Improving resource utilization with Virtual Media Function decomposition

Gourav Prateek Sharma, Didier Colle,
Wouter Tavernier, Mario Pickavet
IDLab, Department of Information Technology
Ghent University - IMEC,
Email: {gouravprateek.sharma, didier.colle, wouter.tavernier, mario.pickavet}@ugent.be

***Abstract*—For many years, media transport and processing in professional media environments have been accomplished using specialized hardware. The flexibility of IP networking and Media Function Virtualization (MFV) can enable broadcasters to build cost-efficient architectures for the deployment of media services. These architectures can be further exploited to perform decomposition of Virtual Media Functions (VMFs) resulting in improved utilization of the MFV Infrastructure (MFVi) resources. To this end, this paper presents an algorithm aimed at optimizing the VMF Forwarding Graphs (VMF-FGs) media services. A first-fit based heuristic is also proposed for deploying media services on a given MFVi. The evaluation results indicate that VMF-FG decomposition can significantly improve the request acceptance ratio and reduce resource requirements.**

***Index Terms*—media, broadcast, IP, production, decomposition, square-division, SMPTE, video, 4K**

I. Introduction

TV broadcasters are facing multiple challenges in order to survive in the current competitive TV industry. The key challenge is to minimize the total incurred costs, i.e. Capital Expenditure (CAPEX) and Operational Expenditure (OPEX) while still offering high-quality media services to its customers [1]. The pressure to reduce costs has forced broadcasters to explore more cost-efficient architectures for media production. In legacy broadcast studios, media processing and transport are typically achieved using bespoke hardware appliances. The fixed-function hardware usually results in high-performance both in terms of the amount of total traffic rate and latency, in addition to lower power-consumption [2]. However, these hardware appliances tend to be inflexible and rigid. Moreover, replacement/up-gradation of proprietary hardware is hard due to their high cost and long design (ASIC) cycle.

In recent years, broadcasters have been looking into Commercial-Off-the-Shelf (COTS) alternatives to the specialized hardware. The idea is to use standard IT hardware (e.g., compute, network, and storage) for media transport and processing. On one hand, IP networking can carry media streams inside broadcast studios and on the other hand, general-purpose hardware (e.g. Intel Xeon servers) can run software instances of Media Functions (MFs) [3].

The deployment of network services in Network Function Virtualization (NFV) environment has been studied extensively [4]. However, due to the peculiar characteristics of Virtual Media Functions (VMFs) vis-a-vis Virtual Network Functions (VNFs) (discussed in section II), a special attention towards Media Function Virtualization (MFV) is required. Therefore, the problem of efficient deployment of media services in a virtualized broadcast studio needs to be studied, i.e., how special opportunities arising as a result of IP-based media transport and virtualization of media processing can be exploited?. Especially, algorithms to optimize the VMF Forwarding Graph (VMF-FG) for a media service to produce another VMF-FG; which is functionally equivalent to the original VMF-FG but less resource-intensive, are required. To the best of our knowledge, there has been a limited work exploiting these opportunities [5]. Furthermore, physical resources need to be allocated to the media services; that is accomplished by solving the VMF Placement and Chaining (VMF-PC) problem. The rest of this paper is organized as follows. In section II, we provide the related work and background information about various technologies involved in a professional media environment with a focus on MFV. Section III describes the problem of VMF-FG decomposition, an algorithm to solve VMF-FG decomposition and a first-fit based heuristic to solve VMF placement and chaining (VMF-PC). Section IV presents and discusses the performance of the proposed algorithm and the heuristic and Section V concludes the paper as well as discusses the potential future research.

II. Background

A. Media Transport

Media broadcast facilities have traditionally relied upon Serial Digital Interface (SDI) technology to interconnect various equipment in broadcast studios [3]. Media streams are carried as serial digital signals which are circuit-switched, i.e., a non-blocking switching matrix connects a specific switch's input to a specific output. Media transport based on SDI is robust, deterministic, and reliable. As no multiplexing of signals is possible in SDI, each signal in SDI is carried over a separate coaxial cable resulting in a large number of cables. Moreover, broadcasters are finding it difficult to transport newer high-resolution media streams (e.g., 4K or 8K) over SDI. Since the bitrate of uncompressed streams can be as high as tens of Gbps, the switching becomes impractical or too expensive with the current SDI routers. Lately, as an alternative to SDI, IP networking is gaining great attention among broadcasters [6].

In IP-based studios, media streams are broken at the sender into multiple packets and then transported independently over the network to the receiver where they are re-assembled. Unlike SDI, IP links are bidirectional and are capable of carrying multiple media streams; therefore resulting in a lesser amount of cabling. Since IP systems have been around for many decades, inexpensive COTS switches that are capable of switching several 4K uncompressed media streams [6] are widely available in the market. To further scale the studio infrastructure, a high-speed switching fabric using COTS switches can be built along the lines of data center networks. The Society of Motion Picture and Television Engineers (SMPTE) has released a suite of standards that describe how to transport uncompressed media streams over an IP network [7], [8], [9], [10]. Unlike earlier standards (e.g. ST 2022-6) [11], ST 2110 allows putting distinct media components, i.e., video, audio, and ancillary data signals to be transported and processed as independent streams.

