Algorithms for Advance Bandwidth Reservation in Media Production Networks

Maryam Barshan[1], Hendrik Moens[1], Jeroen Famaey[2], Filip De Turck[1]

[1]Department of Information Technology, Ghent University – iMinds
Gaston Crommenlaan 8/201, B-9050 Gent, Belgium
[2]Department of Mathematics and Computer Science, University of Antwerp – iMinds
Middelheimlaan 1, 2020 Antwerpen, Belgium
Email: maryam.barshan@intec.ugent.be

***Abstract*—Media production generally requires many geographically distributed actors (e.g., production houses, broadcasters, advertisers) to exchange huge amounts of raw video and audio data. Traditional distribution techniques, such as dedicated point-to-point optical links, are highly inefficient in terms of installation time and cost. To improve efficiency, shared media production networks that connect all involved actors over a large geographical area, are currently being deployed. The traffic in such networks is often predictable, as the timing and bandwidth requirements of data transfers are generally known hours or even days in advance. As such, the use of advance bandwidth reservation (AR) can greatly increase resource utilization and cost efficiency. In this paper, we propose an Integer Linear Programming formulation of the bandwidth scheduling problem, which takes into account the specific characteristics of media production networks, is presented. Two novel optimization algorithms based on this model are thoroughly evaluated and compared by means of in-depth simulation results.**

***Index Terms*—Advance bandwidth reservation, media production network, video streaming, deadline-aware scheduling.**

I. Introduction

The production of media content is a complicated process involving a wide range of actors, such as production houses, facility providers, broadcasters and advertisers. Throughout the production process, huge amounts of raw video and audio content need to be transferred between geographically distributed locations (e.g., from an external filming location, to the production studio, to the broadcaster). Currently, the distribution of media production content is generally performed by either people transporting the content on a physical storage medium or over dedicated point-to-point high-speed optical links. Clearly, these are highly inefficient and costly methods. Connecting the different actors involved in the media production process to a shared network substrate would greatly reduce capital expenditures and increase network resource utilization. Currently, such shared media production networks, connecting many actors across a large geographical area (e.g., a country), are being deployed.

A key characteristic of traffic in media production networks, is its predictability. The timing, locality and bandwidth requirements of data transfers are often known hours and sometimes even days in advance. As such, the use of advance bandwidth reservation (AR) techniques [1] would result in greatly increased bandwidth utilization and reduced costs. In AR networks, users submit requests for future data transfers, generally encompassing a start time (either immediately or at some point in the future), a deadline, and total data transfer size (or rate). Subsequently, a scheduling algorithm allocates the necessary bandwidth resources to ensure that all admitted requests finish before their specified deadline, while admitting as many requests as possible. Clearly, AR has several advantages for next generation media production networks. It allows network operators to better plan resource usage, leading to greatly increased resource utilization and guaranteed Quality of Service (QoS).

In order to implement AR scheduling, the underlying network has to support bandwidth reservations. Current research on the topic mostly focuses on optical networks in combination with wavelength division multiplexing [1]. However, Software Defined Networking (SDN) techniques, such as OpenFlow, provide high-level bandwidth reservation abstractions, hiding the details of the underlying physical mechanisms. As a first contribution, this paper presents an AR-based media production platform that is generic in terms of the underlying reservation techniques (e.g., wavelength- or time-based multiplexing), which can be used in conjunction with SDN.

As a second contribution, 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 [1]. Second, in media production networks, multiple requests may depend on each other (e.g., the start time of sending edited material to the broadcaster depends on the end time of sending recorded material to the production office). Until now to the authors' knowledge, this aspect remained unexplored. Third, it should be possible to split requests over multiple paths, in order to further optimize bandwidth utilization. We present an Integer Linear Programming (ILP) model to solve this variant of the AR scheduling problem. Based on this model, two scheduling algorithms are presented. The Static Advance Reservation

Algorithm (SARA) assumes all requests are known at the start of the reservation period (e.g., at the start of the day). In contrast, the Dynamic Advance Reservation Algorithm (DARA) supports rescheduling in order to incorporate new requests at runtime. We provide a thorough analysis of both algorithms based on in-depth simulation results. They are compared and the impact of their parameters on the solution is evaluated.

