future internet

MDPI

Article

Scheduling for Media Function Virtualization

Gourav Prateek Sharma *, Wouter Tavernier, Didier Colle and Mario Pickavet

IDLab, UGent-IMEC, Technologiepark 126, Zwijnaarde, B-9052 Gent, Belgium;
Wouter.Tavernier@UGent.be (W.T.); Didier.Colle@UGent.be (D.C.); mario.pickavet@ugent.be (M.P.)
* Correspondence: gouravprateek.sharma@ugent.be; Tel.: +32-0-9264-3316

Abstract: Broadcasters are building studio architectures based on commercial off-the-shelf (COTS) IT hardware because of advantages such as cost reduction, ease of management, and upgradation. Media function virtualization (MFV) leverages IP networking to transport media streams between virtual media functions (VMFs), where they are processed. Media service deployment in an MFV environment entails solving the VMF-FG scheduling problem to ensure that the required broadcast quality guarantees are fulfilled. In this paper, we formulate the VMF-FG scheduling problem and propose a greedy-based algorithm to solve it. The evaluation of the algorithm is carried in terms of the end-to-end delay and VMF queuing delay. Moreover, the importance of VMF-FG decomposition in upgradation to higher-quality formats is also highlighted.

Keywords: virtualization; broadcasting; MFV; NFV; SMPTE

Citation: Sharma, G.P.; Tavernier, W.; Colle, D.; Pickavet, M. Scheduling for Media Function Virtualization. *Future Internet* **2021**, *13*, 167. https://doi.org/10.3390/fi13070167

Academic Editor: Sachin Sharma

Received: 7 June 2021
Accepted: 24 June 2021
Published: 28 June 2021

1. Introduction

The rising demand for new TV broadcast services, e.g., over-the-top (OTT), and high-quality content forces broadcasters to regularly upgrade studio infrastructure, resulting in high expenditures [1]. Cutthroat competition further accentuates the challenge of keeping the business viable. This challenge has forced broadcasters to seek alternative studio architectures for broadcast media production.

For several decades, serial digital interface (SDI) has been the most preferred solution for media transport within studios because of its robust and reliable performance [2]. However, upgrading SDI-based infrastructure to support higher quality media, e.g., full high definition (FHD) and ultra high definition (UHD) videos, requires high-speed SDI routing matrix, thus rendering the upgradation phase quite expensive and complicated [3]. As an alternative to SDI, packet-switched media transport architectures are gaining the attention of broadcasters. Internet protocol (IP) is a versatile solution to interconnect devices on the Internet as well as in private networks. Due to the decades of evolution in IP, the speed of packet forwarding devices has grown manifold. For instance, network cards and switches with Gbit/sec Ethernet (10GbE) interfaces are now commonly available. As a result of this evolution, all-IP studio architectures are being explored to transport uncompressed media streams across studios [2].

A complementary transition is happening in the media processing domain to move from bespoke hardware toward commercial off-the-shelf IT hardware (e.g., Intel Xeon Servers) running software applications to process media streams [4]. COTS hardware being inexpensive and ubiquitous, the upgrade of media processing infrastructure is much more economical compared to the specialized hardware boxes.

The progress made on these two fronts—COTS hardware exploitation for (i) media transport and (ii) media processing—is crucial in accelerating the adoption of media function virtualization (MFV). Analogous to network function virtualization (NFV), where network processing is accomplished via software deployed on COTS hardware, MFV aims to implement media functions (MFs) as software [4,5]. Despite its advantages, MFV faces a unique set of challenges that are not relevant to NFV. Broadcast production quality

standards are quite stricter than for networking services; for instance, a packet loss or even excessive delay in the arrival of a packet at the receiver can ruin the viewer experience. Deterministic networking (DetNet) can be exploited to provide end-to-end timing guarantees for the transport of media streams [6]. Specifically, DetNet mechanisms such as cycle specified queuing and forwarding (CSQF) allow scheduling packet transmissions along the network path that can fulfill the real-time requirements [7].

Media services are represented using a directed graph, referred to as a virtual media function forwarding graph (VMF-FG). The deployment of a media service entails the mapping of the given VMF-FG to the underlying infrastructure; this is referred to as the VMF placement and chaining (VMF-PC) problem. The naive VMF-PC algorithm assumes best-effort networking for media transport between VMFs, and thus does not provide any guarantee on metrics such as packet loss, end-to-end delay, or jitter. Overprovisioning bandwidth for media streams could result in an improvement in these metrics, yet deterministic performance cannot be guaranteed. Therefore, the VMF-FG mapping process needs to incorporate scheduling of VMFs and virtual links to ensure broadcast-quality guarantees.

In this paper, we first formulate the VMF-FG scheduling problem and then propose a greedy heuristic to solve it. The performance of the heuristic is evaluated by conducting numerical experiments.

The rest of the paper is structured as follows. The context of the research is presented in Section 2, followed by related works in Section 3. The problem statement and the proposed heuristic are described in Section 4. The evaluation setup and results are described in Section 5. Finally, our conclusions and final remarks are summarized in Section 6.

2. Research Context

2.1. IP and Virtualization

Figure 1 shows an overview of MFV architecture. The bottom layer is the MFV infrastructure layer that consists of the hardware and software components required to transport and process media streams in the broadcast studio. Usually, the hardware, i.e., compute, storage, and network, is virtualized. The VMF layer consists of software instances of MFs, referred to as VMFs, running in virtual machines and/or docker containers. VMFs can be chained together to realize complex media services such as the one shown in Figure 1. The topmost layer represents the media service layer.

