Deadline-aware Advance Reservation Scheduling Algorithms for Media Production Networks

Maryam Barshan[a,*], Hendrik Moens[a], Jeroen Famaey[b], Filip De Turck[a]

[a] *Department of Information Technology, Ghent University – iMinds*
Gaston Crommenlaan 8/201, B-9050 Gent, Belgium
[b] *Department of Mathematics and Computer Science, University of Antwerp – iMinds*
Middelheimlaan 1, 2020 Antwerpen, Belgium

Abstract

In the media production process a substrate network can be shared by many users simultaneously when different media actors are geographically distributed. This allows sophisticated media productions involving numerous producers to be concurrently created and transferred. Due to the predictable nature of media transfers, the collaboration among different actors could be significantly improved by deploying an efficient advance reservation system. In this paper, we propose a model for the advance bandwidth reservation problem, which takes the specific characteristics of media production networks into account. Flexible and time variable bandwidth reservations, meeting delivery deadlines, supporting splittable flows and interdependent transfers and all types of advance reservation requests imposed by the media production transfers are incorporated into this model. In addition to the optimal scheduling algorithms, which are presented based on this model, near optimal alternatives are also proposed. The experimental results show that the proposed algorithms are scalable in terms of physical topology and granularity of time intervals and obtain a satisfactory performance, executing significantly faster than an optimal algorithm and within 8.78% of the optimal results.

Keywords: Advance bandwidth reservation, media production network, video streaming, deadline-aware scheduling.

1. Introduction

In the media production industry, a team of artists, editors, reporters and producers works simultaneously at geographically distributed locations producing and processing content, music, commentary, special effects, etc. Various producers and actors could then access these individual elements over a shared network to integrate them and thereby produce a complete product. In the media creation process, reliability of the transport is of crucial importance.

Predictability is a key feature of traffic in media production networks. Traffic characteristics in terms of bandwidth requirements, the time when the contents are ready, and the deadline for the data to be completely transferred to the destinations, are mostly known several hours ahead of time. The predictable nature of these transfers makes it possible to use resource reservation techniques. Therefore, a management system can efficiently manage the transmission. In general, two types of resource reservation can be distinguished [1]: Immediate Reservation (IR) and Advance Reservation (AR). While just-in-time reservation is applied in IR, the principle behind AR relies on the resource reservation times before the actual time when the resource is used. Assuming prior knowledge of the network structures and different requests, advance reservation makes it possible to schedule network requests optimally.

In computer networks, bandwidth is a valuable resource. Particularly for multimedia transfers, where large amounts of content, such as video files, have to be transmitted, efficient bandwidth management is an important factor [2]. In bandwidth-limited networks, an efficient bandwidth reservation mechanism needs to be defined to meet the QoS requirements and deadlines. The next generations of media production networks are expected to efficiently support advance reservation systems for different delivery services, so the desired QoS requirement and resource utilization could be ensured.

In this paper, we propose a set of novel AR scheduling algorithms, optimized for media production networks. Such networks impose requirements not supported by existing AR scheduling techniques. First, the start time of requests is generally flexible, the deadline is fixed, and the reserved bandwidth may vary over the lifetime of the reservation. This combination of flexible start times and elastic bandwidth allocation has not received much attention in research to date [3]. Second, in media production networks, multiple requests may depend on each other. Until now to the authors' knowledge, this aspect has remained unexplored. Third, it should be possible to split requests over multiple paths, in order to further optimize

*Corresponding author
Email address: maryam.barshan@intec.ugent.be (Maryam Barshan)

bandwidth utilization.

We propose a centralized advance reservation system which based on our evaluation scales to the size of realistic media production networks and demand patterns. We present a model to solve this variant of the AR scheduling problem, and propose various advance reservation algorithms based on our designed model. This model is an instance of time variable scheduling which is known as multicommodity over time problem. In this context, commodity corresponds to a telecommunication traffic demand between two media actors. It has been proven that the complexity of multicommodity flow over time without caching is strongly NP-hard [4]. Therefore, we also came up with efficient and near-optimal heuristic solutions which are more practical. In both optimal and heuristic approaches the main goal is threefold: 1) delivery of the requests before their deadline. 2) maximizing the number of admitted requests. 3) processing requests as quickly as possible.

Both approaches can be used in static and dynamic settings. The Static Advance Reservation Algorithm (SARA) assumes all requests are known at the start of the reservation period. By contrast, the Dynamic Advance Reservation Algorithm (DARA) supports rescheduling in order to incorporate new requests at runtime. The dynamic advance reservation system tries to admit new arrival requests while rescheduling the previously admitted requests is a must. We provide a thorough analysis of the algorithms based on in-depth simulation results. They are compared and the impact of their parameters on the solution quality is evaluated.

The remainder of this paper is organized as follows. Section 2, reviews the related work. Section 3 describes the scenarios as well as architecture and components of the proposed media production network. Section 4 explains the concepts, assumptions and AR scheduling formulation for media production networks. The proposed algorithms are explained in Section 5. A comparison of optimal and near-optimal algorithms and a performance evaluation of offline and online settings are provided in Section 6. Finally, Section 7 concludes the paper.

2. Related work

There has been a large number of theoretical as well as practical experimental work [5, 6, 7, 8] related to the advance reservation problem. Here we study some of the most relevant work. The authors in [9] and [10] focus on re-routing in advance reservation networks. Our formal model is inspired by their ILP-based solutions called GILP and DILP [9]. The GILP assumes that the entire set of requests is known beforehand and the DILP is designed to work in an online setting. However our approach is different as their models assume only streaming requests with fixed time intervals and dedicated bandwidth remains fixed and equal to the demand during the entire reservation. Dependencies among the requests are also ignored. [10] is the extension of [9] in realistic multi-domain networks which addresses the implementation challenges related to advance reservation solutions.

Charbonneau et al. [3], survey the literature on advance reservation routing and scheduling algorithms, specifically focused on WDM networks. It has defined four types of advance reservation requests based on whether their start time and their duration are specified or not. All these classifications, which will be discussed in detail later, are supported in our approach. In addition, our approach is elastic which means that the allocated bandwidth is variable over time. According to this reference only two AR scheduling algorithms have been proposed that support elastic reservations [11, 12]. However, they both assume a fixed start time. Sharma et al. [13] present an algorithm called RRPC which addresses multiple flexible requests for bandwidth reservation between two end points. RRPC is deadline-aware in which any reservation that meets the deadline is acceptable. However all the requests have a same source and destination, flow splitting is not allowed and a single path is chosen for all the requests. Another work [14] focuses on dynamically transporting of large volume of data in e-science networks. The optimization consists of two steps admission control and scheduling. Periodically the central controller gathers all the new requests, runs admission control, and then schedules new and unfinished jobs.

Furthermore, the problem addressed in this work is related to the multicommodity flow problem (MCFP). The multicommodity flow problem can be described as follows: a set of individual flows have to be transferred in a dimensioned network without violating the capacity limits [15]. The resource allocation algorithm should find an optimal routing path to transfer the flows through the network. In [16], unsplittable flow and single path MCFPs are studied. Comprehensive surveys on the approaches to solve multicommodity flow problems (MCFP) and their variants are provided in [17, 18]. Our approach further deals with the problem of flow variation over time and solves an MCFP as a subproblem. In network flow problems, having variable flows over time is crucial. Dynamic flows or flow variation over time are primarily introduced by Ford and Fulkerson [19, 20]. They introduced variable flows over time as equal as static flow problem, building another temporal dimension over the network. This makes use of time-expanded networks. A time-expanded network is a copy of network in each discrete time step. Also, Fleischer et al. [21] have mentioned that in literature hardly any results on multicommodity over time are noted.

Jiancong et al. in [22] have stated that the single-path approach on which the Internet routing protocols is based, could not meet the delay requirements when the video streams are transferred over bandwidth-limited networks. They proposed a multipath routing of video contents over bandwidth limited network. However the main focus of their work is on delay and over the Internet, and therefore no reservation is considered.

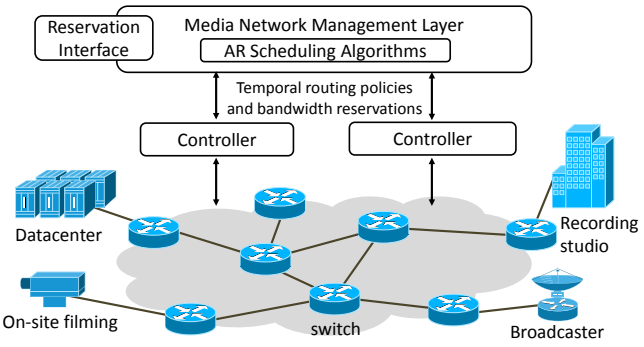


Figure 1: Media production network architecture and components.

Balman et al. [23] have focused on advance bandwidth reservation for on-demand data transfer in scientific applications. However, their work differs from our approach as they purely focus on data transfers, not video streaming sessions, and the routing mechanism is based on single-path in contrary to our multi-path approach. In addition, our approach considers dependencies among requests. To the best of our knowledge, dependencies among requests which is explicitly incorporated in our algorithms, have not received adequate attention in the literature by state of the art approaches.

