ns-2 platform [22] and tested the following algorithms:

The Dijkstra shortest path algorithm. Bursts are routed on the shortest delay path. 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.

The optimal multicost algorithm described in Sections V that chooses the path that minimizes the burst reception time at the destination over all non-dominated paths. The domination relationship of Eq. (3) is used for calculating the non-dominated paths.

The Availability Weighting (AW) heuristic multicost algorithm, where the pseudo-domination relationship of Eq. (4) is used for calculating the non-pseudo-dominated paths.

The b-Consecutive Slot Availability (CSA) heuristic multicost algorithm, where the pseudo-domination relationahip of Eq. (5) is used.

The Dijkstra shortest path algorithm with Contention Avoidance (Dijkstra/CA). The shortest paths are computed at the beginning of the simulation. To schedule a burst, the source combines (using the $\oplus$ operator) the utilization profiles of the links on the shortest path to avoid contention at subsequent nodes.

The optimal multicost algorithm, and its AW and CSA 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 and the Dijstra/CA algorithms, always select the shortest path to transmit a burst, and thus the difference between these two algorithms lies in the computation of the time offset (*TO*) after which the burst transmission should begin: the Dijkstra algorithm uses information regarding reservations made only for the first link on the shortest path, while the Dijkstra/CA algorithm takes into account the reservations made at all the links of the shortest path.

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, as in [23]. We assume that dropped bursts are retransmitted because (a) this is what would happen in a real network, where the loss of data is usually not acceptable (without retrials, a higher layer protocol such as TCP would retransmit dropped packets) and (b) this is a more reliable model for evaluating the performance of the proposed algorithms. Note that if retrials were not incorporated, the network would tend to drop bursts that travel more hops, and the throughput results would not be representative of actual performance. The ingress node stores the burst in a limited-size buffer until it is successfully transmitted. In our simulation experiments the size of the ingress buffer was set equal to 256Mbytes per node. For the traffic loads simulated we never observed a case of a burst being dropped due to buffer overflow.

We obtained simulation results for two network topologies: a 5x5 mesh with wraparounds, and the NSF network topology. In the mesh topology, the nodes were arranged along a two-dimensional grid, with neighboring nodes placed at distance of 50 km from each other. In the NSFnet, in addition to the actual link physical lengths (as found in [24]), we also experimented with link lengths that are a fraction (10%, 30%, and 60%) of these lengths. All links were assumed to be bi-directional and the link bandwidths C was set equal to 1 Gb/s.

We used the average end-to-end delay experienced by a burst as the main metric for assessing performance. Additional performance metrics used were the average number of computed paths per request, and the average number of operations required to handle a request. The average number of operations is defined as the number of bitwise comparisons,