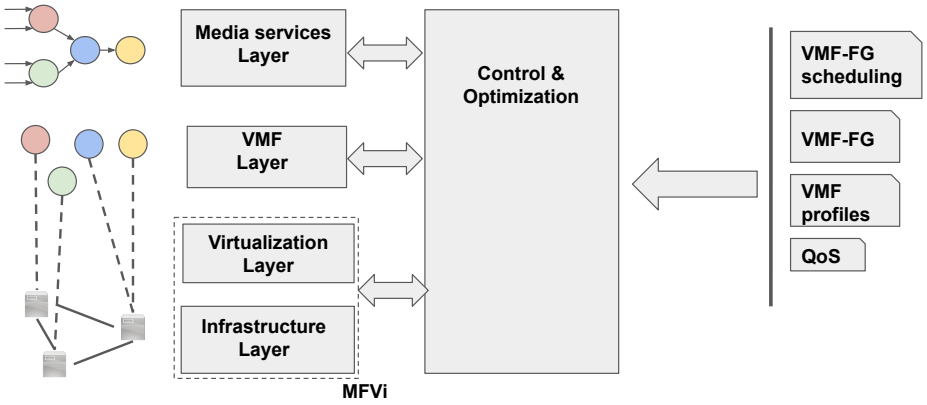

Figure 1. Overview of the MFV architecture.

Another important architecture component is the control and optimization (CO) layer. The roles of the CO layer include the management of other layers and the distribution of control signals to VMFs. In addition, the layer is responsible for scheduling the media service's VMF-FG to the MFVi. The CO layer is responsible for solving the VMF-FG scheduling problem by taking various input parameters as described in Section 4 into account. An optimization algorithm then generates a solution that describes the mapping of VMF-FG along with the VMF and virtual link schedules, as explained next. The mapping

and the schedule can then be used to configure various MFVi components so that the required guarantees are met.

2.2. DetNet and CSQF Scheduling

The efforts to incorporate performance guarantees such as zero packet loss and bounded end-to-end delay and jitter have continued for many years. A set of standards referred as time-sensitive networking was developed under IEEE 802.1 Ethernet to provide deterministic performance over local area networks (LANs). To this end, TSN employs mechanisms such as priority queuing, frame preemption, traffic shaping, etc. However, these mechanisms are not suited for large-scale IP networks. The IETF deterministic networking (DetNet) working group builds upon the TSN mechanism to support deterministic performance in large networks [6].

Cycle specified queuing and forwarding (CSQF) is a DetNet mechanism that evolved from a TSN mechanism called cycle queuing and forwarding [7]. CQF, also known as a peristaltic shaper, proposes to have, in each node, two queues per output port that operate in a ping-pong manner. At any instance of time, one queue is receiving packets from the upstream nodes and the other queue is transmitting to the downstream node. The time interval for which a queue receives (or transmits) packets is fixed and referred to as cycle time (T_c). It can be proved that using the two-queue model along a path in the next results in bounded delay and latency. However, CQF is incompatible with large-scale networks, as it requires the packets transmitted during a cycle time to be received in the same cycle time and transmitted in the next cycle, thus limiting the length of the physical links. In contrast, CSQF has no such requirement; this is discussed next.

Contrary to CQF, CSQF specifies the transmission time of packets along the path between the source and destination. The requirement to receive the packet in the cycle it was transmitted is no longer required. To achieve this, each CSQF-enabled node is equipped with multiple queues N_q (generally $N_q \geq 3$) per port, as shown in Figure 2. During a cycle time, one queue receives packets from the upstream nodes, and another queue transmits the packets it has received in the previous cycle. Here, another queue allows reception of the packets that were supposed to arrive in the next cycle. The early arrival could be due to variation in the processing time of the previous node. Moreover, this queue can be used to delay the arrival of specific packets by T_c. The role of packet transmission (and reception) is rotated cyclically among all the queues in each T_c. The time after which the application sender repeats the transmission pattern is referred to as a hypercycle, usually expressed as a multiple of cycle times.

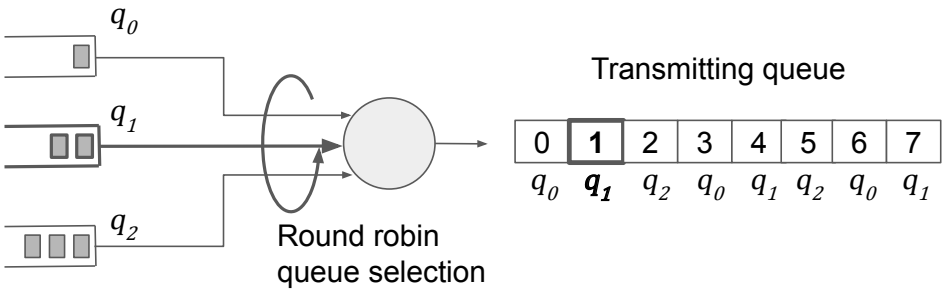

Figure 2. Mapping of cycle times to queues in the CSQF operation.

By having more than two queues in CSQF, a packet could be delayed by a specific amount ($(N_q - 2)T_c$)) by queuing the packet in a specific queue. In other words, at each node, the received packet should not only be forwarded to the right node, i.e., routing, but it should also be received in the right queue, i.e., scheduling. This can be achieved using segment routing (SR). After a packet is received at a node, its SID (SR identification) is used to first decide the output port. Each output port is associated with a number of SIDs, each of which corresponds to a queue. After the port selection, the SID is mapped to a

specific queue for reception and the packet is received in the mapped queue. By appending the packet to a stack of SID labels, the routing and scheduling at all the nodes of a path can be determined. It is worth noticing that there is no need to maintain a per-flow state in the intermediary nodes [7]. This allows scaling to large networks along with supporting a large number of flows.

As mentioned previously, an upper limit on timing metrics is required to ensure that broadcast quality is maintained. To this end, media streams between VMFs can be scheduled using CSQF. This will ensure that each VMF receives its input frames within the required time window so that the output frame is produced and then received in a timely manner by the downstream VMFs.

3. Related Works

Broadcast media production is undergoing a massive transformation propelled by IP and virtualization technologies. The migration to IP-based media transport from SDI is expected to accelerate in the near future. The interest in IP among broadcasters is emphasized by the fact that the SMPTE has released a suite of ST 2110 standards [8–10]. These standards describe how different media essences, i.e., video, audio, and ancillary data, can be transported independently using IP. Many all-IP broadcast studios are now being built based on these standards.

