Comparing Topology and Stream Based Strategies for Modeling Service Function Chains

Hendrik Moens and Bruno Volckaert
Ghent University – iMinds, Department of Information Technology
iGent, Technologiepark-Zwijnaarde 15, 9052 Ghent, Belgium
e-mail: hendrik.moens@intec.ugent.be

Abstract—**Using Network Functions Virtualization (NFV), network functionality can be virtualized and deployed on generic servers. This increases network flexibility, and makes it possible to allocate network services in multiple locations throughout the network. Virtual Network Functions Placement (VNF-P) solves the problem of where network functions must be allocated within the network, and makes use of the increased flexibility of NFV networks to improve service quality or lower allocation costs. An important consideration when designing VNF-P solutions is the network service representation, as this representation can both impact the complexity of the specified services and the complexity of the management system managing and deploying them. In this paper, we define and analyze three different strategies for representing NFV-based network services: topology-based, extended topology-based and stream-based. Using three video streaming scenarios, we evaluate these approaches, comparing their expressiveness, their development complexity, and the management complexity they result in. We find that the stream-based approach results in the highest flexibility and expressiveness.**

I. Introduction

Network Functions Virtualization (NFV) is an upcoming paradigm where network functionality is virtualized and split up into multiple building blocks chained together using Service Function Chains (SFCs) to provide a network service. Using NFV, network functions can be migrated from costly, often dedicated hardware appliances to Virtual Network Functions (VNFs), dynamically allocated virtualized instances deployed on generic servers using cloud technologies. The network in which NFV-based services are deployed can contain multiple cloud, edge cloud and hardware nodes where VNFs can be deployed. Depending on where an SFC's VNFs are deployed, the quality of the service, and the cost with which it is offered can vary. Using remote clouds may e.g. result in a higher service latency, while edge clouds would instead result in a higher quality but more expensive service, as edge clouds are smaller and therefore benefit less from economies of scale. Therefore, research into VNF Placement (VNF-P) is required, in order to automate the decision of where to deploy VNFs within the network given a set of SFCs.

In this paper, we analyze three different strategies for modeling SFCs, each resulting in different VNF-P requirements: Topology-Based Modeling (TBM), eXtended Topology-Based Modeling (XTBM) and Stream-Based Modeling (SBM). TBM uses an intuitive approach, where the various network functions are defined and interconnected using a service topology

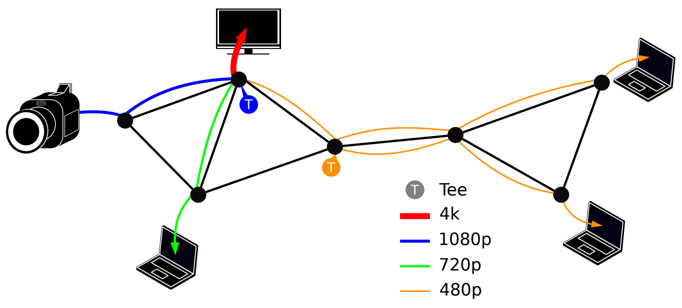


Fig. 1: A media streaming service deployed within a network. Video from a single source must be delivered to multiple network nodes at differing quality levels. To achieve this, the media streams must be transcoded and replicated.

graph, and focuses specifically on the VNFs and their interconnections. This approach is commonly used when managing multi-component applications in clouds [1], [2] and when designing applications composed using web services [3]. XTBM extends this approach, adding scenario-specific functionality to reduce the complexity of the defined SFCs. SBM approaches SFC modeling differently, by focusing on the content of the source and destination network flows. The source and destination network streams are defined, and in addition various intermediate stream converters are specified. In this approach, no specific service topology is imposed, and instead, the service is chained together at runtime by converting the stream generated by the source using various intermediate components which may be located throughout the network, and routing the resulting stream to the destination node.

The difference between these modeling approaches is the viewpoint they convey: TBM and XTBM focus on the definition of services and their interaction, while SBM focuses on the different types of streams that are present within the network. We analyze these approaches from a development, management and expressiveness perspective. While these modeling approaches are applicable to all types of SFCs, we focus on an NFV-based media streaming context, where a media flow is generated, transmitted over the network, and transformed using multiple VNFs, as this is an area where traditional SFC modeling approaches fall short. The transformations applied to the media content include transcoding, upscaling, downscaling, encryption and Tee-ing (duplicating the flow to transmit it to multiple destinations). An example media streaming scenario is illustrated in Figure 1.

The remainder of this paper is structured as follows. The next section discusses related work. Section III elaborates on the VNFs needed to support media streaming. In Section IV, TBM, XTBM and SBM are presented in-depth, followed by a discussion on how conversion between these SFC representations can be achieved in Section V. In Section VI the approaches are analyzed using three media-based transmission scenarios. The results are discussed in Section VII, followed by our conclusions in Section VIII.

II. Related work

The NFV Management and Orchestration (MANO) specification [4] supports the definition of a VNF Forwarding Graph Descriptor (VNFFGD). This allows the definition of a SFCs by specifying the VNFs it's composed of. The VNFFGD modeling approach is closely related to TBM which we analyze in-depth in this paper, and which we compare to alternative approaches.

Multiple recent works have focused on solving the (network) function placement problem in NFV networks to allocate SFCs within the network [5], [6], [7]. In this paper, we however focus specifically on comparing multiple alternative SFC specification approaches, and not on how they are allocated, evaluating the impact of the chosen SFC modeling approach on expressiveness, optimality and management complexity.