Owing to the high bitrates, transport and processing of uncompressed high-resolution video streams in a broadcast studio can be challenging. Decomposing a high-bandwidth stream into multiple low-bandwidth streams can alleviate this issue. To this end, SMPTE has described three stream decomposition techniques, namely– (i) Phased, (ii) Square-Division (SD), and (iii) Sample Interleave (SI), in its recommended practices document [12]. As only SD decomposition is relevant to our work, we describe only this decomposition technique next.

M-way SD decomposition of a video stream would mean

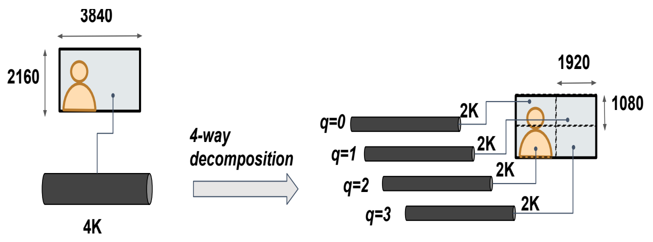

Fig. 1. An example M-way stream decomposition.

each frame of the video stream is divided into M sub-frames which are carried by M separate sub-streams. For example, consider an uncompressed 4K (3840x2160p30) video stream with the bitrate 4.976 Gbps shown in Fig. 1. After 4-way SD decomposition, corresponding to the four quadrants of the 4K frame, four 2K (1920x1080p30) sub-streams each with the bitrate 1.244 Gbps are obtained. These low-bandwidth sub-streams can be transported and processed much easily as compared to one high-bandwidth stream.

B. MFV Overview

Due to the high amount of processing required and need for real-time processing (low latency) broadcasters have usually preferred specialized hardware appliances to process media streams. But the performance of commodity hardware has improved lately so that processing of media streams can also be done on commodity hardware. For example, consider an MF which mixes two video streams with a special effect (e.g., wipe or dissolve transition). This MF is typically realized using a piece of special hardware equipment called vision-mixer. However, this function can also be realized in software running on commodity hardware (x86 or ARM CPU) [5].

Adoption of COTS IT platforms to build services is not an isolated transition happening in the TV broadcast industry. Telcos have been moving towards Network Function Virtualization (NFV): a new architecture where packet-processing functionality, traditionally based on bespoke hardware (middleboxes), is now being implemented in Virtual Network Functions (VNFs) running on commodity hardware [4]. By leveraging virtualization technologies, e.g. Virtual Machines and Docker Containers, further benefits like management and deployment flexibility, and resource scaling can be achieved [4]. Taking NFV as an analogy: we refer to the architectural framework where media processing is done using Virtualized Media Functions (VMFs) running in a virtualized environment as Media Function Virtualization (MFV). A few examples of VMFs are listed in Table I.

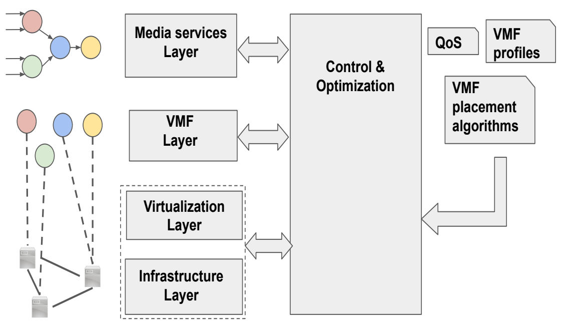

Fig. 2. Overview of the MFV architecture.

TABLE I
EXAMPLES OF VMFS ALONG WITH A SHORT DESCRIPTION

VMF	Interfaces	Description
brg-adj	1	Adjusts brightness (luminance) values of the input video stream.
γ-corr	1	Non-linear transformation of luminance values of the input video stream.
pip	($\geq$) 2 inputs	Embeds the given (small resolution) video streams into another video stream at a given location.
splt-sc	2-3 inputs	Lays middle portion of 2/3 video streams next to each other.
chrm-key	2	Replaces the background (green screen) of a video stream with another stream.

An overview of the MFV architecture is shown in Fig. 2. The lowest layer is the MFV infrastructure (MFVi) layer consisting of COTS hardware for compute, network, and storage. Also, re-configurable hardware, e.g. FPGA and GPUs, are occasionally leveraged to implement MFs [13], especially to ensure real-time processing and/or to reduce energy consumption. Above the MFVi, sits the virtualization layer which abstracts the underlying hardware for media processing applications running on it. Depending on the selected virtualization technology, the virtualization layer can be a Hypervisor (Type 1 or 2) or a Docker engine. Next, the VMF layer consists of Virtual Machines and/or containers responsible for hosting one or more media processing applications. Media services are implemented by linking multiple VMFs together, as discussed in the next section. Analogous to MANO in NFV architecture, Control and Optimization layer in MFV is responsible for tasks like management of resources (physical and virtual), placement and chaining of VMFs, in addition to maintaining QoS requirements of media services [4].