This work is an extension of our previous work [24], in which only the static and dynamic ILP-based models are introduced. The main focus of this previous work was to investigate the viability of AR mechanisms in media production networks and to find the optimal solutions, determining the most appropriate objective function in our optimal models. We defined two objective functions and compared their performance. We found that the so called ASAP objective function, which in addition to maximizing the number of admitted requests also tries to schedule requests in earlier time slots leads to better results. In this article several new heuristic approaches are proposed which are near-optimal and computationally less-complex compared to ILP-based approaches. In our evaluation, their performance is compared with the highest quality optimal algorithms (i.e. algorithms based on the ASAP objective function).

3. Media production network architecture

The envisioned media production network is depicted in Figure 1. The different actors and locations involved in the media production process, such as for example recording studios, on-site filming crews, broadcasters, and storage datacenters, are connected to a shared wide-area network, consisting of interconnected switches. The network supports the exchange of raw and encoded multimedia content between an arbitrary set of actors, both in the form of file transfers and streaming. The management layer provides a reservation interface, that allows the users of the network to reserve bandwidth over certain time periods in the future. The AR scheduling algorithms are responsible for reserving the required amount of bandwidth resources for all requests. With each request, they associate one or multiple paths from source to sink with a specific amount of reserved bandwidth. In case the deadline of a transfer cannot be guaranteed, the reservation interface rejects it. When multiple transfers depend on each other, either all or none of them are admitted.

The output of the scheduling algorithms takes the form of a set of temporal routing policies (i.e., the paths associated with all requests over time) and bandwidth reservations (i.e., the amount of bandwidth resources to associate with each flow over time). This information can be transferred to the network controllers, that use it to configure the switches in the media production network. The controllers keep track of the temporal aspects of the policies, adjusting configurations when necessary.

In the media production industry multiple actors, which are involved in one production project, are interacting and transferring media content. If one of those transfers is not successfully done the whole project can be affected. This forms dependencies among different transfers. We refer to the set of all transfers of a project as a scenario. The scenario consists of several interdependent video transfers. We refer to each single transfer as a request. A request can have a fixed start time, end time and/or duration, or may depend on the other requests.

The video transfer types which are supported in this work are of two types which can either be video streams (VS) or file based videos (FB). We assume that for FB requests, volume and for VS requests duration is always known. As stated by [25, 26, 3] advance reservation requests are classified into four individual categories.

- STSD: Start time of the request is specified, its duration is also specified.
- STUD: Start time of the request is specified, but its duration is unspecified.
- UTSD: Start time of the request is unspecified, but its duration is specified.
- UTUD: Start time of the request is unspecified, its duration is also unspecified.

In this article all four classes are taken into account. As for VS requests the duration is always specified, if the start time of a VS request depends on other requests, this stream belongs to the UTSD class. The class of independent VS requests is STSD. In case of file transfers, the reserved resources for a request may vary over time, as long as the delivery deadline is satisfied. Moreover, even if the start time of FB request is specified, it refers to the time when the file is ready to be transferred. The time when the file transfer starts could be in future time slots. Since for file transfers duration and start time might be undefined and fluid, independent and dependent file based requests are classified as flexible UTUD and flexible STUD respectively. This is illustrated in Table 1.

During the scheduling process, four statuses are defined for each scenario: 1) Submitted: When a scenario entered

Table 1: Media production video request taxonomy.

Request types	Specified start time	Specified duration	Dependent VS	Dependent FB	Independent VS	Independent FB
STSD	yes	yes			X	
STUD	yes	no				X
UTSD	no	yes	X			
UTUD	no	no		X		

to the system, but the admission process has not been started yet. 2) Pending: When a scenario is being processed and it is waiting for the admission decision. 3) Admitted: When all the requests of the scenario could be scheduled and transferred. This status implies the transmission guarantee. 4) Rejected: When the scheduler is not able to respond to any of the scenario requests' demands.

The remainder of this paper focuses on the AR scheduling algorithms.

4. Advance reservation scheduling model

We first present a formal model for the advance reservation scheduling of network bandwidth. The model can be used to schedule collections of requests, that consist of multiple interdependent and deadline-constrained network transfers. The network is represented as a graph with network nodes N and edges E.

Requests are grouped into scenarios, contained in the set S, that represent a complex workflow. These workflows must be executed in their entirety, so when a scenario is admitted, all requests must be executed. The model only admits those scenarios for which sufficient bandwidth can be guaranteed during the reservation period. When a scenario is rejected, none of its requests are executed. The various requests within a scenario may depend on each other, meaning that one request can only start when other requests have been finished.

The requests of all scenarios are stored in R. The model supports two types of network transfers: video streaming and large file transfers. Consequently R consists of both types. To make distinction between two types R_f which refers to file-based flows and R_s which refers to the streaming requests are defined.

In this model the n^{th} request is denoted by $r^n = (s^n, d^n, t_s^n, t_e^n, i^n, b^n)$ comprising of the source of the request s^n, the destination node d^n, the time when the data for file-based request is ready to transfer t_s^n (or fixed start time for video streaming request), the deadline for the transmission of the data of file-based request t_e^n (or fixed end time for video streaming request), the duration of each request i^n and finally the bandwidth demand of the request b^n. In particular, r_f^n and r_s^n refers to file-based and video streaming requests respectively. Moreover, the volume of the files are denoted by v^n and the time slot size by I. Table 2 lists the notations which have been used to define the model.

Table 2: Symbols and notations used in the formal models.

Variable	Description
N	Set of network nodes.
E	Set of network links.
S	Set of all scenarios ($s \in S$).
R_f	Set of file-based video requests.
r_f^n	The n^{th} request of set R_f.
R_s	Set of video streaming requests.
r_s^n	The n^{th} request of set R_s.
R	Set of all requests ($R_f \cup R_s$).
R_o	Set of all old requests.
r^n	The n^{th} request of set R, denoted by $r^n = (s^n, d^n, t_s^n, t_e^n, i^n, b^n)$.
s^n	Source node of request r^n.
d^n	Destination node of request r^n.
t_s^n	Start time for the request r^n. Decision variable when not specified.
t_e^n	Deadline for the request r^n. Decision variable when not specified.
i^n	Duration of request r^n.
b^n	Required bandwidth of r^n.
v^n	Volume of r_f^n for file-based requests (in bit).
$\beta^{n,e,k}$	Decision variable. Dedicated bandwidth over link e, request r^n and time interval k.
$SU^{n,k}$	Binary decision variable. 1 iff in time slot k any reservation is done for request n, 0 otherwise.
A^n	Binary decision variable. 1 iff request r^n is admitted, 0 otherwise.
A^s	Binary decision variable. 1 iff scenario s is admitted, 0 otherwise.
I	Duration of each time interval (in second).
t_s^{min}	Minimum start time of all reservations.
t_e^{max}	Maximum end time of all reservations.
B^e	Bandwidth capacity of link e.
E_v^{out}	This collection contains all edges starting from node v (egress).
E_v^{in}	This collection contains all edges ending in node v (ingress).

4.1. Decision variables

The goal of the model is to determine when and how requests are transferred over the network. Binary decision variables A^s and A^n are used to represent whether or not scenario s or request n are admitted. When the scenario is admitted, a collection of decision variables $\beta^{n,e,k}$ determines the amount of bandwidth for a request n that is sent over edge e during time slot k.

$$A^n \in [0,1] \qquad \forall r^n \in R$$
$$A^s \in [0,1] \qquad \forall s \in S$$
$$\beta^{n,e,k} \in \mathbb{R}^+ \qquad \forall r^n \in R, \forall e \in E, k \in [t_s^{min}, t_e^{max}]$$

For some requests their start and end times are not specified and depend on the start or end time of other requests. In this case, the t_s^n, t_e^n or both of a request n may become

decision variables of which the value is determined during the optimization process. To support these kinds of scenarios additional decision variables and constraints need to indicate whether a request is active during a given time slot. Therefore, we define the binary time slot use decision variable $SU^{n,k}$ that takes on value 0 when a request n is inactive during time slot k. These variables are defined for all requests where t_s^n, t_e^n or both are decision variables, but not for requests of which start and end time are known.

$$SU^{n,k} \in [0,1] \qquad \forall r^n \in R, k \in [t_s^{min}, t_e^{max}]$$
$$t_s^n \in \mathbb{R}^+ \qquad \forall r^n \in R \text{ if start time is variable}$$
$$t_e^n \in \mathbb{R}^+ \qquad \forall r^n \in R \text{ if end time is variable}$$

4.2. Objective function

The objective function, shown in Expression 1, maximizes the number of admitted requests, but also tries to schedule requests as soon as possible. This is done by adding a second factor to the objective function that achieves higher values when requests are scheduled in earlier timeslots. This second term is normalized to ensure it will not interfere with the primary objective of maximizing the number of accepted requests.