In [5], a network function specification and placement approach is introduced. To specify SFCs, a context-free language is introduced which supports flexible definition of SFCs as a collection of modules. While the presented approach offers increased flexibility in the specification of SFCs, allowing the re-ordering of VNFs, it does not allow the specification of multi-destination flows which are needed in a multimedia streaming context. In addition, the SFC modeling approach in [5] results in a significantly more complex management system that is specialized to handle these specific SFCs. In this paper, we by contrast compare three alternative SFC specification approaches where TBM and SBM allow for the use of a highly generic management system and where XTBM and SBM allow (limited) reordering of VNFs.

In [6], the authors introduce an approach for allocating virtualized Deep Packet Inspection (DPI) engines within communication flows using a service representation similar to SBM introduced in this paper. The SFC model and management system in [6] are however specifically designed and tailored for the virtualized DPI scenario, while they have been generalized in this paper, making them applicable for generalized SFCs.

In our previous work [7], we introduced an approach for SFC resource allocation in hybrid NFV networks. SFCs are modeled using TBM, which allows for the definition and allocation of a large number of network services, but which is not always suitable when considering complex media streaming scenarios. In this paper, we focus specifically on alternative SFC modeling approaches, analyzing how TBM can be extended to XTBM and how these approaches compare to SBM in a media streaming context.

Resource allocation in NFV networks has some similarities to application placement approaches used within datacenters and clouds [8], specifically to network-aware application placement, where a collection of services is deployed within the network. It is similarly related to the problem of virtual network embedding in software defined networks [9], which focuses on how virtual network requests, in the form of a collection of Virtual Machines (VMs) and their interconnections can be deployed on physical networks. Both network-aware cloud application placement and virtual network embedding approaches however make use of service topologies, similar to TBM discussed in this paper, which can be inflexible when a service can be created using multiple topologies. This paper by contrast introduces two additional approaches, XTBM and SFC, and compares it to the more traditional TBM, finding that these more flexible SFC modeling approaches are more suited when specifying multimedia streaming SFCs.

III. NFV media streaming

Before formalizing TBM, XTBM and SBM, we first define the various virtual and physical network functions that must be provided to fully support media streaming scenarios. Some of these functionalities are specific to the media streaming context, while others are more generic:

- **Video source:** This service generates a video stream and transmits the stream to other services within the network. In the envisioned media service, this is a live camera, but this could also be a media server.
- **Transcoders:** These services decode an incoming video stream, transform, and encode it. A transcoder may be used to scale video sources up or down, changing their resolution (in which case it is referred to as an *upscaler* or a *downscaler* respectively), and/or it may change the encoding of the format (e.g. h264 to h265).
- **Video tranformation:** The video stream can also be transformed in other ways, e.g. by adding watermarks and filters.
- **Tees:** A Tee service is used to replicate a single incoming video stream, transferring it to multiple destinations. A Tee service may have a limited number of outputs, and may be designed to only handle specific stream types.
- **Compositor:** A compositor service composes multiple incoming video streams into a single composed stream, which it transfers to a single destination. Each compositor can generally only handle a specific set of input file formats, and may also have a limited number of inputs.
- **Encryption and decryption:** As the video streams may contain sensitive information, it may be needed to encrypt them before transmitting them over public network links.
- **Network address translation:** In some cases, the video stream is transmitted from/to a private network without a public IP. In this case, a Network Address Translation (NAT) service is needed to translate the private IP to a public IP.

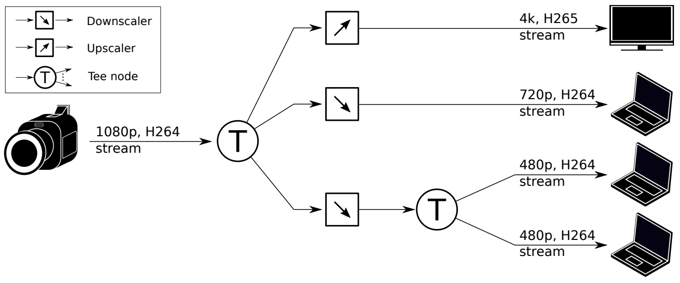


Fig. 2: A service topology in a live video streaming context.

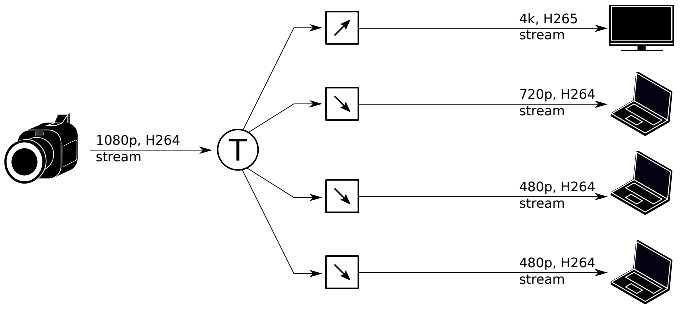


Fig. 3: An alternative service topology resulting in the same network service as the topology in Figure 2.