A live TV broadcast based on IP was produced by the British Broadcast Corporation (BBC) during the Glasgow 2014 Commonwealth Games [11]. UHD streams captured by multiple cameras in several competition venues were delivered to the software-defined production facility. The final program, after HEVC, was also delivered over IP. Likewise, a new broadcast facility in Wales, UK was built based on IP.

The Canadian public broadcaster CBC/Radio-Canada has constructed an all-IP facility responsible for over 100 TV, radio, and online program in Montreal, Canada [12]. The switching network is based on leaf-spine topology with 100 Gbps links. The network can support real-time multicast traffic along with redundancy for media streams. The software-defined network ensures that media streams can be distributed uniformly over the topology. The flexibility offered by IP is crucial for COTS-based media processing, as discussed next.

D. Luzuriaga et al. have also demonstrated the PoC for a vision mixer based on open-source software tools (e.g., OBS studio and KX studio) running on a COTS hardware platform [13]. They presented a low-cost replacement for a specialized hardware production system. The evaluation shows that the vision mixing setup has an acceptable delay of about 1.4 s, which is reasonable for professional (nonlive) media production scenarios. Furthermore, live media mixing based on software has been demonstrated by the BBC partnered with Isotama [14]. To operate the setup in real time and control the final live output, the software pipeline is interfaced with a browser-based operation tool.

Broadcast media services in an MFV environment can be represented using VMF-FGs. In our previous work, we proposed a VMF-FG decomposition algorithm [15]. The VMF-FG decomposition algorithm processes an input VMF-FG to produce an optimized VMF-FG that consumes fewer resources.

Recently, some work has been done to address the joint routing and scheduling problem for DetNets [16]. The problem is formulated as an integer linear program (ILP) and two methods—column generation and dynamic programming—are proposed to maximize traffic acceptance. An ultrafast and scalable greedy heuristic that is capable of solving the problem with a small penalty is also proposed. In this paper, the DetNet requests are simply two endpoints (source and sink). The problem of VMF-FG scheduling assumes a timing relationship between different virtual links. To the best of our knowledge, there has not been any research on scheduling VMF-FG to reliably deploy media services. This work aims to first formulate the VMF-FG scheduling problem and also proposes a heuristic to solve it.

4. System Model and Algorithm

In this section, we describe the VMF-FG scheduling problem and propose a greedy-based heuristic to solve it.

4.1. System Model

A media service s is represented using a DAG denoted by $\mathcal{G} = (\mathcal{F}, \mathcal{L})$, where $\mathcal{F}$ is the set of VMFs and $\mathcal{L}$ is the set of virtual links. The bandwidth requirement, in bytes per hypercycle, on a virtual link is denoted by bw_{f_i,f_j}. Here, the hypercycle time is equal to the frame interval, i.e., $|C| \times T_c = T_{frame} = 1/R_{frame}$. Thus, bandwidth requirement per hypercycle for (f_i, f_j) is equal to the frame size, e.g., an FHD (1920 × 1080 or 2 K) @30fps video stream with YCbCr color space and 4:2:2 sub-sampling, the bandwidth requirement per frame is 5184000B (5 MB).

The MFVi network is also denoted with a directed graph $G^I = (N, E)$, where N and E denote the set of physical nodes and links, respectively. The subset of physical nodes includes server nodes ($N_c \subset N$) with computational resources to host VMFs. The available bandwidth in cycle c on physical link (n_u, n_v) is denoted by $bw^c_{n_u,n_v}$.

Deploying a media service in an MFV environment involves mapping of the service's VMF-FG onto the MFVi network. In the problem of VMF-FG scheduling, given a VMF-FG $\mathcal{G}$ and MFVi G^I, (i) the VMFs should be assigned to the server nodes ($\alpha : \mathcal{F} \rightarrow N_c$), (ii) the virtual links should be mapped to the physical paths ($\gamma : \mathcal{L} \rightarrow \texttt{Pths}$), and (iii) the virtual links should also be assigned a packet transmission schedule ($\omega : \mathcal{L} \rightarrow \mathbb{R}^{|C|}$). As shown in Figure 3, the CO block has a view of MFVi network G^I; the scheduler in the CO block takes VMF-FG $\mathcal{G}$ and G^I as input and solves the VMF-FG scheduling to generate a solution. The solution (α, γ, and ω) is then used to configure MFVi, e.g., assign VMFs to servers, segment routing configuration for CSQF schedule, etc. Table 1 lists the various parameters, procedures, and variables involved in the ILP model and heuristic.

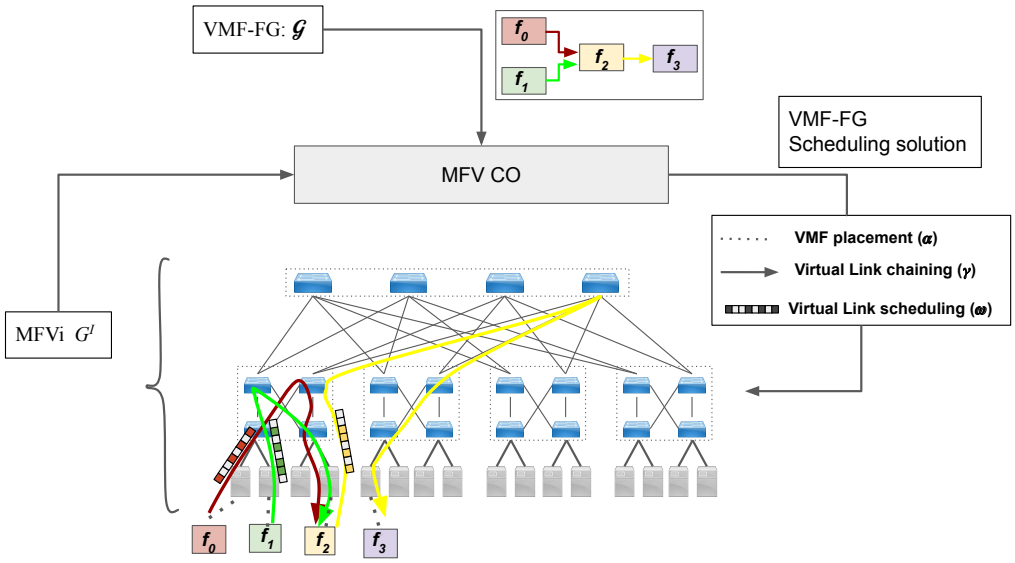

Figure 3. An overview of the VMF-FG scheduling problem.