$$\max \sum_{r^n \in R} A^n + \frac{\sum_{r^n \in R} \sum_{e \in E_{s^n}^{out}} \sum_{k \in [t_s^n, t_e^n]} \frac{\beta^{n,e,k}}{k}}{\sum_{r^n \in R} \sum_{e \in E_{s^n}^{out}} \sum_{k \in [t_s^n, t_e^n]} \frac{B^e}{k}} \quad (1)$$

4.3. Flow constraints

Requests are scheduled over a network, which means they are subject to capacity and network flow constraints. The capacity constraint, shown in Expression 2, ensures that the cumulative bandwidth reservation over each link does not exceed its bandwidth capacity. This constraint is specified for every edge, and for every time slot.

$$\sum_{r^n \in R} \beta^{n,e,k} \leq B^e \qquad \forall e \in E, \forall k \in [t_s^{min}, t_e^{max}] \quad (2)$$

All network nodes that are not source or sink of a flow are subject to a flow conservation constraint, shown in Expression 3, to ensures the incoming flow equals outgoing flow. The network entering and leaving the source and sink of the flow is dependent on the type of request. For a file transfer request, an entire volume v^n must be transferred between the start and end times, shown in Expression 4. For these requests, the amount of data transferred can vary between timeslots. Video streaming requests are handled differently, as they require a constant amount of resources during all time intervals between the start and end time of the request. This is shown in Expression 5. To minimize the occurrence of loops within the network, constraints preventing incoming flow in the source node and outgoing flow in the sink node are added. These constraints are shown in Expressions 6 and 7.

$$\sum_{e \in E_v^{out}} \beta^{n,e,k} = \sum_{e \in E_v^{in}} \beta^{n,e,k} \quad (3)$$
$$\forall r^n \in R, \forall k \in [t_s^{min}, t_e^{max}], \{\forall v \in N | v \notin \{s^n, d^n\}\}$$

$$\sum_{k \in [t_s^{min}, t_e^{max}]} \sum_{e \in E_{s^n}^{out}} \beta^{n,e,k} \times I = v^n \times A^n \quad \forall r_f^n \in R_f \quad (4)$$

$$\sum_{e \in E_{s^n}^{out}} \beta^{n,e,k} = b^n \times A^n \quad \forall r_s^n \in R_s, \forall k \in [t_s^{min}, t_e^{max}] \quad (5)$$

$$\sum_{e \in E_{s^n}^{in}} \beta^{n,e,k} = 0 \qquad \forall r^n \in R, \forall k \in [t_s^{min}, t_e^{max}] \quad (6)$$

$$\sum_{e \in E_{d^n}^{out}} \beta^{n,e,k} = 0 \qquad \forall r^n \in R, \forall k \in [t_s^{min}, t_e^{max}] \quad (7)$$

4.4. Interdependent requests

Start and end times of requests may either be input variables or decision variables. Dependencies between different requests are handled by Expressions 8, 9, 10, 11, 12, 13 and 14. First, Expression 8 ensures either all or none of the requests of a scenario get admitted.

$$A^n = A^s \qquad \forall r^n \in R \quad (8)$$

Expression 9 is defined to connect $\beta^{n,e,k}$ and $SU^{n,k}$ values, which is needed if either the start or end time of a request is a decision variable. This constraint ensures that $SU^{n,k}$ can only become zero if $\beta^{n,e,k} = 0$.

$$\beta^{n,e,k} \leq SU^{n,k} \times B^e \quad \forall e \in E, \forall k \in [t_s^{min}, t_e^{max}], \forall r^n \in R \quad (9)$$

If the start time is known and predefined as an input variable, then Expression 10 ensures that no bandwidth is dedicated to request r^n before t_s^n.

$$\beta^{n,e,k} = 0 \qquad \forall e \in E, \forall r^n \in R, \forall k \in [t_s^{min}, t_s^n) \quad (10)$$

If the start time is not specified and depends on other requests, then t_s^n is a decision variable. In that case, the constraint shown in Expression 11 is used to ensure $SU^{n,k}$ becomes 0 for values of $k < t_s^n$, ensuring nothing is transferred. Dependencies between time variables can then be added as shown in Expression 12, which ensures that the request n is started only when all the requests on which request n depends are finished.

$$t_s^n \leq k + (1 - SU^{n,k}) \times t_e^{max} \qquad \forall r^n \in R, \forall k \in [t_s^{min}, t_e^{max}] \quad (11)$$

$$t_s^n \geq t_e^{n'} + 1 \qquad \{\forall r^n \in R | r^n \textit{ depends on } r^{n'}\} \quad (12)$$

When the end time is an input variable, then Expression 13 ensures that no bandwidth is dedicated to request n after t_e^n.

$$\beta^{n,e,k} = 0 \qquad \forall e \in E, \forall r^n \in R, \forall k \in (t_e^n, t_e^{max}] \quad (13)$$

If the end time is not specified, t_e^n is a decision variable. In this case, a constraint is added to ensure $SU^{n,k}$ becomes 0 for values of $k > t_e^n$, ensuring nothing is transferred after the end time. This constraint is shown in Expression 14.

$$t_e^n \geq k - (1 - SU^{n,k}) \times t_e^{max} \qquad \forall r^n \in R, \forall k \in [t_s^{min}, t_e^{max}] \quad (14)$$

4.5. On-line model

The model described in the previous section can be used to statically compute a schedule for the execution of a collection of scenarios, provided all scenarios are known beforehand. In practical media production networks, the requests however arrive on-line over time. Therefore, a dynamic, on-line approach is needed that adapts the schedule at runtime. We present this on-line model as an extension of the previously discussed static model, meaning is implemented with the previously defined decision variables and objective functions.

The on-line model assumes that a previous schedule exists, and that one or more requests are added that must be scheduled. This results in a new schedule that contains both the original requests, and the new requests. We assume that a request may not be canceled once it has been accepted, meaning that while old requests may be rescheduled, they may not fail. Besides the constraints of the original model, one additional constraint (shown in Expression 15) is therefore added to ensure that previously admitted requests remain accepted.

$$A^s = 1 \quad \forall r^n \in R_o \tag{15}$$

5. Advance reservation algorithms

This section, first describes the static and dynamic reservation schemes. The second part implements the models which have been defined in the previous section. In the third part the heuristic algorithms, which in general we refer to as Sequential Priority Based (SPB), are proposed to resolve the high computational complexity and scalability issue of the ILP solutions.

5.1. Static & dynamic reservation

The algorithms provided in this section are either static or dynamic, which can be used "offline" or "online". The static algorithms, which we refer to as Static Advance Reservation Algorithm (SARA), can be used to generate a schedule when all requests are known in advance. However, in practice, some requests may not be known from the start of the scheduling, making it impractical to use the SARA. Therefore, a dynamic version of the resource reservation algorithm is needed. When not all requests are known from the start, and new ones are added throughout the day, the Dynamic Advance Reservation Algorithm (DARA) can be used. When new scenarios enter to the reservation system, the DARA re-optimize the reservation by re-routing existing reservations in order to accommodate new scenarios' requests. This re-optimization is performed for the entire schedule starting from the next time slot. In DARA, we assume that new incoming scenarios have lower priority as the previous requests are already admitted and rejecting them violates the agreed SLA. Therefore, in DARA requests are divided in three categories based on their progress:

- **Scheduled:** When a request is scheduled, it will start to execute during some time slot in the future. As the request is not yet running during the trigger point, no special considerations are needed.
- **Finished:** A request is considered finished when it has finished executing at the time of the trigger point. The request itself can therefore be removed from the on-line model input. If the start or end times of other requests depend on the end time of this request, the final end time can be added as an input to the model.
- **In progress:** A request is in progress when it has started, but has not finished yet at the time of the trigger point. These requests must still be considered in the on-line model input, but the amount of data that was already transferred must be removed from the total request demand.

5.2. ILP based advance reservation algorithms (ILP)

In this section we define two algorithms based on the model presented in the previous section. The first algorithm is based on the static model. In an "online" setting the second approach, which makes use of the on-line version of the model, can be used.

5.2.1. ILP based Static Advance Reservation Algorithm ($SARA_{ILP}$)

This algorithm is based on the static formal model which assumes that all the scenario arrivals are known beforehand, which results in an optimal schedule.

Using the previously defined constraints, the multi-path model is likely to result in feasible but undesirable solution, as cycles may potentially occur in intermediate network nodes. As the model is implemented using an ILP, these cycles will never impact the optimality of the result as specified by the objective function. There are two possible approaches to address these cycles. 1) Firstly, it would be possible to modify the model by changing the objective function, adding an additional factor that minimizes the edge use. This would however increase the complexity of the model and consequently lead to an increase in execution duration. Furthermore, this would make it more difficult to balance the different optimization objectives. 2) Alternatively, the results of the algorithm can be post-processed by removing the cycles after the ILP has been solved. This approach has the advantage of limiting the complexity of the ILP model, and as stated previously has no impact on its optimality.

Because of these considerations, we use a post processing algorithm after the ILP optimization. During this post-processing phase, we look for cycles in each reserved path and remove them.