IV. NFV SERVICE MODELING

When defining and allocating services composed out of VNFs, multiple approaches for defining the SFC can be used. In this section, we formalize two alternative service representation models that can be used to define SFCs, TBM and SBM, and present XTBM based on the former. For each of these approaches, we describe multiple properties:

1) **Specification correctness:** When considering the specification correctness of an SFC modeling approach, we analyze whether it is possible to validate if network services specified using this approach are feasible, unambiguous and correct.
2) **Deployment variability:** When analyzing the deployment variability of an NFV service modeling approach, we determine whether and how it is possible to model SFCs that can be deployed in multiple alternative ways.
3) **Specification dynamicity:** SFC specifications may change over time. When considering specification dynamicity we analyze how deployed SFCs can be changed over time, e.g. when new streaming destinations are added.
4) **Management system requirements:** The chosen SFC specification can impact the complexity of the required management system used to deploy network services. It may e.g. be possible to adapt existing management approaches to manage SFCs.

A. Topology-based service modeling

A simple approach for modeling services composed out of multiple NFV components is by using service topologies. In this approach, all services are represented using nodes, which are interconnected using a graph. Figure 2 illustrates a simple service topology showing a media application containing a single video source which is replicated using a *Tee* node and sent to multiple end user clients.

Formally, a service topology consists of a collection of service nodes N and a collection of edges E between nodes. When an edge between two nodes $(n_1, n_2) \in E$ exists, the services n_1 and n_2 are connected. In an NFV scenario, these edges can either be unidirectional (connecting a source to a destination for e.g. streaming purposes) or bidirectional (allowing peer to peer communication).

1) Specification correctness: The completeness and strictness of the specified service has the advantage of being unambiguous as to how the various VNFs interconnect to provide a network service: the service can only be deployed exactly as specified by the service topology. This makes the definition of TBM SFCs highly intuitive, making it easy to specify SFCs, for developers to reason using TBM SFCs, and to debug them. Therefore this approach is highly suited for many types of NFV service specifications, especially as many SFCs present themselves in the form of linear service chains without branching.

2) Deployment variability: The main weakness of TBM is that it starts to break down when the service can be deployed in multiple different ways. Figure 3 shows an alternative topology providing the same network service as that of Figure 2. Depending on the structure of the underlying network and the location of computational nodes within this network, these different service topology permutations can significantly impact the cost and quality of the provided service. While, in practice, many of these permutations will lead to services that are clearly inefficient, significantly reducing the number of relevant permutations, this requires runtime information of the underlying network.

This problem is significantly exacerbated by the presence of Tee nodes, but multiple other service requirements may also increase the possible number of service topologies to achieve a given result. Each of the VNFs listed in Section III, except for the video source and destination, can be located at different locations in the network, making it possible to chain them together using different topologies. When the number of different configurations is controllable and limited, an approach where a restricted set of possible service topologies is provided could be used.

3) Specification dynamicity: A second limitation of TBM can be observed when the service topology changes over time. In a media streaming scenario, multiple end users can join and leave over time, and video sources can also change over time (e.g. switching to a different camera). This necessitates changes to the deployed SFC, by adding or removing some of the VNFs, and would therefore result in the construction of a new topology specification that must be deployed. The management system must then deploy this new topology, and it must be able to enact this change without disrupting the already running service.

4) Management system requirements: When deploying TBM SFCs, the topology specification must be mapped to

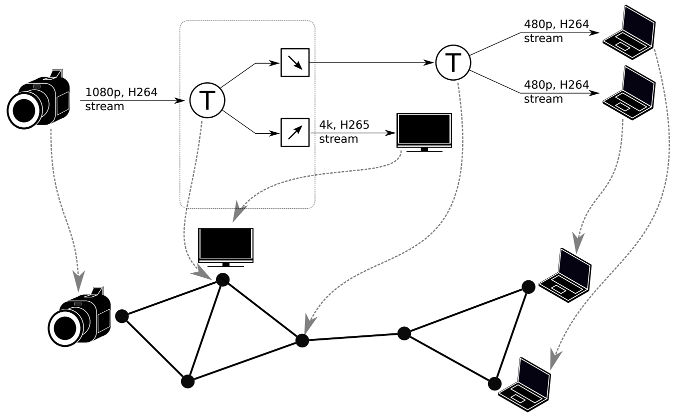


Fig. 4: Deploying and allocating resources for TBM SFCs.

physical nodes within the network, as illustrated in Figure 4. This is a straightforward resource allocation problem which is closely related to existing cloud [10], [11], network [1], [2] and NFV [7] resource allocation problems. An additional management system benefit is that the VNFFGD SFC specification defines a topology, making TBM compatible with existing NFV standards and current research directions.

Finally, to support the discussed media streaming in a dynamic scenario, the management system must also be able to re-configure deployed services as mentioned above, either by modifying deployed topologies, or by removing and re-adding deployed topologies without service interruptions.

B. Extended TBM SFC modeling

An extension to TBM can be created by adding awareness of Tee nodes to the model. A single logical super-Tee node could then be defined within the model which can have an unlimited number of branches. The management system can then map this logical Tee node to an arbitrary multicast tree, consisting of multiple virtual and physical Tee nodes. In this case, the problem of multicast tree permutations is mitigated, at the cost of a more complex management system, which must be made aware of Tee node properties.

Similarly, awareness of the concepts of the various VNFs listed in Section III could be added to the management system. This would result in simple service topologies, but would also require a complex management system which must be aware of the specifics of various scenario-dependent VNFs.