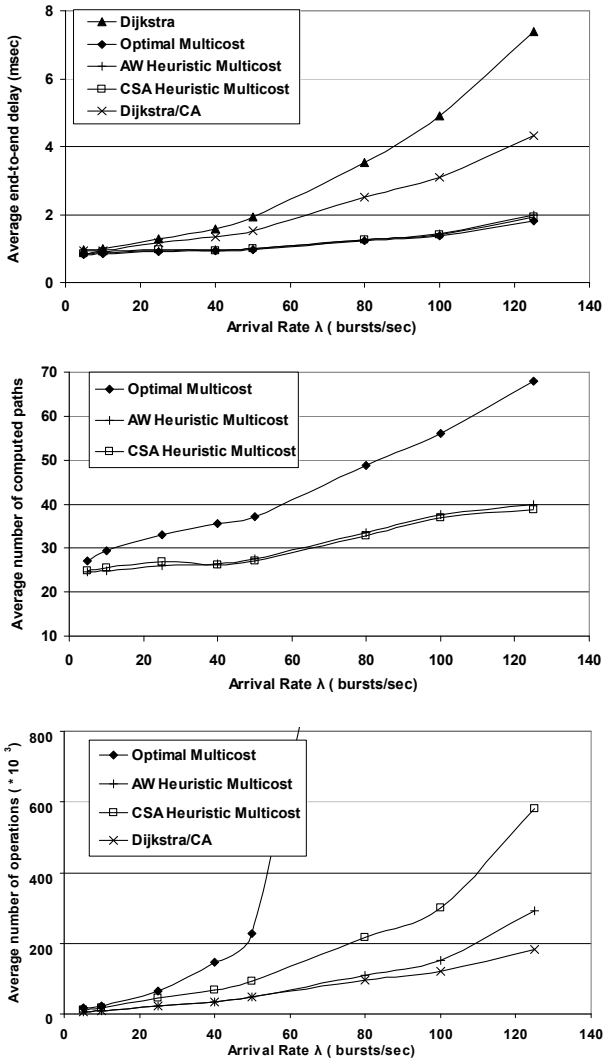

Fig. 4: (a) Average end-to-end delay, (b) average number of computed paths and (c) average number of operations, for $\bar{I}$ =300KB and various λ.

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.

A. Results for the 5x5 mesh with wraparounds network

For the experiments in this section we used: τ=0.01 msec, u=8000 (maximum delay $D = u \cdot \tau$ = 80 msec), $\bar{I}$=300kB and λ varied between 5 and 125 bursts/sec per node.

Fig. 4a illustrates the average-end-to-end burst delay for the algorithms examined. The optimal multicost algorithm outperforms the Dijkstra and the Dijkstra/CA algorithms, while the difference between the multicost algorithm and the AW and CSA heuristics are marginal. Fig. 4b shows the average number of computed paths per request. We have excluded from this graph the results for the Dijkstra and Dijkstra/CA algorithms, since they perform the path computation process only once. As expected, using the pseudo-domination relationship drastically reduces the average number of computed paths. Regarding the average number of operations (Fig. 4c), the multicost algorithm exhibits the worst performance, as expected. The number of

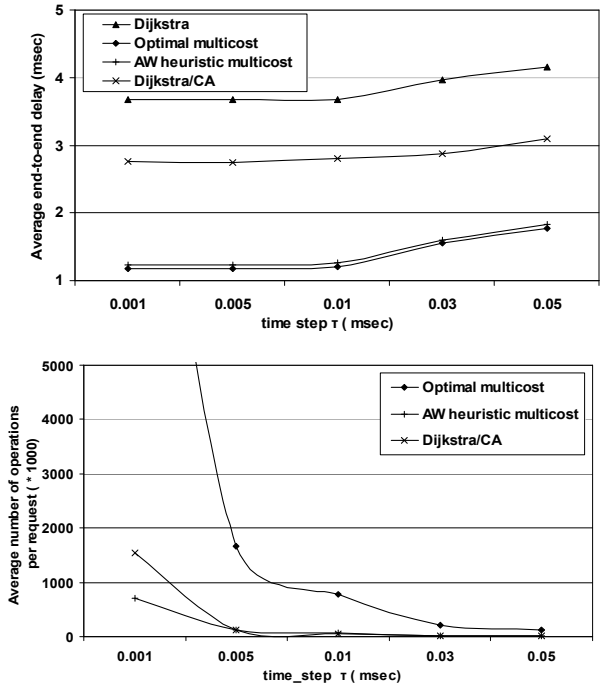

Fig. 5: (a) Average end-to-end delay and (b) average number of operations per burst, for λ=80 burst/sec, $\bar{I}$=300 KB, and various values of τ. The product of τ with u was kept constant.

operations is reduced in the AW and CSA heuristics. The number of operations in the CSA algorithm is larger than that of the AW algorithm, since obtaining the b-consecutive slot availability value requires one complete scan of the CAV vector. The number of operations of the AW algorithm is similar to that of the Dijkstra/CA algorithm for small λ, and becomes increasingly worse than that as λ increases.