4.2. Problem Complexity

Next, we analyze the complexity of the VMF-FG scheduling problem and show its NP-completeness. To this end, we formulate an instance of the VMF-FG scheduling problem as an ILP model. As the simpler instance of the problem is hard to solve, it can be inferred that the complete VMF-FG scheduling problem would be much harder to solve.

Table 1. Description of the notations used for different parameters and variables involved in the system model.

Notation	Description
$\mathcal{G} = (\mathcal{F}, \mathcal{L})$	VMF-FG representation of a media service, where $\mathcal{F}$ and $\mathcal{L}$ are the set of VMF and virtual links, respectively.
$G^I = (N, E)$	Directed graph representation of the MFVi network infrastructure, where N is the set of nodes and E is the set of physical links between the nodes.
R_{frame}	Number of frames transmitted per second on a media stream.
T_c	Cycle time.
N_c	Set of all server nodes ($N_c \subset N$), i.e., nodes with compute resources.
C	The set of all cycles in a hypercycle.
$bw_{n_u,n_v,c}^{f_i,f_j,p}$	The decision variable of the ILP formulation denotes the amount of bandwidth (in bytes) allocated to virtual link (f_i, f_j) on physical link (n_u, n_v) of path p during cycle c.
$x_p^{f_i,f_j}$	The decision variable of the ILP formulation indicates if virtual link (f_i, f_j) is mapped to a physical path or not.
$bw_{n_u,n_v,c}$	Available bandwidth (in bytes) on physical link $(n_u, n_v) \in E$ during cycle $c \in C$.
`DwstrNbrs` $(\mathcal{G}, f)$	The procedure returns a list of all downstream neighbors of f in VMF-FG $\mathcal{G}$.
`UpstrNbrs` $(\mathcal{G}, f)$	The procedure returns a list of all downstream neighbors of f in VMF-FG $\mathcal{G}$.
`Pths` (G^I, n_1, n_2, K)	The procedure returns K shortest paths between n_1 and n_2 in G^I.
`PthDel` (G^I, p)	The procedure returns the delay along p in G^I.
`CpuSch` (f, n, t_1, t_2)	The procedure returns a core, if free, on node n where f can be scheduled between t_1 and t_2.
`GrdAlloc` $(bw, \omega, n_1, n_2, t)$	The procedure returns the CSQF schedule for physical link (n_1, n_2) and the possible start and end cycle in the schedule; the bandwidth requirement is bw and the maximum end cycle is t .
`PthSch` (s_{lnk}, p)	The procedure checks if schedule s_{lnk} on link lnk is compatible with other links in path p; if yes, it returns the schedule for the full path.
α	The variable denotes the node and core assignment of VMFs.
γ	The variable denotes the physical path assignment of virtual links.
ω	The variable denotes the schedule assignment of virtual links.

Let us consider a VMF-FG with a total of $|\mathcal{L}|$ virtual links where a virtual link is denoted by (f_i, f_j). The following is the ILP formulation for the VMF-FG scheduling problem instance.

$$\max \sum_{p \in \texttt{Pths}(G^I, n(f_i), n(f_j), K)} x_p^{f_i, f_j}$$

$$st: \sum_{\substack{\forall (f_i, f_j) \in \mathcal{L} \\ p \in \texttt{Pths}(G^I, n(f_i), n(f_j), K) \\ , (n_u, n_v) \in p}} x_p^{i,j} bw_{n_u, n_v, c}^{f_i, f_j, p} \leq bw_{n_u, n_v, c}, \quad \forall (n_u, n_v) \in E, c \in C.$$

Here, the objective function is to maximize $\sum_{p \in \texttt{Pths}(G^I, n(f_i), n(f_j), K)} x_p^{f_i, f_j}$; where $x_p^{f_i, f_j}$ indicates if virtual link (f_i, f_j) is mapped to physical path p and $bw_{n_u, n_v, c}^{f_i, f_j, p}$ is the amount of bandwidth allocated to virtual link (f_i, f_j) on physical link (n_u, n_v) during cycle c. The constraint ensures that the total allocation during c on physical link (n_u, n_v) does not exceed the available bandwidth, i.e., $bw_{n_u, n_v, c}$. The objective of the formulation is to map as many virtual links to physical paths as possible. This formulation can be thus expressed as a decision problem (yes or no) by setting the threshold for the objective function to $|L|$. Therefore, a given VMF-FG is said to be scheduled if the objective value obtained after solving the ILP is $|L|$.

In order to prove that the VMF-FG scheduling problem is NP complete, we need to show that a known NP-complete problem can be reduced to it, similar to [16]. The k-disjoint paths (kDP) problem decides whether k discrete paths are possible between two nodes n_s and n_d of a given graph G. kDP can be reduced in the above instance of VMF-FG scheduling problem by setting the total cycles in a hypercycle to $|C| = 1$, the bandwidth requirement of virtual link $bw^{f_i, f_j} = 1, \forall (f_i, f_j) \in \mathcal{L}$, and the available bandwidth in each cycle to be $bw_{n_u, n_v, c} = 1, \forall (n_u, n_v) \in E, c \in C$. Assuming $k = |L|$ and the above assumptions, the decision problem for VMF-FG scheduling problem returns *True* if k discrete paths are possible for $\mathcal{L}$. All the above reduction steps have polynomial complexity and any solution can be verified in polynomial time, thus the VMF-FG scheduling problem is NP-complete.

Above, we considered an (simpler) instance of the VMF-FG problem. The hardness of the problem is further accentuated because of multiple choices to place the VMFs and route the virtual links of the given VMF-FG. Next, we present a scalable greedy heuristic to solve the problem.