These modifications improve the expressiveness and flexibility of XTBM compared to TBM, but also increases the complexity and specificity of the management system deploying the services, as it must be aware of additional, scenario-specific functionalities (e.g. Tee nodes, encryption etc.).

C. Stream-based service modeling

SBM moves the focus from the service topology to the components themselves and the streams interconnecting them. Formally, we define a collection of service nodes N, and a collection of flow types F. Service nodes can either be a source $s \in S$, a destination $d \in D$, or a converter $c \in C$,

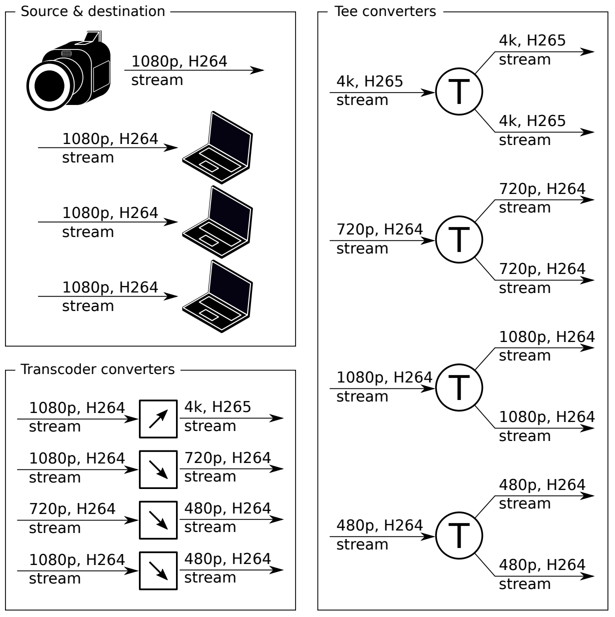


Fig. 5: An SBM SFC.

where $N = S \cup D \cup C$. Every source $s \in S$ has one or more outgoing flows $s^{out} \subseteq F$. Similarly every destination $d \in D$ has an incoming flow $d^{in} \subseteq F$. Finally, a converter node $c \in C$ defines both incoming and outgoing flows $c^{in} \subseteq F$ and $c^{out} \subseteq F$ respectively. Nodes can have a fixed location within the network (e.g. a fixed camera, or a destination laptop), or their location can be determined when the service is deployed (e.g. a transcoder that can be deployed either locally or in a remote cloud).

Based on this specification, the media streaming scenarios can be represented using the components listed in Figure 5 (note that for clarity and due to space constraints only a limited number of codecs and converters have been pictured). Note that separate Tee nodes are defined for every different flow type instead of a generalized Tee node that functions for every flow type, as the performance of Tee nodes may differ based on the flow type, and as not every flow may be splittable, either due to policy or technical reasons.

1) Specification correctness: The flexibility resulting from SBM makes it harder to reason on the deployed services and to guarantee the service correctness. It is for example possible to specify network services that are infeasible which can not be instantiated at all. This could e.g. happen when a source and destination are specified using different formats (such as 1080p and 480p), and when no converter is defined that can transform these formats.

Services may also combine in unexpected ways, and the addition of new streams, destinations and converters could have unforeseen impacts on other deployed flows that were not envisioned when the original flows were specified. When an upscaler from 480p to 1080p is defined, it is e.g. important to ensure that the upscaled stream is considered to be different from the original, non-upscaled 1080p stream. Otherwise, the

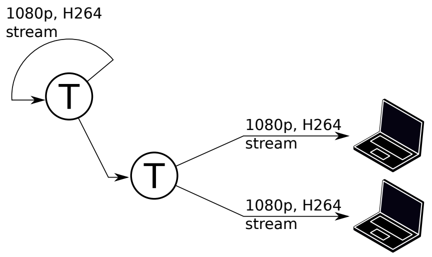


Fig. 6: When not taken into account, cycles in Tee nodes could result in self-feeding behavior.

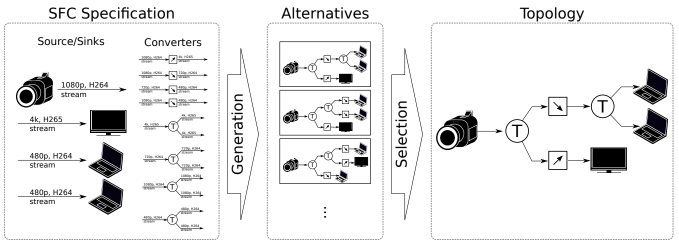


Fig. 7: Topology generation of SBM SFCs.

quality of all existing 1080p destinations may be decreased as they could suddenly be served with the lower quality upscaled stream instead of the original higher quality 1080p stream.

Finally, there is a risk of loops occurring when services are combined. The Tee nodes defined in Figure 5 may e.g. result in self-feeding behavior as illustrated in Figure 6, leading to infeasible SFCs where a destination is no longer connected to an actual source. To prevent this type of behavior, it is important to guarantee that every destination node is connected to a source node using a direct path.