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

In the remainder we will focus only on the AW algorithm since the delay performance of CSA was always very similar to AW, while its complexity was always worse than that of the AW algorithm.

B. Effect of time discretization step τ

The time discretization process and the size u of the binary capacity availability vectors have a considerable effect on performance. Fig. 5a and 5b show the average end-to-end delay and the average number of operations, respectively, for different values of the discretization step τ, for λ=80 bursts/sec and $\bar{I}$=300 kB. For fair comparison, we have kept the product $D = u \cdot \tau$ = 80 msec constant. From Fig. 5a we can observe that the time discretization process affects the average end-to-end delay of all the algorithms. Since reservations are performed for integer number of τ-time units, the performance deteriorates as τ increases. On the other hand, large values of τ correspond to small values of u, and thus the number of operations is inversely proportional to τ (Figure 5b).

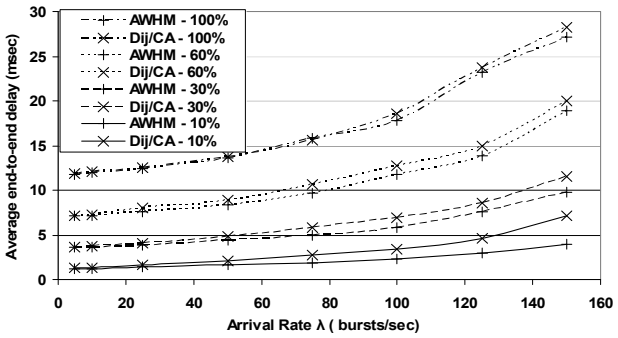

Fig. 6: Average end-to-end delay per burst per burst, for $\overline{I}$ =300 KB and various λ. The lengths of the links are 10%, 30%, 60% or 100% of the actual distances of the NSFnet.

C. NSF network and effect of propagation delays

In the algorithms examined here each node maintains a "picture" of the link states that is used to compute the non-dominated or non-pseudo-dominated paths, respectively, while in the Dijkstra/CA algorithm it is used to compute the time offset of the burst. Clearly, information on a link reservation cannot be used if it reaches a source after path selection. Therefore, we expect propagation delays to have a considerable effect on the algorithms examined, except for the Dijkstra algorithm that only uses information on the first hop.

Fig. 6 shows the average end-to-end delay experienced by a burst for $\overline{I}$=300kB for varying λ and link propagation delays. More specifically, we have experimented with link lengths that are fractions (10%, 30% and 60%) of the original distances. We have graphed only the performance of the AW multicost heuristic algorithm and the Dijkstra/CA algorithm. These results indicate that the performance improvements that can be achieved by the AW multicost heuristic algorithm are substantial when the link lengths are 10% and 30% of the actual lengths (the network propagation delay is then around 1.3 msec and 3.9 msec respectively) and diminish as the propagation delays increase. Note that when the propagation delays increase, the complexity of the depicted algorithms decrease since this reduces the useful update-reservation information that can reach each ingress.

VIII. Conclusions

We presented several multicost algorithms for routing and scheduling connections in networks that support advance reservations. We initially presented an optimal scheme of non-polynomial complexity and then we provided two pseudo-dominations relationship to obtain polynomial time algorithms. The proposed multicost algorithms were evaluated in an Optical Burst Switched network. The performance results showed that the multicost algorithms significantly outperform other algorithms, 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 it can be decreased by increasing τ at a small penalty in terms of end-to-end delay. The proposed polynomial time AW heuristic multicost algorithm yielded delay performance that is very close to that of the optimal algorithm, while maintaining the number of operations at low levels. Finally, the performance improvements are more pronounced when the propagation delays of the network are small, in which case the utilization information of the links maintained at the ingress routers is more accurate and up-to-date.

IX. Acknowledgements

This work has been supported by the European Commission through the 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.