The remainder of this paper is structured as follows. In Section II, we discuss related work. Section III describes the architecture and components of our proposed SDN-based media production network. In Section IV, the concepts, assumptions and AR scheduling problem for media production networks are detailed. Subsequently, the designed AR scheduling algorithms are described in Section V. Section VI provides simulation results, comparing the proposed algorithms. Finally, Section VII concludes the paper.

II. Related work

AR has been a popular topic of study in the area of optical networks with WDM throughout the last decade. Recently, Charbonneau et al. surveyed the state of the art in WDM-based AR, and classified existing approaches based on a novel taxonomy [1]. Three types of AR scheduling algorithms were identified; STSD demands specify a start time and a duration (or deadline), STUD demands specify a start time but no duration, and UTSD demands specify a duration but no start time. STSD, on which most work to date focuses, can be further classified into fixed and flexible start time. In the former case, the bandwidth reservation of the request needs to start at the specified start time or be blocked. In the latter case, the reservation needs to start within a window of possible start times. Another variation is referred to as elastic reservations, where the bandwidth allocated to a request may vary over time. The algorithms we propose for media production networks can be classified as STSD with flexible start time and elastic reservations. According to Charbonneau et al. only two AR scheduling algorithms have been proposed that support elastic reservations [2], [3]. However, they both assume a fixed start time.

Current research on AR scheduling mostly focuses on rescheduling [4], [5], [6], multi-domain reservations [7], and real-life deployments [8], [9], [10], [11]. Rajah et al. [4] propose an AR scheduling algorithm for transferring large volumes of data in e-science networks. The algorithm performs an admission control and scheduling step. During admission control, active requests may be rerouted in order to increase request admission. For the scheduling step, two alternative objectives are evaluated: quick finish (i.e., schedule all requests as soon as possible) and load balancing (i.e., minimize maximum link load). One of the objectives used in our work is inspired by the quick finish approach. Xie et al. [5] study the rescheduling problem in more detail. They propose an ILP-based model, as well as a fast heuristic for re-routing flows in AR networks in order to maximize admittance of new requests. Their ILP model forms a starting point for the model presented in this paper.

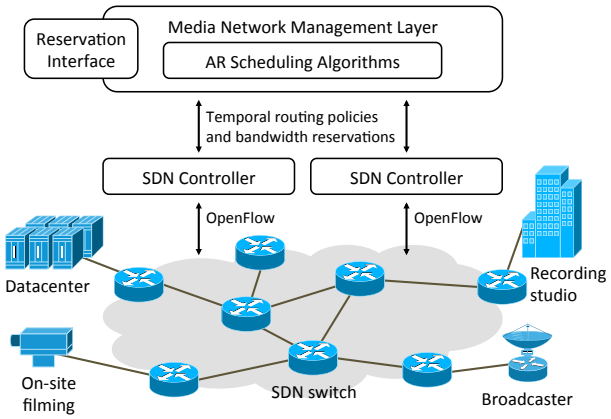


Fig. 1: Media production network architecture and components

In conclusion, the work presented in this paper differs from state of the art research in three ways. First, to our knowledge, we are the first to study the STSD problem with both flexible start times and elastic reservations [1]. Second, in contrast to most existing work, we employ SDN as an enabler for a generalized AR scheduling approach, rather than one specifically aimed at optical WDM technologies. Third, state of the art research does not consider dependencies among requests, which are important in the case of media production networks, and are explicitly incorporated in our model.

III. Media production network architecture

The envisioned media production network is depicted in Figure 1, which we assume to be SDN-based. 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 (i.e., unicast, multicast or broadcast), 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. Note that in case of file transfers, the reserved resources for a request may vary over time, as long as the delivery deadline is satisfied. In case the deadline of a request cannot be guaranteed, the reservation interface rejects it. When multiple requests 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. For performing the configuration, a protocol such as OpenFlow can be used. The remainder of this paper focuses on the AR scheduling algorithms.

IV. AR 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. 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.

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 finished.

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 I lists the notations which has been used to define the model.

TABLE I: Symbols and notations used in the formal models

Variable	Description
N	Physical nodes set.
E	Physical links set.
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).

A. 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 dependent 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.