III. VMF-FG Decomposition and Deployment

A. Problem overview

Assume a media production environment where media processing is accomplished via VMFs. In order to implement the desired media service s, the traffic needs to be processed through a network of VMFs. This network of VMFs is represented using a directed graph referred as VMF Forwarding Graph (VMF-FG) $\mathcal{G}$ akin to VNF-FG in NFV [4]. The set of nodes $\mathcal{F}$ in $\mathcal{G}$ are traffic sources, sinks, and VMFs and the set of edges $\mathcal{L}$ between the nodes are virtual links between sources, VMFs and sinks. Figure 3 (a) shows an

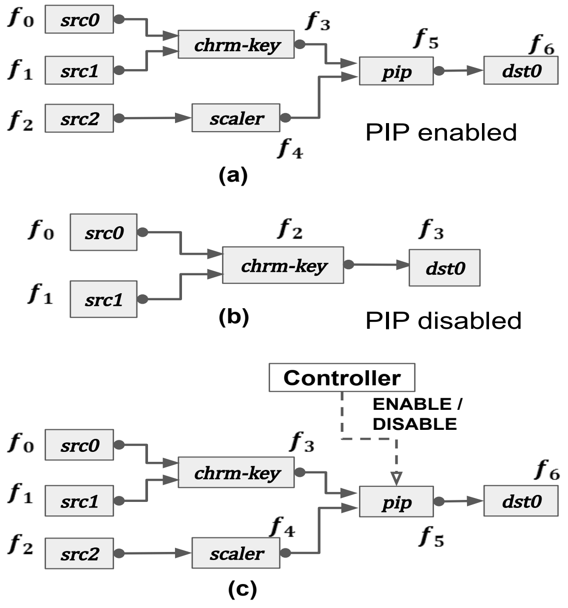

Fig. 3. An example VMF-FG representation of a media service with static (a and b) and dynamic (c) VMF-FG deployment.

example of how VMF-FGs can be used to represent a media service which processes traffic from three sources *src0*, *src1* and *src2* and outputs the processed traffic to the destination *dst0*. The sources in a media service can be a media stream coming from a camera or storage whereas destinations can be a screen, a local storage device or an OTT server. In Figure 3 (a) , a *chrm-key* VMF is first used to implement chroma-keying operation on video streams from two sources *src0* and *src1*. Also, the video stream from source *src2* is scaled by *scaler* VMF, which has been configured to output the video frames of required dimensions. The output frames of the *scaler* VMF are then embedded into the output frames generated by *chrm-key* VMF using *pip* (picture-in-picture) VMF. For this media service, VMF-FG can be expressed as $\mathcal{G} = (\mathcal{F}, \mathcal{L})$, where $\mathcal{F} = \{f_0, f_1, f_2, f_3, f_4, f_5, f_6\}$ and $\mathcal{L} = \{(f_0, f_3), (f_1, f_3), (f_2, f_4), (f_4, f_5), (f_3, f_5), (f_5, f_6)\}$. Other complex media services can be similarly represented using VMF-FGs.

The *pip* functionality of the VMF-FG shown Fig. 3 (a) can be disabled by deploying a new VMF-FG without *pip* VMF as depicted in Fig. 3. Alternatively, a controller can be used to send commands (e.g. ENABLE or DISABLE) to *pip* VMF such that its output either (i) contains *scaler* output embedded in *chrm-key* output (ENABLE) or (ii) just the *chrm-key* output. This paper aims to demonstrate efficient resource usage due to decomposition, which is possible through a simplified modeling approach. Therefore, we focus on deployment of media services in the first scenario (Fig. 3 (a) and (b)) and the second scenario (Fig. 3 (c)) will be considered in future work.

The deployment of media services in a MFV environment entails two types of mappings. First is the assignment of VMFs in the service's VMF-FG to the MFVi server nodes and second is the provisioning of physical-paths to virtual links of the VMF-FG.

IP-based transport in broadcast studios allows decomposition of high-resolution video streams into multiple low-resolution streams, resulting in various opportunities that can be exploited to optimize the VMF-FG mapping as discussed next.

B. VMF-FG Decomposition

There are numerous VMFs that require operation only in a sub-region of the video frame and the rest of frame contains sub-regions from VMF's inputs (e.g. logo-insertion and picture-in-picture). Therefore, when using decomposed sub-streams, these VMFs only need to process a subset of all input's sub-streams. For example, consider pip VMF of the media service shown in Fig. 4 (a) and assume both of its input video streams (s_0, s_1) are decomposed into $M(= 4)$ sub-streams ($\{s_{q0}, s_{q1} \mid \forall q \in [0, \mathbb{N}_{M-1}]\}$) corresponding to the four quadrants (sub-regions) of the frame. With stream decomposition, the original *pip* VMF can be replaced by four smaller pip VMFs, which operate only on the sub-streams corresponding to the four quadrants, see Fig. 4 (b). Except for pip_1 VMF, all other VMFs just pass input frames to its output; thus, do not require any operation on its input. As a result of this decomposition, removing pip_0, pip_2, pip_3 from the VMF-FG results in a functionally equivalent VMF-FG or media service. Next, we discuss an algorithm to obtain the VMF-FG after decomposing media streams in the original

VMF-FG by M smaller sub-streams.