4.3. VMF-FG Scheduling Algorithm

The VMF-FG scheduling algorithm shown in Algorithm 1 can be implemented using two different graph traversal algorithms, i.e., breadth-first search (BFS) and depth-first search (DFS). With BFS, scheduling of all VMFs at the same depth from the starting node is done together, whereas with DFS, scheduling of VMFs along a path (as far as possible) is performed before backtracking. The two flavors of the VMF-FG scheduling utilize two different data structures in order to traverse the given VMF-FG, the queue for BFS, and the stack for DFS. The VMF-FG scheduling variables are initialized and the VMF-FG traversal starts by initializing Q_{vmf} with the sink node. Next, a while loop is used to traverse the VMF-FG starting from vmf_{snk} until Q_{vmf} is empty. A VMF is dequeued (popped) from Q_{vmf} and stored in f.

For each (already placed) neighboring VMF $f_{dw} \in \texttt{DwstrNbrs}(\mathcal{F}, f)$ downstream to f, the scheduling of VMF f and the associated virtual link (f, f_{dw}) is iteratively attempted using procedure `VlinkSch`. The tentative node to place f is selected using the next-fit algorithm, starting from the same node where f_{dw} is placed, i.e., $\alpha[f_{dw}]$, then the next node and so on ($\alpha[f_{dw}] + \delta n$). The neighbors of f are then added to the VMF queue (l. 12–14). Next, we describe the procedure for virtual link scheduling in detail.

The procedure to schedule virtual links is presented in Algorithm 2. The procedure `VlinkSch` takes the bandwidth requirement bw_{f_i, f_j} of the virtual link (f_i, f_j), the server nodes for the potential placement of f_i and f_j: n_1 and n_2, the current placement (α), chaining (γ), and link schedule (ω) as input. As illustrated in Figure 4, the input schedule at VMF f_j must end $d_{proc}(f_j)$ before f_j starts transmitting the processed (output) frame (l. 2).

Next, procedure `PthSchGrd` (discussed next) is called to greedily allocate bw_{f_i,f_j} among $|C|$ cycles between t_{end} and $t_{end} - T_c + 1$. The feasible path p, the associated schedule s, and the timings of the start and end cycles of s are returned by `PthSchGrd`. In the case that a feasible path/schedule does not exist, the procedure is terminated by returning ϕ (l. 5). Else, VMF timings are updated and a schedule for f_i is generated using procedure `CpuSch`. As shown in Figure 4, f_i's output timings are obtained by delaying f_j's input timings (l. 9) by the path delay (`PthDel(`p`)`). `CpuSch` returns an available CPU core on n_1 where f_i can be scheduled between time interval $(T^{out}_{start}[f_i] - d_{proc}(f_i), T^{out}_{start}[f_i])$. Before returning, the placement, chaining, and scheduling variables are updated (l. 14).

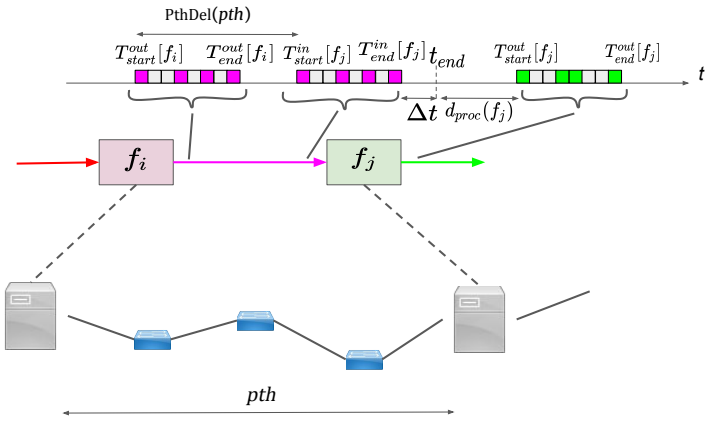

Figure 4. Illustration for the relationship between different VMF timing schedules.

Algorithm 1 Simplified pseudocode for the VMF-FG scheduling algorithm; a given VMF-FG $\mathcal{G} = (\mathcal{F}, \mathcal{L})$ is to be scheduled on the MFVi network $G^I = (N, E)$

```
1  α, γ, ω ← ϕ, ϕ, ϕ;
2  for vmf_snk ∈ F_snk do
3      Q_vmf ← {vmf_snk};          /* initializing a queue if using BFS else a stack if using DFS */
4      while |Q_vmf| > 0 do
5          f ← Q_vmf.dequeue();                    /* if DFS do pop() instead of dequeue()  */
           /* schedule f and vlink (f, f_dw)                                                  */
6          for f_dw in DwstrNbrs(G, f) do
7              δn ← 0;
8              while VlinkSch(f,f_dw,bw_{f,f_dw}, α[f_dw] + δn, α[f_dw], α, γ, ω) do
9                  δn ← δn + 1;
10             end
11         end
           /* enqueue the upstream neighbors                                                  */
12         for nbr in UpstrNbrs(G, f) do
13             Q_vmf.enqueue(nbr)
14         end
15     end
16 end
```

Procedure `PthSchGrd` iterates over all the paths between n_1 and n_2 and returns a feasible schedule, if it exists. More precisely, for path p, first, a greedy schedule is allocated for the destination link $((p[-2], p[-1]))$ and it is checked whether it is compatible with the other links in p using procedure `PthSch`. If not, ϕ is returned; otherwise, `PthSch` returns the packet schedule for the whole path p. The current path p, the associated schedule s, and the start and end timings t_{start}, t_{end} of s are returned.