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

B. Objective function

We consider two different objective functions which we refer to as maxA and ASAP. By the former, shown in Expression 1 we aim at maximizing the rate of request admittance which is determined by summing the A^n variables.

$$\max \sum_{r^n \in R} A^n \tag{1}$$

The alternative ASAP objective function, shown in Expression 2 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.

$$\max \sum_{r^n \in R} A^n + \frac{\sum_{r^n \in R} \sum_{e \in E_{s^n}^{out}} \sum_{k \in [t_s^n, t_e^n]} \frac{\beta^{n,e,k}}{k}}{\sum_{r^n \in R} \sum_{e \in E_{s^n}^{out}} \sum_{k \in [t_s^n, t_e^n]} \frac{B^e}{k}} \quad (2)$$

C. 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 3, 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.

$$\sum_{r^n \in R} \beta^{n,e,k} \leq B^e \qquad \forall e \in E, \forall k \in [t_s^{min}, t_e^{max}] \quad (3)$$

All network nodes that are not source or sink of a flow are subject to a flow conservation constraint, shown in Expression 4, which 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, which is shown in Expression 5. For these requests, the amount of data transferred can vary between timeslots. Video streaming requests are handled behave 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 6. To minimize the occurrence of loops within the network, constraints preventing incoming flow in the source node and outgoing flow in the sink node is added. These constraints are shown in Expressions 7 and 8.

$$\sum_{e \in E_v^{out}} \beta^{n,e,k} = \sum_{e \in E_v^{in}} \beta^{n,e,k} \quad (4)$$
$$\forall r^n \in R, \forall k \in [t_s^{min}, t_e^{max}], \{\forall v \in N | v \notin \{s^n, d^n\}\}$$

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

$$\sum_{e \in E_{s^n}^{out}} \beta^{n,e,k} = b^n \times A^n \quad \forall r_s^n \in R_s, \forall k \in [t_s^{min}, t_e^{max}] \quad (6)$$

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

$$\sum_{e \in E_{d^n}^{out}} \beta^{n,e,k} = 0 \qquad \forall r^n \in R, \forall k \in [t_s^{min}, t_e^{max}] \quad (8)$$

D. Interdependent requests

Start and end times of requests may either be input variables or decision variables. Dependencies between different requests are handled by Expressions 9, 10, 11, 12, 13 and, 14 and 15. First, Expression 9 ensures either all or none of the requests of a scenario get admitted.

$$A^n = A^s \quad \forall r^n \in R \quad (9)$$

Expression 10 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 $\beta^{n,e,k} = 0$.

$$\beta^{n,e,k} \leq SU^{n,k} \times B^e \quad \forall e \in E, \forall k \in [t_s^{min}, t_e^{max}], \forall r^n \in R \quad (10)$$

If the start time is known and predefined as an input variable, then Expression 11 ensures that no bandwidth is dedicated to request r^n before t_s^n.

$$\beta^{n,e,k} = 0 \qquad \forall e \in E, \forall r^n \in R, \forall k \in [t_s^{min}, t_s^n) \quad (11)$$

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 12 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 than be added as shown in Expression 13, which ensures that the request n is started only when all the requests on which request n depends are finished.

$$t_s^n \leq k + (1 - SU^{n,k}) \times t_e^{max} \quad \forall r^n \in R, \forall k \in [t_s^{min}, t_e^{max}] \quad (12)$$

$$t_s^n \geq t_e^{n'} + 1 \quad \{\forall r^n \in R | r^n \textit{ depends on } r^{n'}\} \quad (13)$$

When the end time is an input variable, then Expression 14 ensures that no bandwidth is dedicated to request n after t_e^n.

$$\beta^{n,e,k} = 0 \qquad \forall e \in E, \forall r^n \in R, \forall k \in (t_e^n, t_e^{max}] \quad (14)$$

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 is constraint shown in Expression 15.

$$t_e^n \geq k - (1 - SU^{n,k}) \times t_e^{max} \quad \forall r^n \in R, \forall k \in [t_s^{min}, t_e^{max}] \quad (15)$$

E. 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 at various times in on-line manner. 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 16) is therefore added to ensure that previously admitted requests remain accepted.