5.2.2. ILP-based Dynamic Advance Reservation Algorithm ($DARA_{ILP}$)

The $DARA_{ILP}$ invokes the Integer Linear Programming (ILP) formulation of the model multiple times whenever new scenarios arrive. In this algorithm, an initial

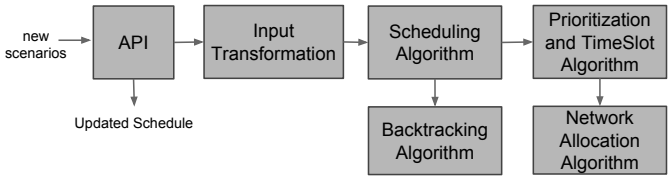


Figure 2: Different components of the Sequential Priority Based Advance Reservation Algorithm (SPB).

schedule is generated using the static model, which is then iteratively updated using the on-line model as new scenarios arrive. The input of the on-line model must however be modified at every trigger point to take into account the work that has already been executed. The demands of previously admitted, unfinished and in progress requests are updated based on the the data that has already been transferred. Then new requests as well as modified requests are scheduled together.

5.3. Sequential Priority Based advance reservation algorithms (SPB)

In general, the ILP-based algorithms have a high computational overhead, particularly with fine-grained time slot sizes and large physical networks. The Sequential Priority Based (SPB) advance reservation algorithm is a heuristic approach which is proposed to resolve the scalability limitations of the ILP solutions. The admission control process is also integrated into the algorithm and once a scenario is admitted, it will never be denied by the scheduler in future. In this approach, the scenarios are sequentially admitted and scheduled.

Different components of the SPB advance reservation algorithm are illustrated in Figure 2. As can be observed from this figure, the new scenarios enter to the reservation system through an API. Then any transformation can be applied. For example in dynamic approach before the scheduling algorithm invocation, the previously admitted scenarios' demand needs to be updated. In the next step the scenario requests are prioritized and in each time slot the algorithm sequentially calls the network allocation algorithm for each scenario request. If this process is successfully terminated the new scenario is admitted, and the schedule is updated. Otherwise, the previous scheduling and network state remain untouched and the scenario is rejected. Again we discern two algorithm variants: $SARA_{SPB}$ and $DARA_{SPB}$.

5.3.1. Sequential Priority Based Static Advance Reservation Algorithm ($SARA_{SPB}$)

In this algorithm, first the scenarios are sorted and then sequentially processed. This sorting is based on the earliest average start time of the scenario's requests. If two scenarios have the same value, the one requiring more resources is chosen. As can be seen in Algorithm 1, the network resource usage, the requests information and the current scheduling are saved for each scenario.

```
input: scenarios' requests, network infrastructure
sortedQueue ← AverageStartSort(all scenarios);
for (scenario ∈ sortedQueue) do
    Set scenario status as Pending;
    currentstate ← Save the current system state;
    Prioritization(scenario's requests);
    sysReqList.Add(scenario's requests);
    feasible ← TimeSlot(sysReqList);
    if (feasible) then
        Update the scheduling;
        Set scenario status as Admitted;
    else
        Set current system state to CurrentState;
        Set scenario status as Rejected;
    end
end
```

Algorithm 1: The SARA Sequential Priority Based (SPB) algorithm.

Then each scenario in the sorted list is processed as follows. The prioritization algorithm is another component which assigns priorities to the scenario's requests. In the prioritization step two factors play a role: the estimated hard deadline and the volume. Since the deadline may not be specified for all the requests, the hard deadline (i.e. the latest possible deadline) for those with no specific deadline should be estimated. This time is calculated by assuming that all requests on which the request depends use the entire network at once. This gives the latest possible deadline for the request. In the prioritization algorithm, the main factor is the estimated hard deadline: the sooner the deadline, the higher the priority. The second factor, volume, comes into consideration only when the hard deadlines are equal, the higher the demand, the higher the priority.

Then the scenario's requests are added to the list of system requests and this list is given to the TimeSlot algorithm. Based on the result of TimeSlot algorithm, SPB decides to admit or reject the scenario. If the TimeSlot algorithm achieves a feasible schedule, the previous reschedule is updated, otherwise the algorithm has to backtrack to the previous feasible situation.

The TimeSlot algorithm, which is shown in Algorithm 2, iterates over the time slots and consists of 5 sub-algorithms for each time interval.

1. TimeSlotRequests: First, the algorithm has to determine which unserved requests can be served in the current time slot. For independent requests the algorithm looks at the start time. If the current interval is greater or equal the request start time, these requests are eligible to be added to the list of current requests. For requests with start time dependencies, the algorithm checks whether the requests on which this request depends are finished or not. If all the requests on which the request depends are fulfilled, this request can be started. Then it will be added to the

```
input: sysReqList, timeSlotGraphs
for (t ∈ Time Intervals) do
    currentRequests←TimeSlotRequests(t,sysReqList);
    if (currentRequests ≠∅) then
        Limit(currentRequests);
        sortedList ← Sort(currentRequests);
        reservation ← BWallocation(sortedList);
        if
        (!UpdateAndCheckFeasibility(reservation))
        then
            return false;
        end
    end
end
return true;
```

Algorithm 2: The TimeSlot algorithm.

list of current requests.

2. Limit: In order to avoid the extra reservation for the requests, a limitation needs to be defined for each request. The limit for the video streams is their required demand, because their demand is fixed and non-variable. The limit of file-based requests is their residual demand which is modified whenever a part of video file is transferred.
3. Sorting: In this step the requests of different scenarios selected in the previous step are sorted based on their priorities.
4. BWallocation: The most important sub-algorithm in TimeSlot is the bandwidth allocation algorithm. We have defined two different bandwidth allocation algorithms for video streams and video files. These algorithms are presented later in detail.
5. UpdateAndCheckFeasibility: based on the provided result of the previous step, BWallocation, and by calculating the residual demands, the requests requirements are updated and the feasibility of the results is checked. If the hard deadline of a request is reached, but part of the request has not been transferred yet and the residual demand is not zero, the hard deadline has not been met. In this step rescheduling is infeasible and the TimeSlot algorithm returns false.

The Bwallocation algorithm, which is shown in Algorithm 4, determines the maximum possible bitrate and associated paths for the requests. This algorithm first assigns cost to the network links using the CostAllocation algorithm. There are several approaches to find a path between source and destination of a flow. Since in advance reservation the other flows and all their specifications are often known, the most logical way is to take their preferred deployment into account. As a result in this step the algorithm tries to find the most desired paths and give them the highest cost. The cost of each link is the sum of desirability of this link for all requests. To find how important

```
input: sortedReqList
costAllocation(Links);
for (req ∈ sortedReqList) do
    if (req is FB) then
        reservation ← BWallocationFB(req);
    else
        reservation ← BWallocationVS(req);
    end
end
return reservation;
```

Algorithm 3: The BWallocation algorithm.

a link is for a request, the maximum flow (maxflow) from source to sink of each request is determined. Having the maxflow and the utilization per link, the desirability of the link is measured by dividing the link utilization by the maxflow. The maxflow is measured using the Edmonds-Karp maximum flow approach [27].

Then according to the type of the request either BWallocationFB or BWallocationVS algorithm, shown in Algorithm 4 and Algorithm 5 respectively, is invoked. The BWallocationFB algorithm is in charge of the FB requests and is based on maxflow and least-cost path algorithms. In order to transfer the files as soon as possible, first the maxflow is applied based on the Edmonds-Karp algorithm. If the maxflow is lower than the request limit, all the maxflow paths are reserved for this request. Otherwise, the algorithm forms a graph out of the maxflow paths and the k-shortest path is the second alternative. In this step finding the least-cost path is repeated till the total bandwidth offered by the paths is sufficient for the request. The shortest path applied here is a modified version of the Dijkstra shortest path algorithm [28] in which the cost of the links are taken into account instead of path length.

The BWallocationVS algorithm deals with video stream requests. This algorithm is partially similar to the second part of the BWallocationFB algorithm where the maxFlow is higher than the request limit. The algorithm iteratively looks for the least cost path on the whole graph and sums up the minimum available bandwidth of the paths. In each step the network capacities are modified and the path is reserved. These steps are repeated over the residual graph while the total bandwidth provided by the paths fulfills the request demand. If no more paths are left, the request could not be served and feasibility of rescheduling is false.