Algorithm 2 Procedure for allocating a schedule for a virtual link

```
   /* VMF input output timings                                                    */
   Global  : G^I, T^out_start, T^out_end, T^in_start, T^in_end
1  Procedure VlinkSch(f_i, f_j, bw_{f_i,f_j}, n_1, n_2, α, γ, ω):
2      t_end ← T^out_start[f_j] − d_proc(f_j);
3      p, s, t^in_start, t^in_end ← PthSchGrd(bw_{f_i,f_j}, n_1, n_2, ω, t_end);
4      if p == φ then
5          return False;
6      end
7      else
8          T^in_start[f_j], T^in_end[f_j] ← t^in_start, t^in_end;
9          T^out_start[f_i], T^end_in[f_i] ← t^in_start − PthDel(G^I, p), t^in_end − PthDel(G^I, p);
           /* get a core on n_1 for f_i scheduling                                */
10         core ← CpuSch(f_i, n_1, T^out_start[f_i] − d_proc(f_i), T^out_start[f_i]);
11         if core == φ then
12             return False
13         end
14         α[f_i], γ[(f_i, f_j)], ω[(f_i, f_j)] ← (n_1, core), p, s;
15         return True;
16     end
17 end
```

We are interested in scalability of the above heuristic with the problem size; thus, we analyze the run time complexity in terms of VMF-FG and MFVi network size, i.e, $|\mathcal{F}|$, $|\mathcal{L}|$, $|N|$, and $|E|$.

The run time of Algorithm 3 is dominated by procedure `Pths` that returns a maximum of K paths between n_1 and n_2, and thus has complexity $O(|K||N|(|E| + |N|log(|N|)))$; thus, procedure `VlinkSch` in Algorithm 2 also has $O(|K||N|(|E| + |N|log(|N|)))$ complexity. The given VMF-FG procedure is traversed using BFS (Algorithm 1) that has a complexity of $(|\mathcal{F}| + |\mathcal{L}|)$ and $\forall(f, f_d w) \in \mathcal{L}$ procedure `VlinkSch` is called. This results in an overall run time complexity of the heuristic to be $O(|\mathcal{F}| + |\mathcal{L}||K||N|(|E| + |N|log(|N|)))$.

Algorithm 3 Procedure for chaining and scheduling on a path

```
   Global : G^I, |C|
1  Procedure PthSchGrd(bw, n_1, n_2, ω, t_end):
2      t̄_end ← t_end%|C|;
3      for p in Pths(G^I, n_1, n_2, K) do
4          for Δt in [0, |C|] do
5              t̄^new_end ← (t̄_end − Δt)%|C|;
6              s_lnk, t_start, t_end ← GrdAlloc(bw, ω, p[−2], p[−1], t̄^new_end);
7              if s_lnk ≠ φ then
8                  s ← PthSch(s_lnk, p);
9                  if s ≠ φ then
10                     return p, s, t_start, t_end;
11                 end
12             end
13         end
14     end
15     return φ, φ, φ, φ;
16 end
```

5. Evaluation

In this section, we perform a few numerical experiments to evaluate the performance of our VMF-FG scheduling algorithm. First, we describe the setup used for evaluation. Next, the performance of our algorithm is evaluated in terms of the VMF-FG's delay. We also present the impact of media quality formats on resource utilization.

5.1. Evaluation Setup

Studio architectures have stringent performance requirements such as lossless and high-speed (multi-Tbps) switching, support for point-to-multipoint communication, etc. In addition to fulfilling these requirements, data-center topologies, such as fat-tree, are also easier to upgrade for high-quality media formats [2]. Therefore, the MFVi considered for evaluation is arranged in the fat-tree topology. The topology consisting of κ pods, where each pod has $(\kappa/2)^2$ server nodes, $\kappa/2$ access layer switches, and $\kappa/2$ aggregate layer switches and the core layer contains $(\kappa/2)^2$ switches. The total number of server nodes in the topology are $\kappa^3/4$. We assume $\kappa = 8$, i.e., there are a total of 128 servers, with each server node containing 20 cores. All the devices (switches and server nodes) in the MFVi physical network are equipped with 10 GbE interfaces.

As mentioned earlier, media services are represented via VMF-FGs. The focus of this paper is on VMF-FG scheduling; we therefore pay attention to the resource allocation aspects, not to the specific VMF-FG functionality. To this end, the VMF-FG considered for the evaluation is shown in Figure 5. It consists of 11 VMFs (and 6 endpoints) and 16 virtual links. The VMF-FG is considered in three media formats: (i) HD @ 30 fps, (ii) HD @ 60 fps, and (iii) FHD @ 30 fps. The subsamping and sample encoding of the VMF-FG are assumed to be 4:2:2 and 10 bits, respectively. These media formats are typically employed in today's media production environments [3].

The other parameters and their corresponding values (range) for media service requests, MFVi, and VMFs are listed in Table 2.

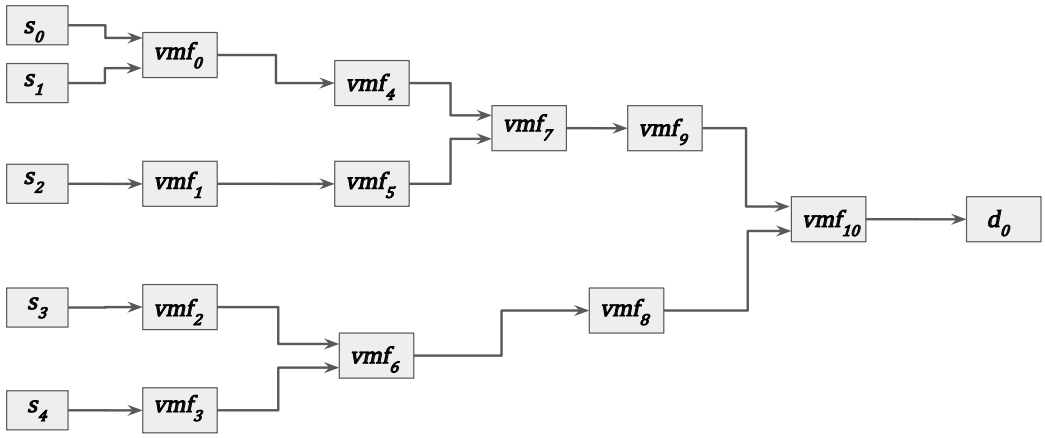

Figure 5. Illustration of the VMF-FG used for the evaluation.