$$A^s = 1 \quad \forall r^n \in R_o \quad (16)$$

V. ILP BASED ADVANCE RESERVATION ALGORITHMS

In this section we define two algorithms based on the model presented in the previous section. The Static Advance Reservation Algorithm (SARA) can be used to generate a schedule when all requests are known before execution, and is based on the static model. When not all requests are known from the start, and new ones are added throughout the day, the Dynamic Advance Reservation Algorithm (DARA), which makes use of the on-line version of the model, can be used. The proposed algorithms are implemented with both the maxA and ASAP objective functions, resulting in four algorithm variants: $SARA_{maxA}$, $SARA_{ASAP}$, $DARA_{maxA}$ and $DARA_{ASAP}$. Both algorithms are implemented in Java 1.7 and make use of Integer Linear Programmings (ILPs) that are solved using the IBM ILOG CPLEX Optimization software package.

A. Static Advance Reservation Algorithm (SARA)

The SARA algorithm is based on the formal model that was presented in the previous section, which was implemented as an ILP using CPLEX. In this algorithm we assume that all the scenario arrivals are known beforehand, which results in an optimal schedule.

Having just 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, 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. Futhermore, 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 the latter solution. Therefore, we use a post processing algorithm after the ILP optimization. During this post-processing phase, we look for cycles in each reserved path and to get rid of extra reservations in each cycle the reserved bandwidths are modified.

B. Dynamic Advance Reservation Algorithm (DARA)

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. The DARA invokes the ILP formulation of the model multiple times whenever new scenarios arrive. When this happens, 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. We assume that new incoming scenarios have lower priority as the previous requests are already admitted and rejecting them violate the agreed SLA.

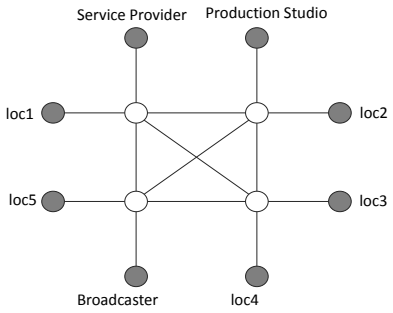


Fig. 2: Media production network topology used in the evaluation

In the DARA algorithm, an initial schedule is generated using the static model, which then iteratively updated using the on-line model as new requests 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. Therefore, requests are divided in three categories based on their progress:

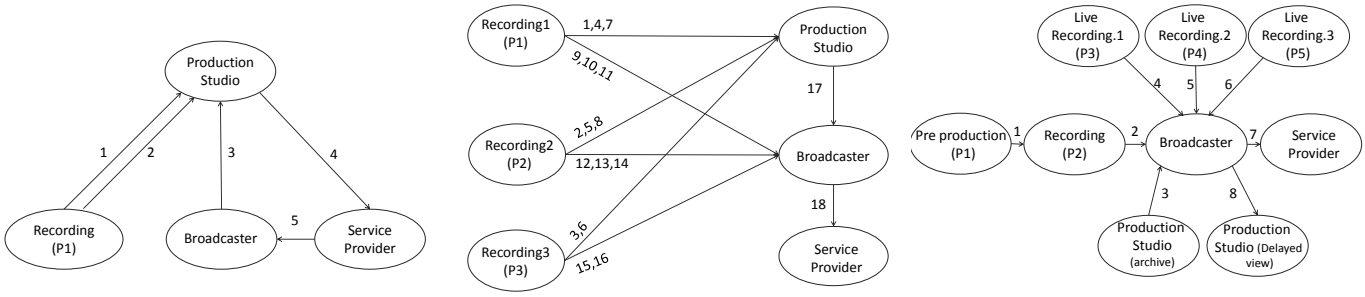
- **Scheduled:** When a request is scheduled, it will start to execute during some time slot in the future. As the request is not yet running during the trigger point, no special considerations are needed.
- **Finished:** A request is considered finished when it has finished executing at the time of 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 the 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.

VI. RESULTS & DISCUSSION

This section evaluates the proposed AR scheduling algorithms. The DARA algorithm and its two proposed objective functions are compared using the optimal SARA algorithm as a benchmark. The influence of the available bandwidth, the percentage of requests known in advance, and the time granularity are assessed.