In general, multiple steps can be taken to prevent unexpected consequences of changes. 1) Loop-prevention must be added to prevent infeasible SFC configurations. This can be achieved by building a management system that guarantees that every destination has a path to its source. 2) Nodes can be clustered based on the flow types they operate on, and changes of the nodes will generally only impact a single cluster, isolating the impact to only a subset of SFCs. When e.g. a 3-tuple *(content, resolution, format)* is used to represent flows, all flows containing a given *content* will be isolated from all flows containing other contents, limiting the number of SFC impacted by additions or changes to the service nodes to only flows using the same content. 3) A rigid definition of flow types and nodes without semantic connotations reduces the flexibility of the approach, but also makes it easier to reason on the resulting SFCs and to evaluate its correctness, as it is less likely for changes to have unintended consequences.

2) Deployment variability: SBM does not impose a specific service topology, and defers topology generation to the moment when SFCs are deployed or modified. This makes it possible to deploy the SBM SFC in the most efficient way depending on the network topology and the current state of the network, and allows the management system to determine the most optimal multicast tree and converter configurations at runtime. A single SBM SFC can thus result in a large collection of topologies. The final topology is decided at runtime, and does not impact the complexity of the SBM SFC model specification.

3) Specification dynamicity: When the SFC changes during its deployment, the impact on the SBM SFCs can remain limited, possibly resulting in only a small change in the deployed topology. The addition of a single new destination node would e.g. add a single destination node to the service specification, without impacting the other nodes of the specification. A management system can then determine how the change to the service specification can be applied to the deployed network service.

4) Management system requirements: Deploying SBM SFCs increases the complexity of the management system as it now has two responsibilities: (1) determining the final SFC topology and (2) deploying the topology within the network. These management tasks can be executed sequentially by first determining the service topology as illustrated in Figure 7 and subsequently allocating it as shown earlier in Figure 4. In practice, taking both into account at the same time will however lead to better results, albeit with a higher management complexity, as the optimal topology will depend on the network capacity and resource allocation decisions.

V. Conversion

While TBM and SBM lead to different management approaches, it is possible to convert a TBM SFC into an SBM SFC. Similarly, SBM SFCs can be converted into a (set of) TBM SFCs. This makes it possible to use either SFC modeling approach irrespective of the used management system. In this section, we will discuss these conversion approaches.

A. Topology-based to stream-based services

A TBM SFC can trivially be converted into a SBM SFC using a simple transformation algorithm. As explained in Section IV-A, TBM defines a graph containing service nodes N and edges E. These collections can be transformed into the N' and F' collections needed for SBM as follows:

1) For every edge $e \in E$, we define a new, unique flow type f. The set of flow types F' of the SBM SFC therefore equals E.
2) The service nodes N' needed in SBM are identical to the service node collection N defined by TBM.
3) To determine the incoming flows types of a service node $n \in N$, we determine the incoming flows and outgoing flows as defined in the edges E. Formally, this can be determined as follows:

$$n^{in} = \{(n_1, n_2) \in E | n_1 \in N \wedge n_2 = n\} \tag{1}$$

$$n^{out} = \{(n_1, n_2) \in E | n_1 = n \wedge n_2 \in N\} \tag{2}$$

B. Stream-based to topology-based services

The process of converting SBM SFCs to TBM SFCs is more complex, as an additional service matching must be executed. To achieve this, a path must be created from every destination node to a source node, adding service nodes in between if the source and destination types differ. It is possible for this process to result in multiple different TBM SFCs resulting in the same service, e.g. making use of different service permutations or different Tee node hierarchies. As the number of permutations can be very large, or even infinitely large (e.g. by infinitely chaining encode-decode blocks after each other), an additional selection step is needed which removes such flows. Ideally, this selection step should be aware of the current service state and network load, to ensure it selects the most optimal topology.

When many topologies can occur, this conversion can be complex, and therefore slow to execute. To speed up this process, the set of flow types F can be partitioned into subsets that can be processed independently, making it possible to parallelize the matching process. This can be achieved by creating a disjoint set datastructure [12] containing all of the flow types in F and subsequently joining $(f_i, f_j) \in F^2$ iff $\exists c \in C : f_i \in c^{in} \wedge f_j \in c^{out}$.

VI. Analysis

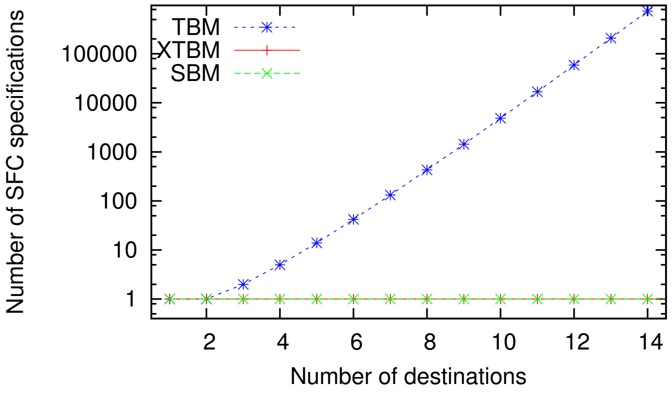
In our analysis we compare three different modeling approaches: TBM, SBM, and an XTBM approach which is aware of the Tee node concept and which abstracts an entire hierarchy of Tee nodes as a single node within the model. We analyze the modeling approaches using three media delivery scenarios and compare them based on two characteristics:

1) **Number of SFCs:** the number of SFCs that must be specified to fully express all possibilities of the scenario.
2) **Number of model artifacts.** the number of artifacts that must be specified within each the different models. For the TBM and XTBM this is the number of nodes within the topology N, for SBM, this is the number of service nodes defined within the specification.