5.3.2. Sequential Priority Based Dynamic Advance Reservation Algorithm ($DARA_{SPB}$)

The $DARA_{SPB}$ is invoked several times whenever new scenarios are submitted to the reservation system. The dynamic approach is partly similar to the $SARA_{SPB}$. However in addition to the static approach and same as dynamic ILP-based algorithm, it has to update the previously admitted requests' demands based on whether the request is scheduled, is finished or is in progress.

```
input: an FB request
maxFlow ← EdmondsKarp.getMaxFlow(graph);
if (maxFlow = 0) then
    Return;
else if (maxFlow ≤ Limit(req)) then
    reservation(req, maxFlowPath);
else
    graph(maxFlowPath);
    path ← LeastCostPath(graph);
    while (path ≠ ∅) do
        minBW ← minBandwidth(path);
        flow ← flow+minBW;
        if (flow ≥ Limit(req)) then
            minBW ← minBW−(flow−Limit(req));
            reservation(req, path, minBW);
            update the residual graph(minBW,
            path);
            feasibility ← true;
            return reservation;
        else
            reservation(req, path, minBW);
            update the residual graph(minBW,
            path);
        end
        path ← LeastCostPath(graph);
    end
end
```

Algorithm 4: The BWallocationFB algorithm for file based requests.

```
input: a VS request
path ← LeastCostPath(graph);
while (path ≠ ∅) do
    minBW ← minBandwidth(path);
    flow ← flow+minBW;
    if (flow ≥ Limit(req)) then
        minBW ← minBW−(flow−Limit(req));
        reservation(req, path, minBW);
        update the residual graph(minBW, path);
        feasibility ← true;
        return reservation;
    else
        reservation(req, path, minBW);
        update the residual graph(minBW, path);
    end
    path ← LeastCostPath(graph);
end
feasibility ← false;
```

Algorithm 5: The BWallocationVS algorithm for video streaming requests.

6. Experimental Results

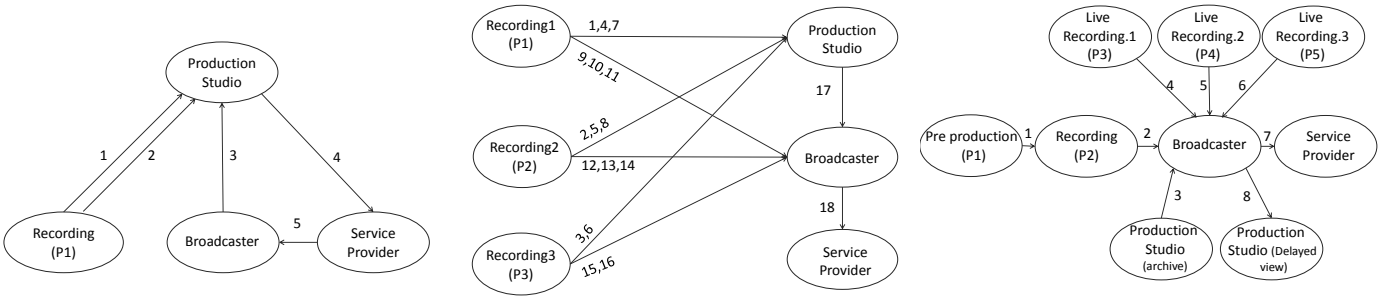
In this section, we first evaluate the SPB static and dynamic algorithms by comparing their performance with the optimal results provided by the ILP models as a benchmark. Then, we make an extensive evaluation on the SPB algorithms, determining the influence of the available bandwidth, the percentage of requests known in advance, the number of scenarios, and the time granularity.

6.1. Evaluation Setup

Based on interviews with several Belgian media production actors, including a broadcaster, service provider, and recording facility provider, a set of use case scenarios was defined that serve as a basis for the evaluation. Figure 3 depicts the interactions between actors in the three defined use cases. Use case 1 represents a soccer after-game discussion program and comprises 5 different file transfer requests. Use case 2 is a 30 minute infotainment show and consists of 18 file transfer requests. Finally, use case 3 is a news broadcast, consisting of 4 file transfer and 4 video streaming requests. Several instances of each use case are generated, based on randomized input parameters. A detailed overview of the randomized variables of each use case and its requests is shown in Table 3. The variable names used in the table header are defined in Table 2. $\#t_s^n dep.$ refers to the number of requests on which the start time of the request (i.e., t_s^n) depends. If a request does not depend on others, $t_s^n/dep.On$ is defined as the start time of the request, otherwise it points to those interdependent requests. The variables $\#t_s^e dep.$ and $t_s^e/dep.On$ are similarly defined for the end time of the request. The variable s used in the table represents the earliest time on which the file-based request could be started. In addition, st, d, and et deal with the streaming requests and refer to the start time of the broadcast on television, the deadline of the request to get started, and the end time of the request respectively

Because of the limited scalability of ILP-based algorithms, two different topologies are used for the evaluation of media production network reservation system. The smaller size, depicted in Figure 4, consists of media production actor sites, switches and bidirectional WAN links. This topology contains 12 nodes, 8 of which are devoted to different media production actors e.g. the production studio, broadcaster, service provider and recording locations. The 4 remaining nodes are the intermediate switches, connected in a full mesh topology. The larger test topology, shown in Figure 5, is only used to evaluate the performance of the more scalable SPB algorithms. This network is the well-known ATT North America topology which consists of 25 nodes and 56 links [29].

Each simulation run covers a 24 hour period. When using the SARA algorithm, it is assumed that all scenarios are known in advance. When using DARA some use case instances are assumed to be known only throughout the day, at least one hour before t_s^n of its earliest request. Throughout this section, DARA$XX\%[YY, ZZ]$ denotes that $XX\%$ of the use case instances are known at



(a) Use case 1: Soccer discussion program (b) Use case 2: Infotainment show (c) Use case 3: News broadcast

Figure 3: Interactions between media production actors in the three considered use case scenarios.

Table 3: Details of the use case requests

Use case 1	*Type*	s^n	d^n	$\#t_s^n dep.$	$t_s^n/dep.On$	$\#t_e^n dep.$	$t_e^n/dep.On$	i^n	b^n
Req1	r_f	$P1$	Production studio	0	$rand(s+1hrs, s+5hrs)$	1	$Req3$	$90min$	$200Mbps$
Req2	r_f	$P1$	Production studio	0	$rand(s, s+6hrs)$	1	$Req3$	$90min$	$200Mbps$
Req3	r_f	Broadcaster	Production studio	2	$Req1, 2$	1	$Req4$	$90min$	$200Mbps$
Req4	r_f	Production studio	Service provider	3	$Req1, 2, 3$	0	st	$180min$	$15Mbps$
Req5	r_f	Service provider	Broadcaster	0	$st + 3hrs$	0	$24hrs$	$180min$	$15Mbps$
$P1 = \text{rand}(\text{loc1}, \text{loc2}, \text{loc3}, \text{loc4}, \text{loc5})$; $s = \text{rand}(1, 9)$ hrs; $st = \text{rand}(17, 19)$ hrs									
Use case 2	*Type*	s^n	d^n	$\#t_s^n dep.$	$t_s^n/dep.On$	$\#t_e^n dep.$	$t_e^n/dep.On$	i^n	b^n
Req1,9	r_f	$P1$	Production Studio, Service Provider	0	$rand(s, 17hrs)$	1	Req17	$(50-60)min$	$200Mbps$
Req2,10	r_f	$P2$	Production Studio, Service Provider	0	$rand(s, 17hrs)$	1	Req17	$(50-60)min$	$200Mbps$
Req3,11	r_f	$P3$	Production Studio, Service Provider	0	$rand(s, 17hrs)$	1	Req17	$(50-60)min$	$200Mbps$
Req4,12	r_f	$P1$	Production Studio, Service Provider	0	$rand(s, 17hrs)$	1	Req17	$(50-60)min$	$200Mbps$
Req5,13	r_f	$P2$	Production Studio, Service Provider	0	$rand(s, 17hrs)$	1	Req17	$(50-60)min$	$200Mbps$
Req6,14	r_f	$P3$	Production Studio, Service Provider	0	$rand(s, 17hrs)$	1	Req17	$(50-60)min$	$200Mbps$
Req7,15	r_f	$P1$	Production Studio, Service Provider	0	$rand(s, 17hrs)$	1	Req17	$(50-60)min$	$200Mbps$
Req8,16	r_f	$P2$	Production Studio, Service Provider	0	$rand(s, 17hrs)$	1	Req17	$(50-60)min$	$200Mbps$
Req17	r_f	Production studio	Broadcaster	16	$Req1..16$	1	$Req18$	$60min$	$200Mbps$
Req18	r_f	Broadcaster	Service provider	1	$Req17$	0	st	$60min$	$15Mbps$
$P1, P2, P3 = \text{rand}(\text{loc1}, \text{loc2}, \text{loc3}, \text{loc4}, \text{loc5})$; $s = \text{rand}(1, 15)$ hrs; $st = \text{rand}(18, 22)$ hrs									
Use case 3	*Type*	s^n	d^n	$\#t_s^n dep.$	$t_s^n/dep.On$	$\#t_e^n dep.$	$t_e^n/dep.On$	i^n	b^n
Req1	r_f	$P1$	$P2$	0	$rand(s, 9hrs)$	1	$Req2$	$(30-50)min$	$200Mbps$
Req2	r_f	$P2$	Broadcaster	1	$Req1$	0	$rand(10, 12)hrs$	$(30-50)min$	$200Mbps$
Req3	r_f	Production studio	Broadcaster	0	$rand(s, 9hrs)$	0	$rand(10, 12)hrs$	$(30-50)min$	$200Mbps$
Req4	r_s	$P3$	Broadcaster	0	$rand(st, d)$	0	et	$(8-10)min$	$15Mbps$
Req5	r_s	$P4$	Broadcaster	0	$rand(st, d)$	0	et	$(8-10)min$	$15Mbps$
Req6	r_s	$P5$	Broadcaster	0	$rand(st, d)$	0	et	$(8-10)min$	$15Mbps$
Req7	r_s	Broadcaster	Service provider	0	st	0	st+0.5hrs	$30min$	$15Mbps$
Req8	r_f	Broadcaster	Production studio	0	st+0.5hrs	0	$24hrs$	$30min$	$15Mbps$
$P1, P2, P3, P4, P5 = \text{rand}(\text{loc1}, \text{loc2}, \text{loc3}, \text{loc4}, \text{loc5})$; $s = \text{rand}(1, 7)$ hrs; $st = \text{rand}(12, 16)$ hrs; $d = (st + 0.5 - i^n)$ hrs; if $(i^n < I)$ then $(et = T_s^n + 1)$ else $(et = T_s^n + i^n)$									