A. Evaluation Setup

The media production network topology used for the evaluation, depicted in Figure 2, contains 12 nodes. The network consists of media production actor sites, SDN-enabled switches and bidirectional WAN links. 8 out of 12 nodes are devoted to



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

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

different media production actors e.g. the production studio, broadcaster, service provider and recording locations. The 4 remaining nodes are the intermediate SDN switches, connected in a full mesh topology. It should be noted that this topology is chosen because of the limited scalability of ILP-based algorithms. In future work, by proposing scalable near-optimal heuristics, higher scalability will be achieved.

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 II. The variable names used in the table header are mostly defined in Table I. $\#t_s^n dep.$ 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 deals with the streaming requests and refers 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

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*] denotes that *XX%* of the use case instances are known at the start of the simulated day and the objective *YY* is used (i.e., ASAP or MaxA). We found that both ASAP and MaxA objective functions yield identical results for the SARA algorithm, which is why this algorithm is denoted by SARA without mention of the used objective function. All results are averaged over 50 runs with different randomized inputs, error bars denote the standard error.

B. Impact of available bandwidth

1) Scenario: The media network infrastructure has been configured for different available bandwidths to investigate the impact of network capacities on the performance of our algorithms. Bandwidth capacity per link varies from $900Mbps$ to $1.5Gbps$. The number of use case instances equals 20, of which 7, 7 and 6 are of use case 1, use case 2 and use case 3 respectively. This results in a total of 209 requests. A fixed time interval granularity of 1 hour is used.

2) Results: Figure 4 compares the percentage of admitted requests of SARA to the maxA and the ASAP objective functions of DARA, where for the latter either 0% or 50% of use case instances are known in advance. As expected, $SARA$ outperforms $DARA$, as knowing all requests gives more freedom to schedule everything, making it easier to determine the subset of requests to reject. From the figure, we can conclude that the ASAP outperforms MaxA in a dynamic scenario, as it schedules requests as soon as possible, freeing up more resources for requests that may arrive in the future.

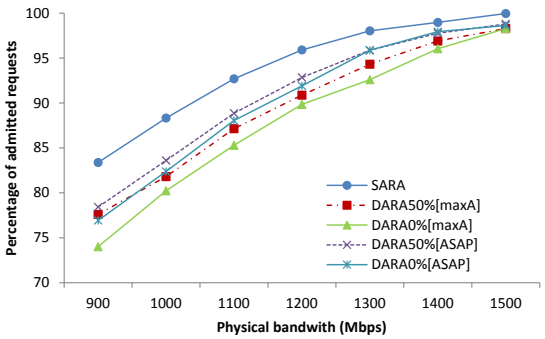


Fig. 4: Impact of bandwidth capacity on request admission rate, comparing objective functions maxA and ASAP

TABLE II: 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	0	$Req1, 2$	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	$24hrs$	$180min$	$15Mbps$
$P1 =$ rand(loc1, loc2, loc3, loc4, loc5); $s =$ rand$(1, 9)$ hrs; $st =$ 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	$60min$	$15Mbps$
$P1, P2, P3 =$ rand(loc1, loc2, loc3, loc4, loc5); $s =$ rand$(1, 15)$ hrs; $st =$ 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	$24hrs$	$30min$	$15Mbps$
$P1, P2, P3, P4, P5 =$ rand(loc1, loc2, loc3, loc4, loc5); $s =$ rand$(1, 7)$ hrs; $st =$ rand$(12, 16)$ hrs; $d = (st + 0.5 - i^n)$ hrs; if $(i^n < I)$ then $(et = T_s^n + 1)$ else $(et = T_s^n + i^n)$									

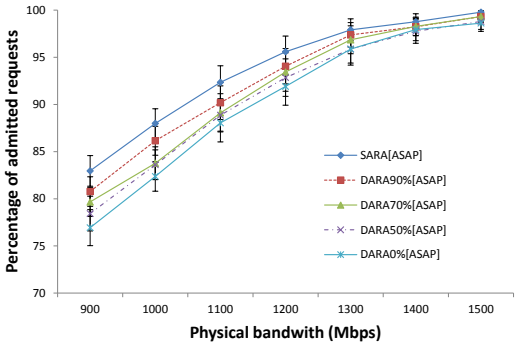


Fig. 5: Impact of bandwidth capacity and percentage of known requests on admission rate