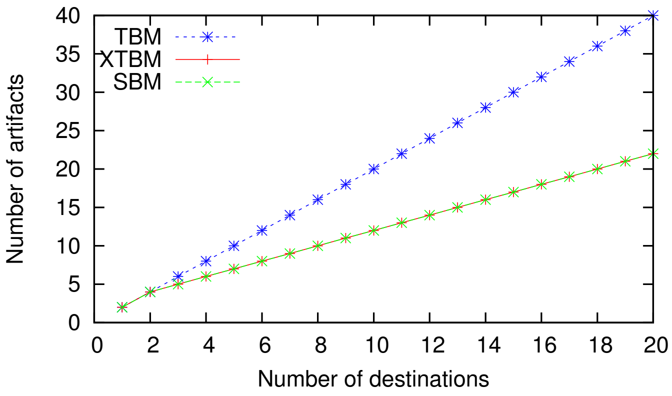
A. Basic streaming

We first consider a basic source-destination streaming scenario. In this scenario, a single source stream is transmitted to a collection of destination nodes where it must be delivered. To enable multicast, multiple Tee nodes are added to replicate the incoming streams and forward them to their destination. The streaming network in this scenario can be represented as a tree connecting a single source node to a set of destination nodes using a collection of Tee nodes. As every Tee node duplicates the incoming stream, this is a full binary tree (where every vertex either has two children or none at all).

Figure 8a shows the number of SFC specifications that are needed to fully represent this scenario. For SBM this is a single model, irrespective of the number of destinations, containing a source, the destinations, and a single Tee definition. In TBM, every permutation of the topology results in a separate SFC. The total number of full binary trees with n leaves can be represented with the Catalan number C_{n-1}, as shown in [13], meaning that the total number of SFC permutations needed for TBM is also given by C_{n-1}. XTBM, like SBM, only results in a single model, in this case consisting of a source



(a) Number of possible SFCs.



(b) Number of artifacts per SFC.

Fig. 8: Basic source to multi-destination streaming scenario.

node connected to a generalized Tee node, which is in turn connected to all destinations.

When comparing the number of model artifacts needed to represent both scenarios, as shown in Figure 8b we observe that TBM requires more model artifacts than XTBM and SBM, increasing twice as fast as the others when the number of destinations increases. This is the result of TBM containing the source, destination and a hierarchy of Tee nodes, while the other approaches only define a single, generic Tee node.

As TBM must enumerate the possible permutations, this approach will either be impractical (as too many permutations must be defined) or suboptimal (as only a small subset of potential configurations is represented). When instead XTBM is used where a hierarchy of Tee nodes is abstracted as a single node, this limitation of TBM can be mitigated, making it perform as well as SBM.

B. Scaled streaming

Next we consider a scenario where two different video sizes occur, and where the source-destination stream has to be downscaled for half of the destination nodes (rounded down). In this scenario, a downscaler must be added somewhere within the stream before it reaches its destination. In the extreme cases this results in a single downscaler node followed by a Tee node hierarchy, or a topology where downscalers are placed in front of half of the destinations. The total number of TBM SFCs with n destinations can in this scenario be determined as follows:

1) Define $p = \lceil n/2 \rceil$ the number of non-scaled destinations and $q = \lfloor n/2 \rfloor$ the number of downscaled destinations.
2) The complete tree structure can now be split into multiple subtrees: a multicast tree starting from the source node going to all p non-scaled destinations and going to one or more downscalers, and a multicast tree rooted in every downscaler going to the q downscaled destinations (in such a way that every downscaled destination is a leaf in exactly one of these subtrees).
3) When k subtrees are created, each having q_i ($i \in 1 \ldots k$) elements, with $q = \sum_{i \in 1 \ldots k} q_i$, the number of permutations can be determined as $C_{p+k-1} \times \prod_{i \in 1 \ldots k} C_{q_i - 1}$.
4) The total number of possible SFCs can now be determined by summing the above formula for all possible k and q_i values. These possible configurations can be easily determined exhaustively, ensuring $q_i \geq q_{i+1}$.

Using a similar approach, the number of XTBM SFCs can be determined. In this case the number of tree permutations is however 1, as a single super-Tee node is used instead of a Tee node hierarchy with C_{i-1} permutations.

Figure 9a shows the number of SFCs needed to model this scenario as the number of destinations increases. Here, we observe that XTBM scales much better than TBM, making it feasible for practical use. XTBM however still scales much worse than SBM, which only requires a single SFC model.

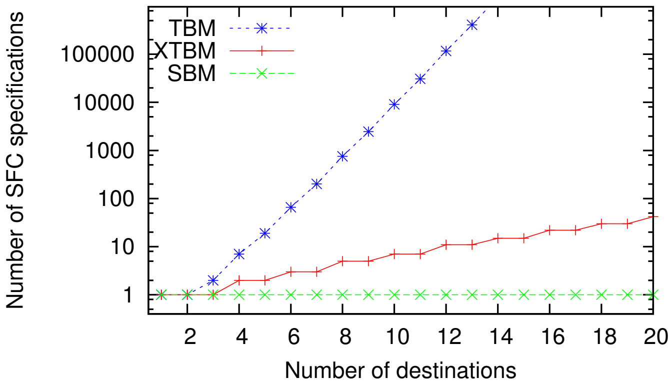
Comparing the number of artifacts in every model, as shown in Figure 9b, we observe that the number of TBM artifacts can vary depending on the chosen configuration, as the number of downscalers can vary between 1 and $\lfloor n/2 \rfloor$. The number of XTBM artifacts can be computed similarly, and requires fewer artifacts than TBM as only a single super-Tee node is used instead of an entire hierarchy. Finally, the number of SBM artifacts equals the minimal number of artifacts of XTBM.