The pseudo-code of the VMF-FG decomposition algorithm for a given value of the decomposition parameter M is shown in the procedure dcmpVMFFG (Alg. 1). Assume, the original VMF-FG of the service is represented by $\mathcal{G} = (\mathcal{F}, \mathcal{L})$. The endpoints (sources $\mathcal{F}^{src}$ and destinations $\mathcal{F}^{dst}$) of the VMF-FG are stored in $\mathcal{F}^{endpts}$ (ln. 27). Then, the nodes ($\mathcal{F}_{dcmp}$) and the links ($\mathcal{L}_{dcmp}$) for the decomposed VMF-FG are initialized with ϕ (ln. 28). We make M copies of all the VMFs in $\mathcal{F}$ and add them to $\mathcal{F}_{dcmp}$ (ln. 29-31). Next, for each virtual link $(vmf_i, vmf_j) \in \mathcal{L}$ of the original VMF-FG, a set of corresponding virtual links in the decomposed VMF-FG is obtained using procedure linksVMF (ln. 32-34). Depending on the type of VMF f_j, a specific set of links that corresponds to the upstream VMFs required as an input for the decomposed VMFs of f_j, is obtained. For example, corresponding to the link (f_3, f_5) in Fig. 3 (a) and $M = 4$, linksVMF returns the set $\{(f_{03}, f_{05}), (f_{13}, f_{15}), (f_{23}, f_{25}), (f_{33}, f_{35})\}$ and for (f_4, f_5) the returned set of links is $\{(f_{04}, f_{15}), (f_{14}, f_{15}), (f_{24}, f_{15}), (f_{34}, f_{15})\}$ as shown in Fig. 5.

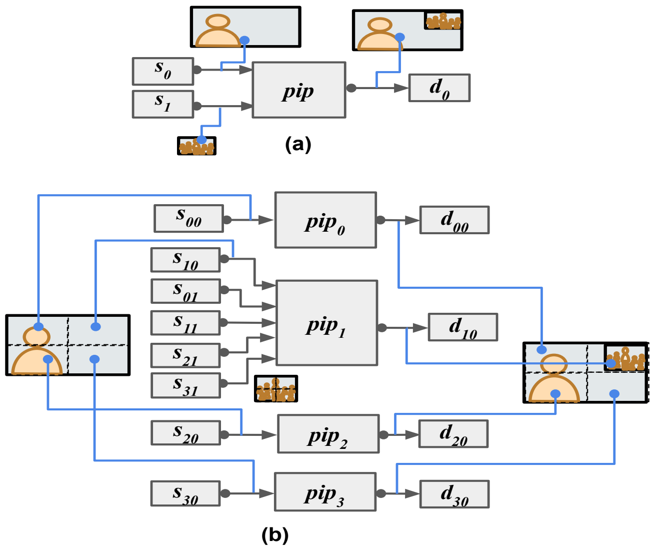

Fig. 4. 4-way Decomposition of the pip VMF.

The rest of the procedure is responsible for optimizing the decomposed VMF-FG by pruning the non-operational VMFs and links from $\mathcal{G}_{dcmp}$ using pruneVMFFG (ln. 35).
For each VMF f in $\mathcal{F}$, all the downstream VMFs, i.e. VMFs whose inputs are connected to f's output are stored in $\mathcal{F}^{nxt}$ (ln. 17). If f's output is not connected to any other VMF, f is removed from $\mathcal{F}$ using removeVMF that also prunes the whole branch recursively ($recursive$ = **True**) comprising of VMFs and links only feeding f (ln. 3-9).
After returning from removeVMF in pruneVMFFG, it is checked, using the function isBypassable, whether a VMF f can be bypassed or not (ln. 21). For a VMF like pip, if its inputs are only connected to a single (type) upstream VMF's output, no processing is required in this VMF; as input frames are just passed to the VMF's output without any modification. For such VMF instances, it is required to re-arrange the corresponding links of $\mathcal{G}$; doBypass calls removeVMF non-recursively ($recursive$ = **False**) to do so (ln. 13).
The resulting VMF-FG $\mathcal{G}_{dcmp}$ is functionally similar to the original $\mathcal{G}$. For the media service example discussed in Fig. 3, the decomposed VMF-FG $\mathcal{G}_{dcmp}$ obtained using dcmpVMFFG is shown in Fig. 5.

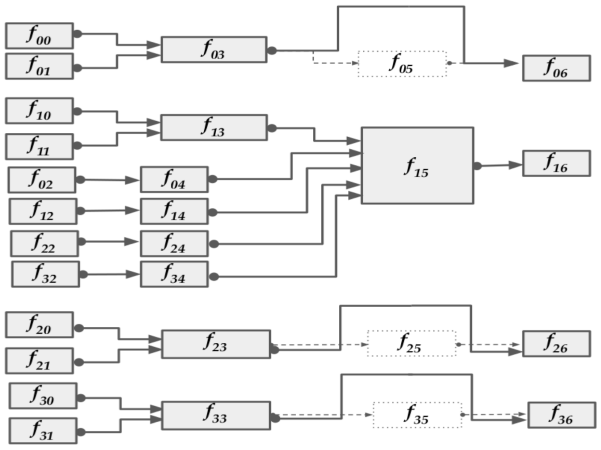

Fig. 5. VMF-FG decomposition.

C. VMF Placement and Chaining

In this section, we discuss a first-fit based heuristic FFPlaceChain that is used to map a media service VMF-FG (or decomposed VMF-FG) to a given MFVi.