the start of the simulated day and the algorithm YY is used (i.e., ILP or SPB). ZZ is optional and refers to the time slot size in seconds, if the ZZ is not specified, the default time slot size of 3600 seconds has been used. Furthermore, the number of use case instances equals 20 (in total 209 requests) and 50 (in total 519 requests) for all the experiments with the smaller and larger topologies respectively. All results are averaged over 50 runs with different randomized inputs. Error bars denote the standard error.

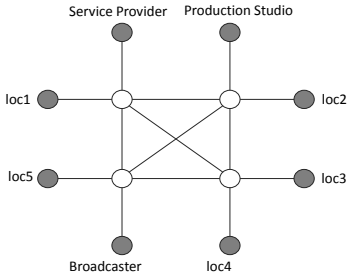


Figure 4: The smaller Media production network topology used in the evaluation.

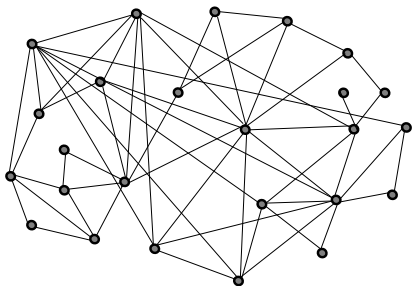


Figure 5: The Larger Media production network topology used in the evaluation. The ATT North America topology consists of 25 nodes and 56 links.

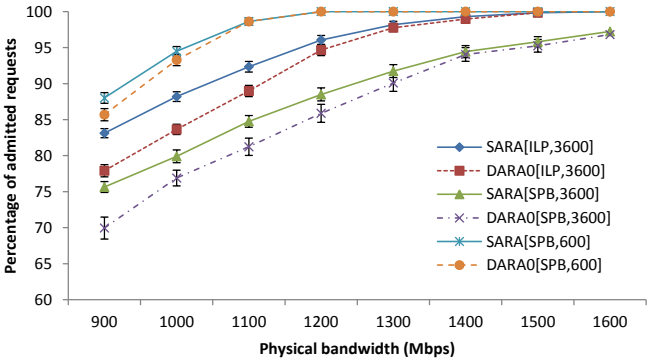


Figure 6: Impact of bandwidth capacity and percentage of known requests on admission rate.

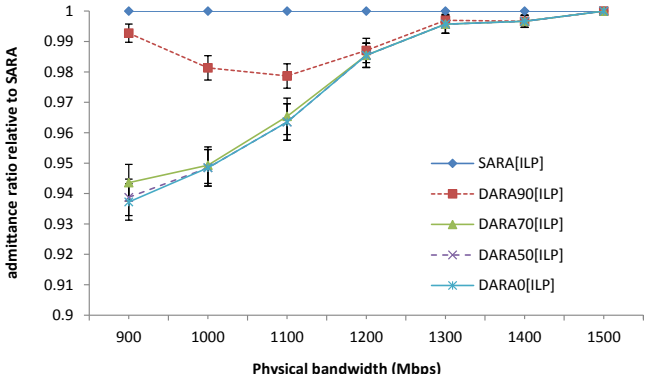


Figure 7: Impact of percentage of known requests on admission rate in ILP-based algorithms.

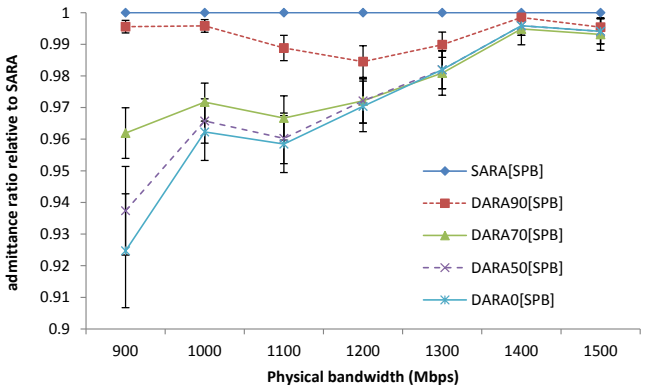


Figure 8: Impact of percentage of known requests on admission rate in SPB algorithms.

All algorithms in this section are implemented in Java 7. ILPs are solved using the IBM ILOG CPLEX 12.6 optimization software package.

6.2. Comparing the SPB algorithms to the ILP-based algorithms

Figure 6 compares the ILP solutions with the SPB algorithms, where for the latter either 1 hour or 10 minute time slot sizes are shown. In this evaluation, all or none of the requests are known in advance. First for the same time slot size of 1 hour, this figure shows that the result of the SPB algorithms is close to the optimal values. The performance of $SARA_{SPB}$ and $DARA_{SPB}$ are within 8.29% of $SARA_{ILP}$ and 8.78% of $DARA_{ILP}$ respectively. For a time slot size of 1 hour, the SPB algorithms reach to 100% admittance only when 2300 Mbps physical bandwidth is available. For the 6-times shorter time interval size (i.e. 10 minutes) and 1200 Mbps capacity, both the static and dynamic SPB algorithms admit all 20 scenarios, whereas the static ILP-based algorithm only achieve 96.05% on average.

Figure 7 and Figure 8 are provided to compare the performance of ILP-based and SPB algorithms respectively, showing the influence of the percentage of known requests in advance on the solutions. Knowing more requests gives more freedom to schedule them and makes it easier to determine the subset of requests to reject. Thus, static algorithms outperform the dynamic ones. Therefore, to have a clear distinction, in both figures the result are shown relative to the *SARA*. As expected, more known requests significantly increase the performance. The results show that when requests are not known at the start of the day, $SARA_{ILP}$ outperforms $DARA_{ILP}$ by up to 5.22% while when 90% are known this is reduced to 1.97% at most. For the SPB algorithms the same trend can be observed. The percentage of admitted requests in $DARA_{SPB}$ is within 5.7% and 1.37% of $SARA_{SPB}$ for the 0% and 90% of known requests respectively.

Furthermore, to have a comparison of the execution times Figure 9 is depicted. This figure compares the execution duration of the static ILP and SPB algorithms. As can be observed from this figure, the ILP is the most complex and the slowest, and the static SPB algorithm with the same time interval size is between 128 up to 520 times faster than the ILP solution.

6.3. Evaluation of the SPB algorithms

6.3.1. Impact of available bandwidth

We now assess the impact of physical network capacity on the SPB algorithm performance. Figure 10 compares the percentage of admitted requests of SARA to DARA,

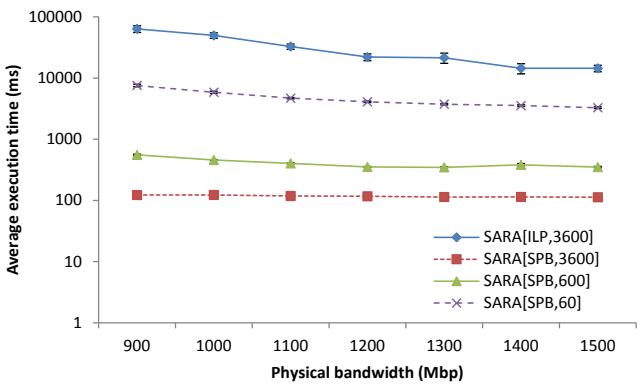


Figure 9: The execution time of SPB and ILP-based approaches.

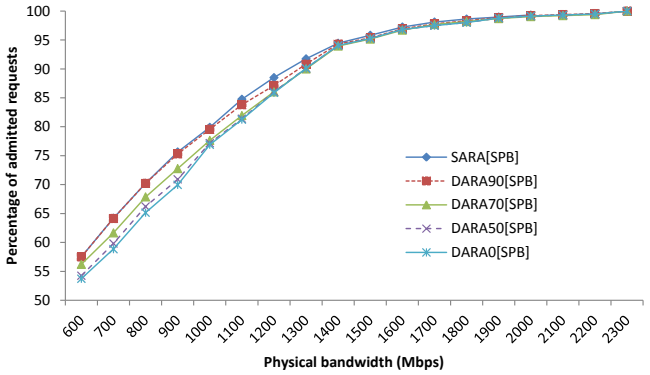


Figure 10: Impact of bandwidth capacity and percentage of known requests on admission rate in SPB algorithms for 12-node topology.