C. *Processed streaming*

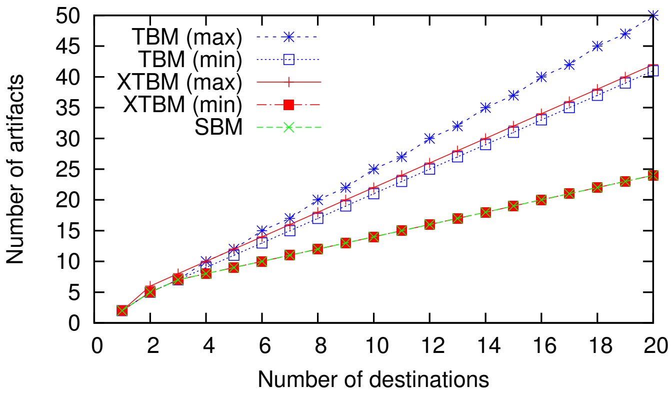
Finally, we consider a scenario where a single source-destination stream is transformed in multiple steps. These steps can be transformations applied to the content of the media stream (e.g. upscaling, downscaling, transcoding, adding watermarks etc.) or to the entire stream (e.g. NAT, DPI, encryption, etc.). While in practice, an ordering will be imposed on some transformations, we consider a worst-case scenario where n transformations must be executed in any order, resulting in the maximum number of different configurations.

In this scenario, the total number of TBM configurations is given by the total number of transformation node permutations $n!$. Each TBM SFC specified in this way consists of $n + 2$ artifacts, one for every transformation, one for the source, and one for the destination node. As there are no Tee nodes in this scenario, XTBM behaves identically to TBM.

Using SBM, this scenario can be modeled using a single SFC, but a large number of service nodes must be created



(a) Number of possible SFCs.



(b) Number of artifacts per SFC.

Fig. 9: Source to multi-destination streaming scenario with downscalers.

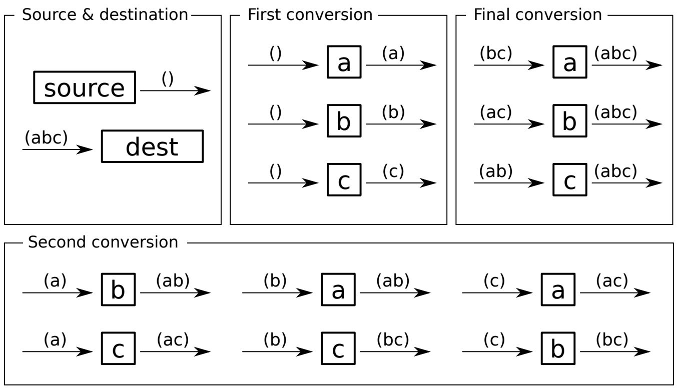
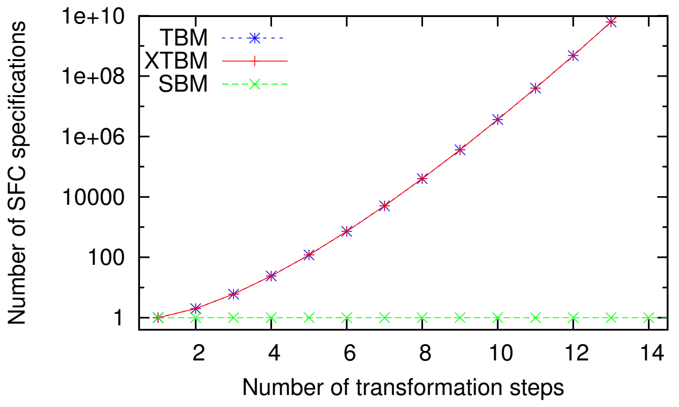
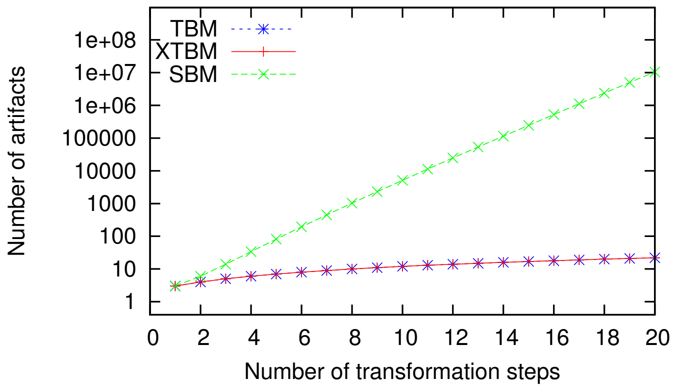


Fig. 10: SBM SFC for a processed streaming scenario with three transformation nodes.



(a) Number of possible SFCs.



(b) Number of artifacts per SFC.