1) Heuristic:

The pseudo-code for FFPlaceChain is shown in Alg. 2. The parameters of the procedure are the media service s, s's VMF-FG (un-decomposed or decomposed) $\mathcal{G} = (\mathcal{F}, \mathcal{L})$, the set of media sources $\mathcal{F}^{src}$, the set of destinations $\mathcal{F}^{dst}$, the graph representation of the MFVi network $G^I = (N, E)$, resource demands (e.g., VMF CPU, physical link bandwidth) Dem of s, and resource capacities (e.g., node CPU, virtual link bandwidth) Cap of the MFVi. First, we initialize variables α (VMF placement map), β (VMF acceleration map), and γ (VMF chain map) each with ϕ (ln. 14). Another variable $usedRes$ (vector), denoting resources used so far in the current FFPlaceChain call, is initialized with zero (ln. 15).
Then, the nodes in $\mathcal{F}$ that correspond to traffic sources ($\mathcal{F}^{src}$) and sinks ($\mathcal{F}^{dst}$) are stored in $\mathcal{F}^{endpts}$. These endpoints are then attached to switches in a network; here they are attached to the data center access-switches N_{axs} (ln. 17-20).
For the assignment of VMFs to the server nodes, the procedure

Algorithm 1: VMF-FG Decomposition Procedure.

```
1  Procedure removeVMF ($f$, $\mathcal{G}' = (\mathcal{F}', \mathcal{L}')$, $recursive$):
2      $\mathcal{G} = (\mathcal{F}, \mathcal{L}) \leftarrow \mathcal{G}'$;
3      if $recursive$ then
4          for $(f_i, f)$ in $\{(f_k, f) \mid \forall (f_k, f) \in \mathcal{L}\}$ do
5              if not $\exists (f_i, f_j) \in \mathcal{L} \setminus \{(f_i, f)\}$ then
6                  $\mathcal{G} \leftarrow$ removeVMF ($f_i$, $\mathcal{G}$, True);
7              end
8          end
9      end
10     return $\mathcal{F} \setminus \{f\}$,
           $\mathcal{L} \setminus [\{(f_i, f) \mid \forall (f_i, f) \in \mathcal{L}\} \cup \{(f, f_j) \mid \forall (f, f_j) \in \mathcal{L}\}]$;
11 end
12 Procedure doBypass ($f$, $\mathcal{G} = (\mathcal{F}, \mathcal{L})$):
13     return removeVMF ($f$,
           $(\mathcal{F}, \mathcal{L} \cup \{(f_i, f_j) \mid \forall (f_i, f), (f, f_j) \in \mathcal{L}\})$, False);
14 end
15 Procedure pruneVMFFG ($\mathcal{G} = (\mathcal{F}, \mathcal{L})$):
       /* VMF-FG optimization                                  */
16     for $f$ in $\mathcal{F}$ do
17         $\mathcal{F}_{nxt} \leftarrow \{f_j \mid \forall (f, f_j) \in \mathcal{L}\}$ ;
           /* remove dangling branches from
              VMF-FG                                            */
18         if $\mid \mathcal{F}_{nxt} \mid == 0$ and $f \notin \mathcal{F}^{dst}$ then
19             $G \leftarrow$ removeVMF ($f$, $(\mathcal{F}, \mathcal{L})$, True);
20         end
21         if isBypassable ($f$, $\mathcal{G}$) then
               /* bypass VMFs in VMF-FG                         */
22             $\mathcal{G} \leftarrow$ doBypass ($f$, $\mathcal{G}$);
23         end
24     end
25 end
26 Procedure dcmpVMFFG (($\mathcal{G} = \mathcal{F}, \mathcal{L}$), $\mathcal{F}^{src}$, $\mathcal{F}^{dst}$):
27     $\mathcal{F}^{endpts} \leftarrow \mathcal{F}^{src} \cup \mathcal{F}^{dst}$;
28     $\mathcal{G}_{dcmp} = (\mathcal{F}_{dcmp}, \mathcal{L}_{dcmp}) \leftarrow \phi, \phi$;
29     for $f$ in $\mathcal{F}$ do
30         $\mathcal{F}_{dcmp} \leftarrow \mathcal{F}_{dcmp} \cup \{f_q \mid \forall q \in [0, \mathbb{N}_{M-1}]\}$;
31     end
32     for $(f_i, f_j)$ in $\mathcal{L}$ do
           /* set returned by linksVMF depends
              on $f_j$.                                         */
33         $\mathcal{L}_{dcmp} \leftarrow \mathcal{L}_{dcmp} \cup$ linksVMF ($\mathcal{G}$, $f_i$, $f_j$, $M$);
34     end
35     return pruneVMFFG (($\mathcal{F}_{dcmp}$, $\mathcal{L}_{dcmp}$));
36 end
```