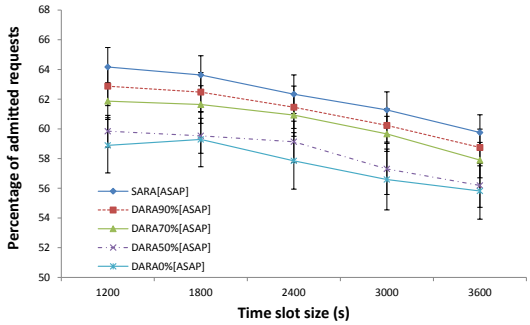


Fig. 6: Impact of timeslot granularity on request admission rate

This shows that a naive approach that merely maximizes request admission works well in a static scenario, but not in a dynamic one. ASAP outperforms MaxA up to 3.27% when 0% of requests are known in advance, and up to 1.97% when 50% are known.

As ASAP results in a higher number of accepted requests, we continue the evaluation with this function only. Figure 5 depicts more detailed results for ASAP, showing the influence of the percentage of requests known in advance on the solution. As expected, more known requests significantly increases performance. When no requests are known in advance, SARA outperforms DARA[ASAP] by up to 6.02%, while when 90% are known this is reduced to 2.19% at most.

C. Impact of time slot granularity

1) Scenario: In this section we evaluate the effect of time slot granularity on the performance of the algorithms, both in terms of solution optimality and execution complexity. The size of the time interval parameter was varied between 1200 and 3600 seconds. The number of use case instances is 15 (5 of each use case) and the number of total requests is 155. A link capacity of 400Mbps is used.

2) Results: Figure 6 studies the impact of time slot granularity on SARA and DARA for the ASAP objective function. As shown in this figure, the fine-grained experiment with shortest time slot results in the best performance. However, although more granularity increases the performance of the model, the complexity of the model significantly increases as well. Fine-grained time slot size increases the number of decision variables and constraints. The result of complexity measurement is provided in Table III. The table shows the average, minimum and maximum number of constraints and decision variables of any solved problem. In case of DARA, several ILP problems need to be solved throughout the day. The average number of invocations of the algorithm throughout the day is depicted as #k. The variable I represents the time slot granularity. When comparing performance for

TABLE III: Complexity of the solved ILP problems, in terms of number of constraints and number of decision variables

		SARA				DARA90%[ASAP]				DARA70%[ASAP]				DARA50%[ASAP]				DARA0%[ASAP]			
	I (sec)	**AVG**	**MIN**	**MAX**	**#k**	**AVG**	**MIN**	**MAX**	**#k**	**AVG**	**MIN**	**MAX**	**#k**	**AVG**	**MIN**	**MAX**	**#k**	**AVG**	**MIN**	**MAX**	**#k**
Constraints	**1200**	317774	306360	329103	1	226255	87407	280811	2.88	183321	66951	246462	5.62	154957	66981	229945	8.1	98525	9775	189878	11.94
	1800	211326	203642	218957	1	150267	57590	187301	2.88	121658	43893	164845	5.42	103631	44573	153981	7.52	67183	6579	126375	10.92
	2400	158130	152281	163828	1	113172	43436	141007	2.82	91661	33475	124094	5.2	77909	33775	116050	7.1	50621	5956	95854	9.84
	3000	127119	122540	131684	1	90667	35773	113279	2.74	73667	27790	100297	5	63249	27712	93268	6.72	41773	4100	76460	9.18
	3600	104862	101024	108640	1	75286	28143	94265	2.72	61218	21862	82471	4.72	52455	21782	77953	6.3	34827	3352	63925	8.42
Variables	**1200**	321265	321265	321265	1	240018	118473	290616	2.88	197887	99639	266422	5.62	168117	97107	253304	8.1	108556	10230	213600	11.94
	1800	214225	214225	214225	1	159564	76853	194087	2.88	131508	64521	176793	5.42	112678	65652	169407	7.52	74269	6832	142796	10.92
	2400	160705	160705	160705	1	120391	58300	146246	2.82	99322	49286	134374	5.2	84916	50002	128002	7.1	56164	5735	108176	9.84
	3000	129485	129485	129485	1	96647	47380	117900	2.74	79926	40680	108725	5	69135	40599	102202	6.72	46307	4182	86235	9.18
	3600	107185	107185	107185	1	80462	38236	97706	2.72	66671	32434	89002	4.72	57537	31523	85909	6.3	38901	3462	71836	8.42