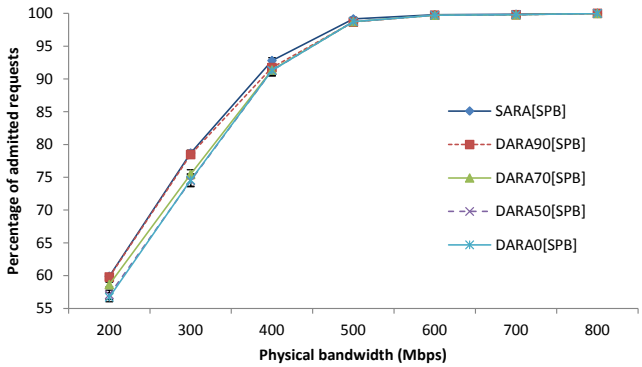


Figure 11: Impact of bandwidth capacity and percentage of known requests on admission rate in SPB algorithms for 25-node topology.

where for the latter various ranges of use case instances are known in advance. The network capacities vary from 600 up to 2300 Mbps and a time slot size of 1 hour is used. This figure shows that when no request is known in advance, SARA shows up to 5.7% higher acceptance rate compared with the dynamic approach.

In Figure 11 the same results are shown for the larger infrastructure. The available bandwidth varies from 200 up to 800 Mbps. As can be observed from this figure, when none of requests are known beforehand, the admittance ratio in *DARA* is within 4.1% of *SARA* and when 90% of requests are known the result is within 1.1%.

6.3.2. Impact of time slot granularity

Figures 12 and 13 study the impact of time slot granularity on SARA and DARA for the 12-node and 25-node topology respectively. For the smaller network the link capacity of 700 Mbps and for the larger topology the link capacity of 200 Mbps is used. As shown in both figures, the fine-grained experiment with shortest time slot size results in the best performance. However, although more granularity increases the performance of the algorithm, the complexity of the algorithm significantly increases as well. In the case of DARA, the algorithm is invoked several times throughout the day.

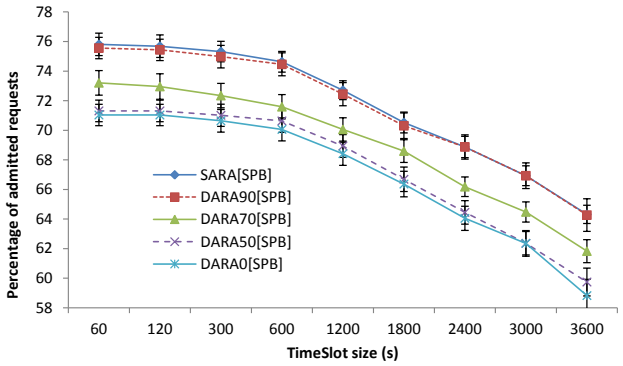


Figure 12: Impact of timeslot granularity on request admission rate for 12-node topology.

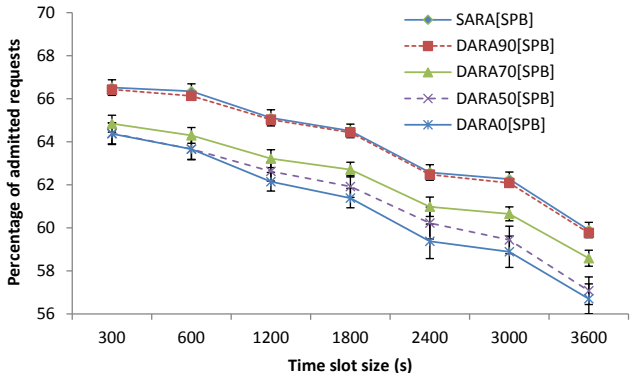


Figure 13: Impact of timeslot granularity on request admission rate for 25-node topology.

In Figure 12, we observe that an interval size of 60 seconds yields 11.5% better results than a size of 3600 seconds for SARA. However, the execution time of the algorithm also increases. For DARA, a similar trend is seen. Moreover, it should be noted that in case fewer requests are known in advance, the complexity of a single DARA algorithm invocation decreases significantly. This can be seen in Figures 14 and 15 for the smaller and the larger topologies respectively. For example in Figure 14, when none of requests are known and the size of time slot is 1 minute, the execution time is on average 18.33 times shorter than when all requests are known in advance. In the former case, the algorithm needs to be executed 19.6 times on average, while in the latter only once. Based on the results, a time slot size of 600 seconds optimises the trade-off between optimality and complexity. This interval size is not the most optimal value for all possible configurations, but is within 1.6% of the optimum in all evaluated cases.

6.3.3. Impact of network load

Figures 16, 17 and 18 compare the influence of number of use case instances and percentage of known requests. In Figure 16 and 17 the smaller topology with network capacity of 600 Mbps and 1 hour time slot size are used. The number of scenarios varies from 1 to 20. As can be seen in Figure 16 by increasing the demands, the percentage of ad-

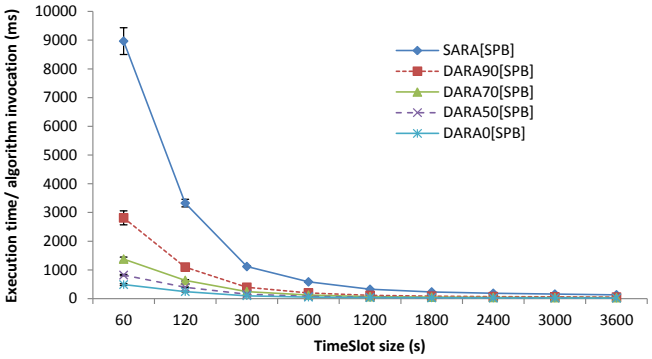


Figure 14: The average execution times of different time slot sizes and percentage of known requests for 12-node topology.

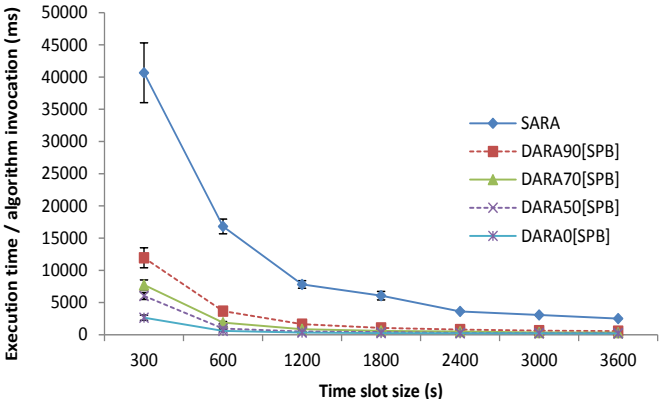


Figure 15: The average execution times of different time slot sizes and percentage of known requests for 25-node topology.

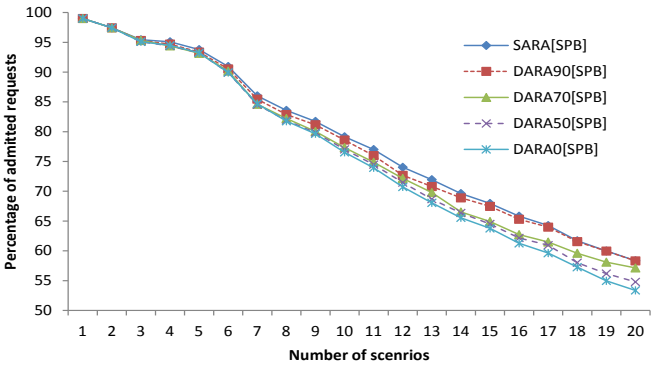


Figure 16: Impact of network load on request admission rate for 12-node topology.

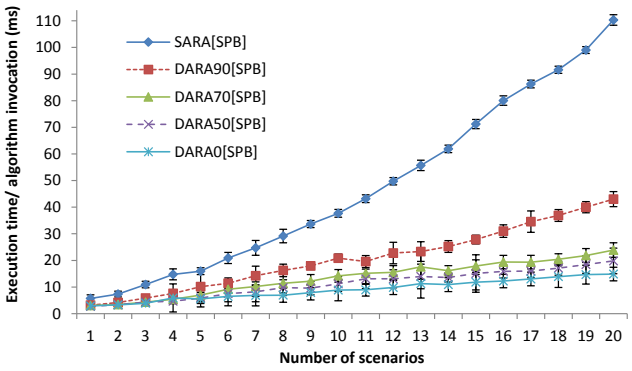


Figure 17: The execution times of different number of scenarios and percentage of known requests for 12-node topology.

mitted requests decreases. *SARA* outperforms *DARA* up to 5% when 0% of requests are known in advance, and up to 3.8% when 50% are known. As depicted in Figure 18 the same trend can be observed for the larger topology with 200 Mbps capacity. *SARA* yields up to 5.8% better results than *DARA*.