5.2. End-to-End Delay

End-to-end delay is an important metric for broadcast media services, especially for live production scenarios. Here, we report the end-to-end (E2E) delay in a VMF-FG, i.e., the maximum delay among all possible paths between the VMF-FG's sources and sinks for both the BFS and DFS traversal of the proposed heuristic. In Figure 6a, we compare the average E2E delay for different cycle time (T_c) values. It can be observed that the E2E delay increases with T_c. This results from the fact that the delay along the physical paths corresponding to the virtual links increases with T_c as the queuing delay in switching nodes increases.

Table 2. Default values/range of various parameters involved in the evaluation.

Parameter	Value or Range	Units
Topology	Fat-tree ($\kappa = 8$)	-
Formats	HD@30fps, HD@60fps, FHD@30fps	-
CPU cores/node	20	-
Link bandwidth	10	Gbps
Cycle time (T_c)	{100, 200, 300, 400}	µs
VMF processing delay	5	ms
Node (switch) processing delay (d_{n_u,n_v})	40	µs
Links delay (d_n)	80	µs

It is interesting to note that the E2E delay as well the VMF queuing delay for DFS is more than with BFS. As scheduling occurs on a path-by-path basis, the scheduling on one path is impacted by the previously scheduled path, thus increasing the E2E delay on the longest path. On the other hand, for BFS, all paths have relatively similar delays, thus reducing the E2E delay along the longest path.

We also report the E2E delay variation with M. The decomposition of a VMF-FG leads to VMF decomposition, which results in the reduction of VMF processing delay. This results in the decreasing E2E VM-FG delay with M.

The are various components of E2E delay. The contribution due to video frame queuing is an important metric because it highlights the efficiency of VMF-FG scheduling. Here, we define the average VMF queuing delay as the mean of VMF queuing delay in all VMFs of the VMF-FG. The smaller the value of the average VMF queuing delay is, the better is the VMF-FG scheduling. The average queuing delay slightly decreases with T_c as shown in Figure 6b. However, VMF-FG decomposition results in increasing the average VMF queuing delay due to higher consolidation of VMFs per node.

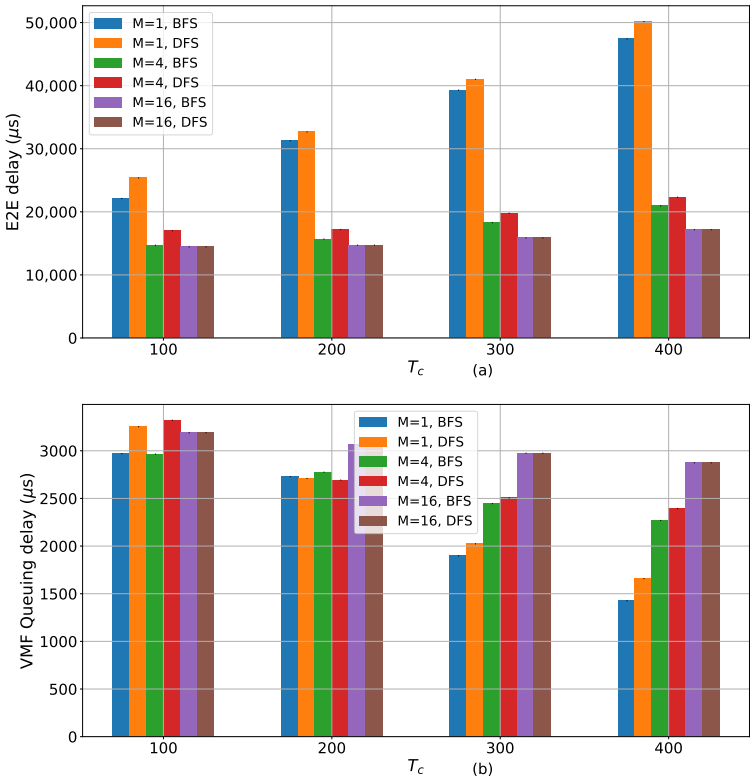

Figure 6. (**a**) The average E2E delay and (**b**) the average queuing delay versus cycle time (T_c).

5.3. Impact of Formats

Here, we report the impact of media formats on the performance of the scheduling heuristic with BFS traversal. The format of media streams affects resource utilization efficiency. Figure 7 compares the average number of cores used per node for three different video formats. With no VMF-FG decomposition ($M = 1$), the average core utilization decreases with higher quality of media formats. The network bandwidth increasingly becomes a bottleneck with an increase in media quality; thus, fewer VMFs are placed at a node even if free cores are available. The average core utilization for DFS is not reported as it remains same as BFS.

It can also be observed that VMF-FG decomposition improves the average core utilization per node. By decomposing VMF-FG, bandwidth requirement per virtual link decreases, thus increasing VMF consolidation per node.

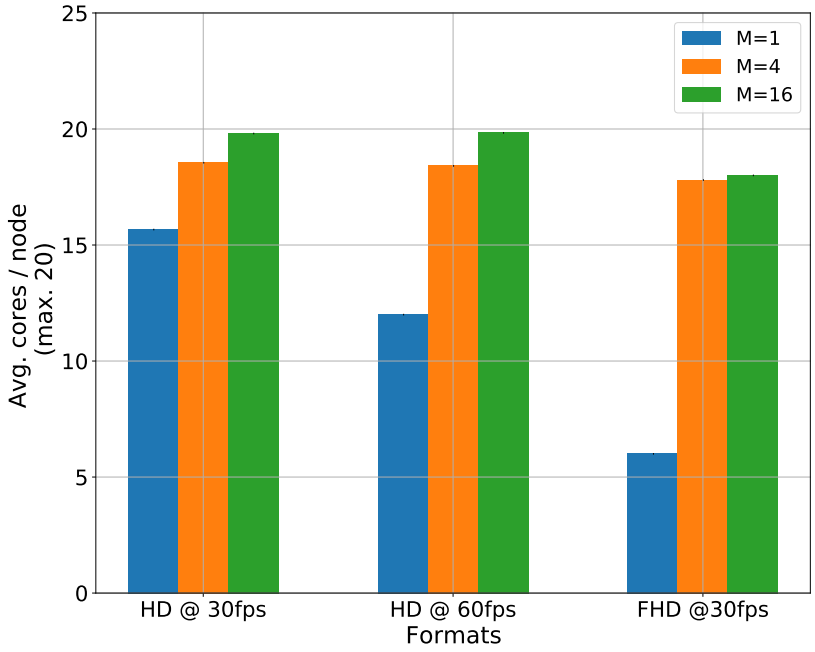

Figure 7. Impact of media formats on resource utilization for different M.