SARA, an interval size of 1200 seconds yields 4.4% better results than a size of 3600 seconds. However, the complexity of the problem also increases, as both the average number of constraints and decision variables are increased threefold. Given the exponential time complexity of ILP solving algorithms, this increase in problem complexity results in an exponential increase in execution time. For DARA, a similar trend is observed. Moreover, it should be noted that in case fewer requests are known in advance, the complexity of a single DARA algorithm invocation decreases significantly. For example, when 0% of requests are known, both the number of constraints and variables are about 3 times smaller than when all requests are known in advance. In the former case, the algorithm needs to be executed between 8 and 11 times on average, while in the latter only once. However, due to the exponential time complexity, it is generally faster to solve a large number of small problems, rather than a small number of big ones.

VII. Conclusion

In this paper, we proposed an architecture for an SDN-based media production network, and a set of scheduling algorithms for advance bandwidth reservation (AR) satisfying the specific requirements of such networks. The algorithms are based on an ILP formulation that incorporates multi-path routing, time-variable bandwidth reservation, flexible start times, and request dependencies. It supports both file-based transfers and streaming sessions. The two proposed algorithm variants operate in an offline (i.e., SARA) and online (i.e., DARA) manner respectively.

Based on simulation results, the viability of AR scheduling in media production networks was assessed. Results show that when a significant portion of requests is known at the start of the day, AR significantly increases bandwidth efficiency and request admittance. Concretely, in case all requests are known at the start of the day, request admittance can be increased up to 6.02% compared to when requests are only known one hour before their desired start time. Additionally, the impact of time interval granularity on performance was evaluated. Time granularity increases algorithm accuracy and optimality in terms of request admittance. However, it also affects the ILP problem size, resulting in an exponential execution time increase. Concretely, a time slot size of 1200 seconds resulted in up to 4.4% more request admittance than a size of 3600 seconds.

In summary, we have proven the viability of using AR scheduling in media production networks to significantly improve bandwidth efficiency and request admittance. As future work, we plan to study very large scale scenarios in detail.

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] N. Charbonneau and V. M. Vokkarane, "A survey of advance reservation routing and wavelength assignment in wavelength-routed wdm networks," *Communications Surveys & Tutorials, IEEE*, vol. 14, no. 4, pp. 1037–1064, 2012.

[2] S. Naikstam and S. Figueira, "Elastic reservations for efficient bandwidth utilization in lambdagrids," *Future Generation Computer Systems*, vol. 23, no. 1, pp. 1–22, 2007.

[3] L.-O. Burchard, H.-U. Heiss, and C. De Rose, "Performance issues of bandwidth reservations for grid computing," in *Symposium on Computer Architecture and High Performance Computing*, pp. 82–90, 2003.

[4] K. Rajah, S. Ranka, and Y. Xia, "Advance reservations and scheduling for bulk transfers in research networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 20, pp. 1682–1697, Nov. 2009.

[5] C. Xie, H. Alazemi, and N. Ghani, "Rerouting in advance reservation networks," *Computer Communications*, vol. 35, no. 12, pp. 1411–1421, 2012.

[6] L. Zuo, M. M. Zhu, and C. Q. Wu, "Fast and efficient bandwidth reservation algorithms for dynamic network provisioning," *Journal of Network and Systems Management*, 2013.

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

[8] C. Guok, E. N. Engineer, and D. Robertson, "Esnet on-demand secure circuits and advance reservation system (oscars)," *Internet2 Joint*, 2006.

[9] B. Gibbard, D. Katramatos, and D. Yu, "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*, pp. 1–9, IEEE, 2006.

[10] J. Gu, D. Katramatos, X. Liu, V. Natarajan, A. Shoshani, A. Sim, D. Yu, S. Bradley, and S. McKee, "Stornet: Integrated dynamic storage and network resource provisioning and management for automated data transfers," in *Journal of Physics: Conference Series*, vol. 331, p. 012002, IOP Publishing, 2011.

[11] S. Sharma, D. Katramatos, D. Yu, and L. Shi, "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), pp. 68:1–68:11, IEEE Computer Society Press, 2012.