References

[1] R. Braden, L. Zhang, S. Berson, S. Herzog, S. Jamin, "Resource Reservation Protocol (RSVP)", RFC 2205, Sept 1997.
[2] R. Guerin, A. Orda, "Networks with Advance Reservations: The Routing Perspective", INFOCOM, 2000.
[3] K. Nahrstedt. R. Steinmetz, "Resource Management in Networked Multimedia Systems", Computer, vol. 28, pp. 52-63, 1995.
[4] C. Qiao, M. Yoo, "Optical burst switching (OBS)–a new paradigm for an optical Internet", Journal of High Speed Networks, vol. 8, pp. 69–84, 1999.
[5] I. Foster, C. Kesselman, C. Lee, R. Lindell, K. Nahrstedt, A. Roy, "A Distributed Resource Management Architecture that Supports Advance Reservations and Co-Allocation", Intl. Workshop on Quality of Service, 1999.
[6] E.A. Varvarigos, V. Sharma, "An efficient reservation connection control protocol for gigabit networks", Computer Networks and ISDN Systems, July 1998, vol.30, (no.12):1135-56.
[7] L. C. Wolf, L. Delgrossi, R. Steinmetz, S. Schaller, "Issues of Reserving Resources in Advance", Lecture Notes in Computer Science, vol 1018, 1995.
[8] D. Ferrari, A. Gupta, G. Ventre, "Distributed Advance Reservation of Real-Time Connections", Multimedia Systems, vol 5, Issue 3, 1997.
[9] A. Schill, F. Breiter, S. Kuhn, "Design and evaluation of an advance reservation protocol on top of RSVP", 4th IFIP Intl. Conference on Broadband Communications, pp. 23– 40, 1998.
[10] J. Zheng, H. Mouftah, "Routing and Wavelength Assignment for Advance Reservation in Wavelength-Routed WDM Optical Networks", ICC, 2002.
[11] J. Turner, "Terabit burst switching", Journal of High Speed Networks, vol 8, 1999.
[12] M. Yoo, C. Qiao, S. Dixit, "QoS performance of optical burst switching in IP-over-WDM networks", Journal on Selected Areas in Communication, vol. 18, Oct. 2000.
[13] T. Coutelen, H. Elbiaze, B. Jaumard, "An Efficient Adaptive Offset Mechanism to Reduce Burst Losses in OBS Networks", GLOBECOM 2005.
[14] 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, vol. 5: pp.705-718, Oct. 1997.
[15] I. Foster, C. Kesselman, "The Grid 2: Blueprint for a New Computing Infrastructure", Morgan Kaufmann, 2003.
[16] W. Smith, I. Foster, V. Taylor, "Scheduling with advanced reservations", IPDPS'00, pp 127-132, 2000.
[17] J. Xing, C. Wu, M. Tao, L. Wu, H. Zhang, "Flexible Advance Reservation for Grid Computing", GCC 2004, pp 241–248, 2004.
[18] F. Gutierrez, E.A. Varvarigos, S. Vassiliadis, "Multicost Routing in Max-Min Fair Networks", 38th Allerton Conference on Communicating, Control and Computing, 2000.
[19] Z. Wang, J. Crowcroft, "Quality-of-service routing for supporting multi-media applications", Journal on Selected Areas in Communications, vol.14, Sept. 1996.
[20] P. Van Mieghem, F. Kuipers, "Concepts of exact QoS routing algorithms", Transactions on Networking, vol. 12, no 5, Oct, 2004.
[21] K. Manousakis, V. Sourlas, K. Christodoulopoulos, E.A. Varvarigos, K. Vlachos, "A Bandwidth monitoring mechanism: Enhancing SNMP to record Timed Resource Reservations", Journal of Network and Systems Management, 2006.
[22] The Network Simulator – ns-2: http://www.isi.udu/nsnam/ns
[23] A. Maach, G.V Bochmann, H. Mouftah, "Robust optical burst switching", Networks 2004, pp. 447-452, 2004.
[24] NSF network: http://moat.nlanr.net/INFRA/NSFNET.html