6. Conclusions

The transition from specialized hardware to COTS IT infrastructure is taking place in broadcast studios, i.e., IP networking for media transport and general-purpose compute for media processing. MFV proposes to utilize COTS compute hardware to run media processing as VMFs that can be chained together in the form of VMF-FGs. To ensure that broadcast-quality guarantees in an MFV environment are met, it is important to not only map VMFs and virtual links to the underlying MFVi but also to schedule them in a timely manner. To this end, we first formulate an instance of the VMF-FG scheduling problem as an ILP and prove its NP-completeness. We have then proposed a greedy-algorithm, based on BFS and DFS graph traversal, to find its solution. The evaluation of the algorithm shows an increase in the end-to-end delay with increasing cycle time. We also highlighted the improvement in the end-to-end delay with VMF-FG decomposition, particularly for high-quality formats.

For future work, we plan to investigate the implementation of a media service in an MFV environment with VMF-FG scheduling.

Author Contributions: Conceptualization, G.P.S., W.T., D.C. and M.P.; funding acquisition, M.P.; investigation, G.P.S., W.T. and D.C.; methodology, G.P.S., W.T. and D.C.; resources, M.P.; supervision, W.T., D.C. and M.P.; writing—original draft, G.P.S.; writing—review and editing, G.P.S. and W.T. All authors have read and agreed to the published version of the manuscript.

Funding: This research was (partially) funded by the Flemish FWO SBO S003921N VERI-END.com (Verifiable and elastic end-to-end communication infrastructures for private professional environments) project.

Data Availability Statement: Not Applicable, the study does not report any data.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

References

1. Fremeije, M. The Rising Need for Media Function Virtualization. Available online: https://www.redhat.com/cms/managed-files/IDC-The_Rising_Need_for_Media_Function_Virtualization.pdf (accessed on 24 June 2021).
2. Kojima, T.; Stone, J.J.; Chen, J.; Gardiner, P.N. A Practical Approach to IP Live Production. In Proceedings of the SMPTE 2014 Annual Technical Conference Exhibition, Amsterdam, The Netherlands, 27–29 October 2014; pp. 1–16.
3. Paulsen, K. Prepping for the IP Transition. Available online: https://fdocuments.in/reader/full/the-definitive-guide-to-prepping-for-the-ip-transition-2020-03-06-the-definitive (accessed on 24 June 2021).
4. Nevion Ltd. The Road to COTS and the Cloud for Real-Time Broadcast Production. Available online: https://nevion.com/resources/whitepapers/whitepaper-cots-and-the-cloud-for-real-time-broadcast-production/ (accessed on 24 June 2021).
5. Herrera, J.G.; Botero, J.F. Resource allocation in NFV: A comprehensive survey. *IEEE Trans. Netw. Serv. Manag.* **2016**, *13*, 518–532. [CrossRef]
6. Finn, N.; Thubert, P.; Varga, B.; Farkas, J. Deterministic networking architecture. Available online: https://tools.ietf.org/id/draft-ietf-detnet-architecture-04.html (accessed on 24 June 2021).
7. Chen, M.; Geng, X.; Li, Z. Segment Routing (SR) Based Bounded Latency. Available online: https://datatracker.ietf.org/doc/html/draft-chen-detnet-sr-based-bounded-latency-01 (accessed on 24 June 2021).
8. ST 2110-10:2017—SMPTE Standard—Professional Media Over Managed IP Networks: System Timing and Definitions. In *SMPTE ST 2110-10:2017*; SMPTE: Hong Kong, China, 2017; pp. 1–17. [CrossRef]
9. ST 2110-20:2017—SMPTE Standard—Professional Media Over Managed IP Networks: Uncompressed Active Video. In *SMPTE ST 2110-20:2017*; SMPTE: Hong Kong, China, 2017; pp. 1–22. [CrossRef]
10. ST 2110-30:2017—SMPTE Standard—Professional Media Over Managed IP Networks: PCM Digital Audio. In *SMPTE ST 2110-30:2017*; SMPTE: Hong Kong, China, 2017; pp. 1–9. [CrossRef]
11. Nicholson, M. The IP Network behind the R&D Commonwealth Games 2014 Showcase. Available online: https://www.bbc.co.uk/rd/blog/2014-07-commonwealth-games-showcase-network (accessed on 24 June 2021).
12. Poulin, F.; Keroulas, P.; Nyamweno, S.; Vermost, W.; Ferreira, P.; Kostiukevych, I. How CBC/Radio-Canada Tested Media-over-IP Devices to Build its New Facility. *SMPTE Motion Imaging J.* **2020**, *129*, 35–44. [CrossRef]
13. Luzuriaga, D.; Lung, C.H.; Funmilayo, M. Software-Based Video–Audio Production Mixer via an IP Network. *IEEE Access* **2020**, *8*, 11456–11468. [CrossRef]
14. Winter, D. Compositing and Mixing Video in the Browser. Available online: https://isotoma.com/blog/2017/06/23/compositing-and-mixing-video-in-the-browser/ (accessed on 24 June 2021).
15. Sharma, G.P.; Colle, D.; Tavernier, W.; Pickavet, M. Improving Resource Utilization with Virtual Media Function Decomposition. In Proceedings of the IEEE 2020 Fourth International Conference on Multimedia Computing, Networking and Applications (MCNA), Valencia, Spain, 19–22 October 2020; pp. 31–37.
16. Krolikowski, J.; Martin, S.; Medagliani, P.; Leguay, J.; Chen, S.; Chang, X.; Geng, X. Joint routing and scheduling for large-scale deterministic IP networks. *Comput. Commun.* **2021**, *165*, 33–42. [CrossRef]