Algorithm 2: First-fit VMF-PC procedure.

```
1  Procedure FirstFit ($s$, $f$, $\alpha$, $\mathcal{G} = (\mathcal{F}, \mathcal{L})$, $G^I$, $Cap$,
       $Dem(f)$, $N$, $checkAcc$):
2      if $checkAcc$ and not (atype ($f$) in $A$) then
3          return None, False, None;
4      end
5      for $n$ in $N$ do
6          $links_f \leftarrow \{(f_i, f) \mid \forall (f_i, f) \in \mathcal{L}\}$ ;
7          $paths_{map} \leftarrow$ chainVMFs ($s$, $f$, $links_f$, $G^I$,
               $node_{sel}$, $Cap$, $Dem(f)$, $\alpha$);
8          if enoughResources ($Dem(f)$, $Cap(n)$
               $checkAcc$) and $paths_{map} \neq$ None then
9              return $n$, $checkAcc$, $paths_{map}$;
10         end
11     end
12     return None,False, None;
13 Procedure FFPlaceChain ($s$, $\mathcal{G} = (\mathcal{F}, \mathcal{L})$, $\mathcal{F}^{src}$, $\mathcal{F}^{dst}$,
       $G^I = (N, E)$, $Dem$, $Cap$):
14     $\alpha, \beta, \gamma \leftarrow \phi, \phi, \phi$;
15     $usedRes \leftarrow 0$;
16     $\mathcal{F}^{endpts} \leftarrow \{f \mid \forall f \in (\mathcal{F}^{src} \cup \mathcal{F}^{dst})\}$;
       /* $N_{axs}$ is a set of access-switches   */
17     for $f$ in $\mathcal{F}^{endpts}$ do
18         $n \leftarrow$ a node in $N_{axs} \subset N$ with lowest no. of
               endpoints placed;
19         $\alpha[s, f] \leftarrow n$;
20     end
21     for $f$ in $(\mathcal{F} \setminus \mathcal{F}^{endpts})$ do
22         $node_{sel}, accel_f \leftarrow$ None, False;
           /* $N_{acc} \subset N_c$ is set of all
              computation nodes with
              accelerators                          */
23         $node_{sel}, accel_f, path_{map} \leftarrow$ FirstFit ($s$, $f$, $\alpha$,
               $\mathcal{G} = (\mathcal{F}, \mathcal{L})$, $G^I$, $Cap - usedRes$, $Dem(f)$,
               $N_{acc}$, True);
24         if $node_{sel}$ == None then
25             $node_{sel}, accel_f, path_{map} \leftarrow$ FirstFit ($s$, $f$,
                   $\alpha$, $\mathcal{G} = (\mathcal{F}, \mathcal{L})$, $G^I$, $Cap - usedRes$,
                   $Dem(f)$, $N_c$, False);
26         end
27         if $node_{sel} \neq$ None then
28             $\alpha[s, f], \beta[s, f] \leftarrow node_{sel}, accel_f$;
29             $\gamma[s, \{(f_i, f) \mid \forall (f_i, f) \in \mathcal{L}\}] \leftarrow path_{map}$;
30             $usedRes \leftarrow usedRes + Dem(f)$ ;
31         end
32         else
33             return $\phi$, $\phi$, $\phi$;
34         end
35     end
36     UpdateRes ($Cap$, $usedRes$);
37     return $\alpha$, $\beta$, $\gamma$;
38 end
```

`FirstFit` is used. First, the server node assignment of f along with the accelerator allocation is attempted by passing the arguments– current resource capacities ($Cap - usedRes$), VMF demands ($Dem(f)$), $N = N_{acc}$ (server nodes with hardware accelerators) and $checkAcc$ = **True** in addition to the other arguments (ln. 23). If there is no accelerator for VMF f in A (set of all available accelerator implementations), `FirstFit` terminates and returns **None**, **False**, **None** (ln. 3). Otherwise, `FirstFit` iterates through the set N_{acc} until it finds a node that has enough resources to accommodate f; using `chainVMFs`, it is also checked if VMFs connected to f's inputs can be chained to it. If enough bandwidth is available for chaining, the virtual link ($links_f$) to physical path mapping is returned to $paths_{map}$ (ln. 7). In case no suitable node is found by `FirstFit`, **None**, **False**, **None** is returned.

If no suitable server node is found in the previous step, another call to `FirstFit` is made to get a suitable node without hardware accelerator allocation by passing $N = N_c$ (all server nodes) and $checkAcc$ = **False** (ln. 24-26).
If f has still not been assigned to a server node, `FFPlaceChain` returns ϕ, ϕ, ϕ (ln. 33); otherwise, the mapping variables (α, β and γ) are updated (ln. 28-29). Also, $usedRes$ is incremented by VMF demands $Dem(f)$

(ln. 30).
After the completion of VMF placement and chaining, the resource capacities are updated using `UpdateRes` before `FFPlaceChain` returns the mapping variables (ln. 36-37).

IV. Evaluation

In this section, we first describe the simulation environment and then present the evaluation results. The physical network used for evaluations is fat-tree data center topology composed of k, where pod each pod contains $(k/2)^2$ server nodes, $k/2$ access layer switches, and $k/2$ aggregate layer switches and the core layer contain $(k/2)^2$ core switches. Therefore, there are $k^3/4$ total number of server nodes in the data center. The simulation parameters for media service requests, MFVi and VMFs are listed in Table II.

TABLE II
Default values/range of various parameters involved in simulation experiments.

Parameter	Value or range
Request arrival rate (Poisson)	3/(100 units)
Request lifetime (Exponential)	1000 units
VMF CPU per Mbps of input	6.2 cores/(100 Mbps)
VMF CPU reduction per Mbps of input	4 cores/(100 Mbps)
Video resolution	1920x1080p30 (4:2:2 sampling, 10bits/sample)
Physical node CPU	24 cores
Physical Link BW	10Gbps

The simulation is carried on Intel Xeon machine @2.40GHz and 12GB RAM. We repeated each experiment ten times and average over all iteration is reported along with corresponding standard deviation. Each media service request has an associated VMF-FG. For each request, we choose one VMF-FG randomly from the set of VMF-FGs ($\{\mathcal{G}_1, \mathcal{G}_2, \mathcal{G}_3\}$) shown in Fig. 6.