In Figure 17 the execution times for each algorithm invocation for a range of percentage of known requests is assessed. Based on this result the DARA is up to 7.4 times faster, when the number of scenarios is 20. However, the algorithm is invoked 9.48 times on average.

7. Conclusion

In this article, an optimal model and a set of novel scheduling algorithms were presented for advance bandwidth reservation in media production networks. Specifically the SPB approach is proposed to resolve the computational complexity associated with the optimal solutions. The bandwidth scheduling algorithms take the specific characteristics of media production processes into account, for example time-variable bandwidth reservation, flexible start times, request dependencies and splittable flows. In our approaches all four types of advance reservation requests are supported. Furthermore, the proposed algorithms operate in both offline and online manners. A detailed performance analysis is conducted to assess the viability of ILP-based and SPB solutions. The influence of the available bandwidth, the percentage of requests known in advance, the network load, the time granularity and the execution time have been evaluated.

Our evaluation showed that the SPB results at least within 8.78% of the optimal admittance rate. Also, when a significant portion of requests is known in advance, AR significantly increases bandwidth efficiency and request admittance. Concretely, in case all requests are known beforehand, request admittance of the optimal and heuristic solutions can be increased up to 5.22% and 5.7% respectively. Additionally, the results showed that time granularity increases algorithm accuracy and optimality in terms of request admittance. SPB can achieve higher scalability in terms of the size of physical network as well as time slot sizes. The size of time intervals can be fine-grained up to 1 minute. Comparing to the ILP-based approaches, the SPB algorithms offer lower operational overhead in terms of problem complexity and execution time.

Future work includes determining the impact on quality and performance of variable time intervals, and adding resilience to improve the robustness of the schedules generated by the advance reservation system.

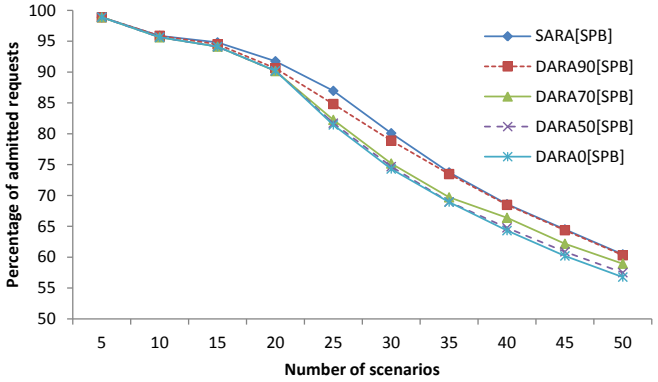


Figure 18: Impact of network load on request admission rate for 25-node topology.

Acknowledgment

The computational resources (Stevin Supercomputer Infrastructure) and services used in this work were provided by the VSC (Flemish Supercomputer Center), funded by Ghent University, the Hercules Foundation and the Flemish Government – department EWI. The research leading to these results was performed within the context of ICON MECaNO. It is a project co-funded by iMinds, a digital research institute founded by the Flemish Government. Project partners are SDNsquare, Limecraft, VideoHouse, Alcatel-Lucent, and VRT, with project support from IWT under grant agreement no. 130646.

References

[1] Varvarigos EM, Sourlas V, Christodoulopoulos K. Routing and scheduling connections in networks that support advance reservations. Computer Networks 2008;52(15):2988–3006.

[2] Nahrstedt K, Steinmetz R. Resource management in networked multimedia systems. Computer 1995;28(5):52–63.

[3] Charbonneau N, Vokkarane VM. A survey of advance reservation routing and wavelength assignment in wavelength-routed wdm networks. Communications Surveys & Tutorials, IEEE 2012;14(4):1037–64.

[4] Hall A, Hippler S, Skutella M. Multicommodity flows over time: Efficient algorithms and complexity. In: Baeten J, Lenstra J, Parrow J, Woeginger G, editors. Automata, Languages and Programming; vol. 2719 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg. ISBN 978-3-540-40493-4; 2003, p. 397–409. URL: `http://dx.doi.org/10.1007/3-540-45061-0_33`. doi:`10.1007/3-540-45061-0_33`.

[5] Guok C, Engineer EN, Robertson D. Esnet on-demand secure circuits and advance reservation system (oscars). Internet2 Joint 2006;.

[6] Gibbard B, Katramatos D, Yu D. Terapaths: end-to-end network path qos configuration using cross-domain reservation negotiation. In: Broadband Communications, Networks and Systems, 2006. BROADNETS 2006. 3rd International Conference on. IEEE; 2006, p. 1–9.

[7] Gu J, Katramatos D, Liu X, Natarajan V, Shoshani A, Sim A, et al. Stornet: Integrated dynamic storage and network resource provisioning and management for automated data transfers. In: Journal of Physics: Conference Series; vol. 331. IOP Publishing; 2011, p. 012002.

[8] Gu J, Katramatos D, Liu X, Natarajan V, Shoshani A, Sim A, et al. Stornet: Co-scheduling of end-to-end bandwidth reservation on storage and network systems for high-performance data transfers. In: Computer Communications Workshops (INFOCOM WKSHPS), IEEE Conference on. IEEE; 2011, p. 121–6.

[9] Xie C, Alazemi H, Ghani N. Rerouting in advance reservation networks. Computer Communications 2012;35(12):1411–21.

[10] Alazemi H, Xu F, Xie C, Ghani N. Advance reservation in distributed multi-domain networks. IEEE Systems Journal 2013;.

[11] Naikstam S, Figueira S. Elastic reservations for efficient bandwidth utilization in lambdagrids. Future Generation Computer Systems 2007;23(1):1–22.

[12] Burchard LO, Heiss HU, De Rose C. Performance issues of bandwidth reservations for grid computing. In: Symposium on Computer Architecture and High Performance Computing. 2003, p. 82–90.

[13] Sharma S, Katramatos D, Yu D, Shi L. Design and implementation of an intelligent end-to-end network qos system. In: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis. SC '12; Los Alamitos, CA, USA: IEEE Computer Society Press. ISBN 978-1-4673-0804-5; 2012, p. 68:1–68:11. URL: `http://dl.acm.org/citation.cfm?id=2388996.2389089`.

[14] Rajah K, Ranka S, Xia Y. Advance reservations and scheduling for bulk transfers in research networks. IEEE Trans Parallel Distrib Syst 2009;20(11):1682–97. URL: `http://dx.doi.org/10.1109/TPDS.2008.250`. doi:`10.1109/TPDS.2008.250`.

[15] Ahuja RK, Magnanti TL, Orlin JB. Network flows. Tech. Rep.; DTIC Document; 1988.

[16] Masri H, Krichen S, Guitouni A. A multi-start variable neighborhood search for solving the single path multicommodity flow problem. Applied Mathematics and Computation 2015;251:132–42.

[17] Ouorou A, Mahey P, Vial JP. A survey of algorithms for convex multicommodity flow problems. Management science 2000;46(1):126–47.

[18] Kennington JL. A survey of linear cost multicommodity network flows. Operations Research 1978;26(2):209–36.

[19] Ford Jr LR, Fulkerson DR. Constructing maximal dynamic flows from static flows. Operations research 1958;6(3):419–33.

[20] Ford L, Fulkerson DR. Flows in networks; vol. 1962. Princeton Princeton University Press; 1962.

[21] Fleischer L, Skutella M. Quickest flows over time. SIAM Journal on Computing 2007;36(6):1600–30.

[22] Chen J, Chan SH, Li VO. Multipath routing for video delivery over bandwidth-limited networks. Selected Areas in Communications, IEEE Journal on 2004;22(10):1920–32.

[23] Balman M, Chaniotakis E, Shoshani A, Sim A. A flexible reservation algorithm for advance network provisioning. In: High Performance Computing, Networking, Storage and Analysis (SC), 2010 International Conference for. IEEE; 2010, p. 1–11.

[24] Barshan M, Moens H, Famaey J, De Turck F. Algorithms for advance bandwidth reservation in media production networks. In: the integrated network management. IM2015. IFIP/IEEE International symposium on. IEEE; 2015,To appear.

[25] Zheng J, Mouftah HT. Routing and wavelength assignment for advance reservation in wavelength-routed wdm optical networks. In: Communications, 2002. ICC 2002. IEEE International Conference on; vol. 5. IEEE; 2002, p. 2722–6.

[26] He E, Wang X, Vishwanath V, Leigh J. Cam03-6: Ar-pin/pdc: Flexible advance reservation of intradomain and interdomain lightpaths. In: Global Telecommunications Conference, 2006. GLOBECOM'06. IEEE. IEEE; 2006, p. 1–6.

[27] Edmonds J, Karp RM. Theoretical improvements in algorithmic efficiency for network flow problems. Journal of the ACM (JACM) 1972;19(2):248–64.

[28] Cormen T. Introduction to Algorithms. MIT Press; 2009. ISBN 9780072970548. URL: `http://books.google.be/books?id=Jwr8jwEACAAJ`.

[29] Knight S, Nguyen HX, Falkner N, Bowden R, Roughan M. The internet topology zoo. Selected Areas in Communications, IEEE Journal on 2011;29(9):1765–75.