A. VMF-FG Decomposition

Before discussing the results pertaining to the VMF-PC heuristic, we first report the impact of decomposition on CPU and bandwidth requirements of each of the three VMF-FGs.

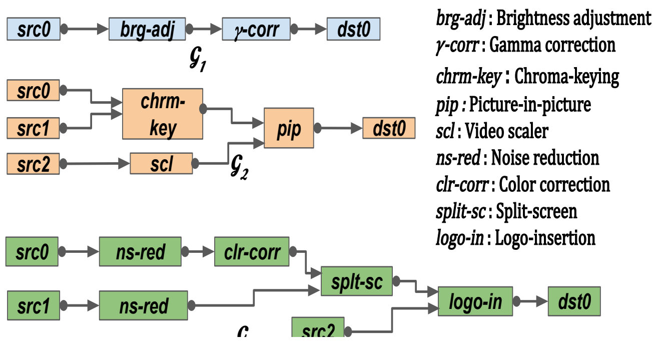

Fig. 6. Set of VMF-FGs used for evaluations.

Normalized CPU (cpu_{nrm}) and total bandwidth (bw_{nrm}) requirements of the three VMF-FGs are shown in Fig. 7 (a) and (b), respectively; where, cpu_{nrm} and bw_{nrm} are defined as follows:

$$\begin{aligned} cpu^s_{nrm}(M) &= \frac{\sum\limits_{vmf \in \mathcal{F}^s_{dcmp}(M)} cpu_{vmf}}{\sum\limits_{\forall vmf \in \mathcal{F}^s_{dcmp}(M=1)} cpu_{vmf}} \\ bw^s_{nrm}(M) &= \frac{\sum\limits_{(\forall (vmf_1, vmf_2) \in \mathcal{L}^s_{dcmp}(M)} bw_{vmf_1, vmf_2}}{\sum\limits_{\forall (vmf_1, vmf_2) \in \mathcal{L}^s_{dcmp}(M=1)} bw_{vmf_1, vmf_2}} \end{aligned} \quad (1)$$

It can be observed that with the increasing M, cpu_{nrm} and bw_{nrm} decreases, except for $\mathcal{G}_1$ where there is no improvement. No improvement for $\mathcal{G}_1$ is because the same number of operations are required whatever the value of M is, as no decomposed VMF is bypassed or any VMF branch is pruned. For $\mathcal{G}_2$ major improvement are due to the decomposition of *pip* VMF. Similarly, initial improvement ($M = 4, 16$) for $\mathcal{G}_3$ corresponds to the reduction number in the total number of VMF operations (pixels processed by all VMFs) required to implement split-screen functionality after the decomposition.

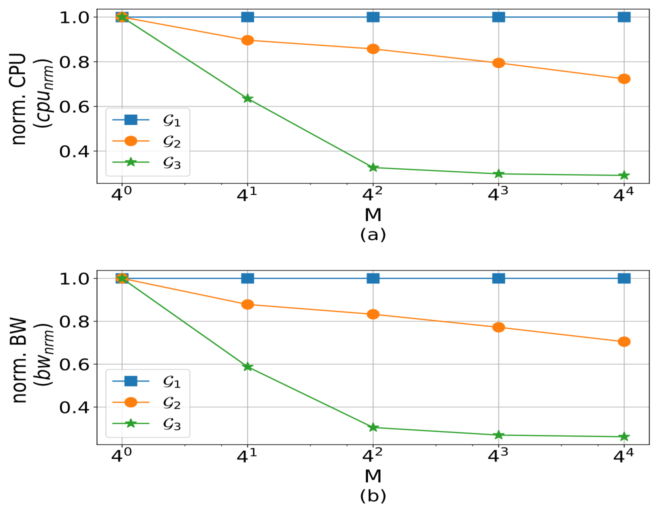

Fig. 7. Variation of normalized required CPU cores and bandwidth for the three VMF-chains in the function of the decomposition parameter M.

B. Deployment

Here, we first report the average acceptance ratio of media service requests over time with data center topology having $k = 4$. The arrival of requests is modeled as the Poisson process with an average arrival rate of 3 requests per 100 time units (tu) and the lifetime of requests is exponentially distributed with an average of 1000 tu. A media service request is accepted if VMF-PC completes successfully, i.e. all VMFs are assigned a server node and all virtual links have been mapped to physical paths. Acceptance ratio is defined as the total number of requests successfully deployed to the total number of requests received. Fig. 8 shows the variation

of average acceptance ratio with time for different value of M. First, the acceptance ratio on an average decreases over time as fewer and fewer resources are available for new requests. Also, these results clearly indicate that acceptance ratio improves with M; starts dropping at t=700tu for $M = 1$, t=1200tu for $M = 4$, t=4800tu for $M = 256$, etc. This improvement is a result of the reduction in the resource demands of media service's VMFs with increasing M (section IV-A).
We also report total server nodes (N_{usd}) used in a given data center topology ($k = 8$) over time. We considered a bigger data center topology as compared to our last evaluation ($k = 4$) as we wanted to reduce deviations at small values of N_{usd}. Fig. 9 depicts total used server nodes with time for different values of M. The reduction of N_{usd} with M can be similarly explained by reasons discussed in section IV-A.

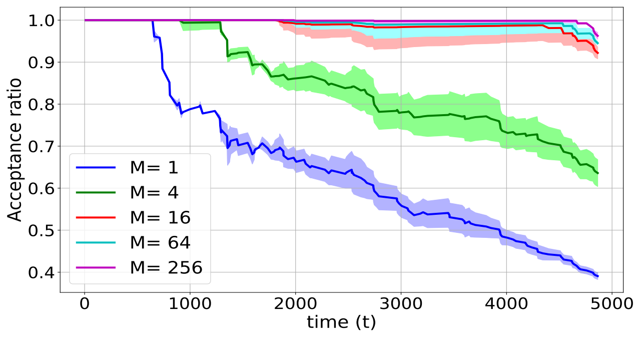

Fig. 8. Acceptance ratio over time for deployment of 150 media services values of M. Data center topology with 16 server-nodes ($k = 4$).

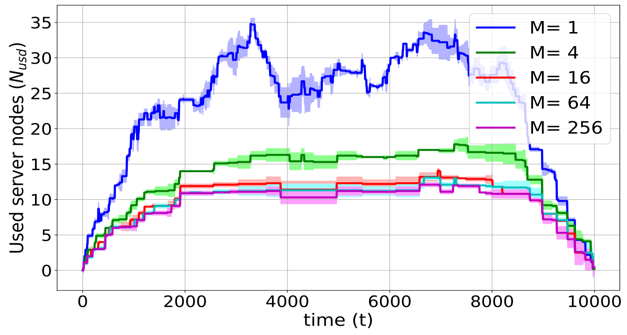

Fig. 9. Used server nodes to deploy 200 media services over time. Data center topology with 128 server-nodes ($k = 8$).

V. Conclusion and Future Work

Lately, broadcasters are adopting IP networking to transport media streams in production studios. Furthermore, using COTS hardware, as an alternative to specialized hardware, to process media streams has also been gaining increasing attention. IP-based transport and virtualized processing of media streams open up many opportunities for cost-efficient deployment of services. To this end, we proposed an algorithm that optimizes the VMF-FG for static deployment of media services. Also, we presented a first-fit based heuristic for VMF-PC. The evaluation results show a significant improvement in request acceptance ratio and server node utilization can be achieved using VMF-FG decomposition.
Future work includes modeling of VMF-PC using ILP with which optimal media service deployments can be accomplished. Furthermore, the VMF-FG decomposition process can be generalized to obtain the optimized VMF-FG for dynamic media service deployments.

VI. Acknowledgment

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.

References

[1] M. Fremeije, "The Rising Need for Media Function Virtualization," RedHat, Tech. Rep., 02 2018.
[2] "The Road to COTS and the Cloud for real-time broadcast production," Nevion, Tech. Rep., 01 2018.
[3] K. Paulsen, "Prepping for the IP Transition," Dell EMC, Tech. Rep., 01 2017.
[4] J. G. Herrera and J. F. Botero, "Resource allocation in NFV: A comprehensive survey," *IEEE Transactions on Network and Service Management*, vol. 13, no. 3, pp. 518–532, 2016.
[5] T. Koyama, J. Kawamoto, M. Kawaragi, T. Kurakake, and K. Saito, "Implementing 8k vision mixer for cloud-based production system," *SMPTE Motion Imaging Journal*, vol. 128, no. 6, pp. 30–37, 2019.
[6] T. Kojima, J. J. Stone, J. Chen, and P. N. Gardiner, "A Practical Approach to IP Live Production," in *SMPTE 2014 Annual Technical Conference Exhibition*, 2014, pp. 1–16.
[7] "ST 2110-10:2017 - SMPTE Standard - Professional Media Over Managed IP Networks: System Timing and Definitions," *ST 2110-10:2017*, pp. 1–17, 2017.
[8] "ST 2110-20:2017 - SMPTE Standard - Professional Media Over Managed IP Networks: Uncompressed Active Video," *ST 2110-20:2017*, pp. 1–22, 2017.
[9] "ST 2110-30:2017 - SMPTE Standard - Professional Media Over Managed IP Networks: PCM Digital Audio," *ST 2110-30:2017*, pp. 1–9, 2017.
[10] "ST 2110-40:2018 - SMPTE Standard - Professional Media Over Managed IP Networks: SMPTE ST 291-1 Ancillary Data," *ST 2110-40:2018*, pp. 1–8, 2018.
[11] "ST 2022-6:2012 - SMPTE Standard - Transport of High Bit Rate Media Signals over IP Networks (HBRMT)," *ST 2022-6:2012*, pp. 1–16, 2012.
[12] "RP 2110-23:2019 - SMPTE Recommended Practice - Single Video Essence Transport over Multiple ST 2110-20 Streams," *RP 2110-23:2019*, pp. 1–27, 2020.
[13] "Broadcast Video Infrastructure Implementation Using FPGAs," Altera, Tech. Rep., 03 2007.