Fig. 11: Source to single destination streaming scenario with multiple transformation nodes.

for all converters, as a large number of different intermediate, partially processed flow types will exist. This is illustrated for three transformation nodes in Figure 10, where in total 14 different converter nodes must be defined to support all permutations. In general for n transformations, the number of nodes needed in SBM is determined as follows:

$$2 + \sum_{i=0\ldots n} \binom{n}{i} * (n - i) \tag{3}$$

This equation can be understood as follows: for the i^{th} conversion step, we determine the number of possible transformations that have already been applied (given by the number of ways i elements can be chosen from n). To these incoming flows, we can apply each of the remaining $n - i$ transformations.

Figure 11 shows the number of permutations needed for the modeling approaches, and the number of artifacts needed for SFCs defined using both. We observe that the number of SFC permutations for TBM and XTBM, and the number of artifacts needed in SBM increases exponentially over time. While TBM and XTBM at first require fewer SFC specifications than SBM requires artifacts, the former grows much faster than the latter when the number of transformations increases. This shows that none of the approaches scale well in terms of the number of transformations in this scenario, but that SBM still works best for large numbers of transformation nodes.

	TBM	XTBM	SBM
Specification complexity	Low	Low	High
Correctness evaluation complexity	Low	Average	High
Flexibility	Low	Average	High
Management system complexity	Low	High	Average
Management algorithm complexity	Low	High	High
Standard compliance	High	High	Low
Deployment optimality	No	Limited	Yes
Suitability for media streaming	Low	Average	High

TABLE I: Comparison of the three SFC modeling approaches.

VII. Discussion

Table I shows a high-level overview comparing the three SFC modeling approaches. TBM results in the lowest complexity, both from a specification and management point of view, and has the added benefit of being similar to the VNFFGD management specification standard. Our analysis however shows that TBM is not well-suited for media streaming as it is unable to represent the large number of Tee nodes in streaming topologies.

XTBM addresses the key weakness of TBM, making it feasible for defining media streaming SFCs, but requires a specialized management system which is aware of the generalized super-Tees it defines. Furthermore, our analysis shows that the possible number of SFCs still increases significantly when transformations are added, either as part of the streaming tree or in sequence, limiting the optimality or feasibility of XTBM SFCs. It may be possible to address these limitations by further making the management system aware of other scenario specifics such as upscaling, downscaling, encryption and other transformations, but this would require a highly specialized and inflexible management system.

Finally, SBM is the most promising, as it is able to represent all possible SFC deployments using a single SFC specification in each scenario. As all service permutations are represented using this single model, this makes it possible to determine an optimal service configuration at runtime at the cost of an increased (runtime) management complexity. Like TBM, SBM can be used to generically manage SFCs without having to be aware of multimedia specific functionalities,

As TBM and SBM SFCs can be converted into one another, it is possible to combine TBM SFCs and SBM SFCs within a single system. TBM SFCs can be trivially executed in a SBM management system, while an additional conversion system is needed to convert an SBM SFC into the most optimal TBM SFC depending on the network topology and service utilization at the time of the conversion.

VIII. Conclusions

In this paper, we presented and discussed three alternative modeling approaches for SFCs: TBM which functions using service topologies, XTBM which extends TBM and adds awareness of scenario-specific elements, and SBM which defines multiple service components. We compared these three approaches and analyzed their performance in three multimedia streaming scenarios. We found that, while TBM results in SFCs that are easy to define and manage, it is unable to represent the large deployment variability that can occur when media streams must be delivered to multiple end users or when multiple transformations are applied to the flow. XTBM addresses this by adding awareness of some scenario-specific concepts to the management system, but is still unable to fully represent all possible configurations using a single SFC. SBM is able to represent all possible configurations using a single SFC, but is harder to use and results in a higher management complexity. As it is possible to convert TBM SFCs into SBM SFCs, we believe it would be beneficial to focus on the development of a SBM SFC management system in the future, as it is able to handle a much higher SFC flexibility.

Acknowledgement

This work is partially funded by the iMinds EMD project and FWO project 11385 "Service-centric management of a virtualized future Internet".

References

[1] M. Rabbani, R. Pereira Esteves, M. Podlesny, G. Simon, L. Zambenedetti Granville, and R. Boutaba, "On tackling virtual data center embedding problem," in *IFIP/IEEE International Symposium on Integrated Network Management (IM 2013)*. IEEE, May 2013, pp. 177–184.

[2] H. Moens, B. Hanssens, B. Dhoedt, and F. De Turck, "Hierarchical Network-Aware Placement of Service Oriented Applications in Clouds," in *Proceedings of the 14th Network Operations and Management Symposium (NOMS 2014)*. IEEE, may 2014, pp. 1–8.

[3] OMG. (2011) Business Process Model and Notation (BPMN) Version 2.0. [Online]. Available: http://www.omg.org/spec/BPMN/2.0/

[4] NFV ETSI ISG. (2014) GS NFV-MAN 001. Network Functions Virtualisation Management and Orchestration V1.1.1. Last accessed: February 2015. [Online]. Available: http://www.etsi.org/index.php/technologies-clusters/technologies/nfv

[5] S. Mehraghdam, M. Keller, and H. Karl, "Specifying and placing chains of virtual network functions," in *3rd International Conference on Cloud Networking (CloudNet)*. IEEE, Oct. 2014, pp. 7–13.

[6] M. Bouet, J. Leguay, and V. Conan, "Cost-based placement of vDPI functions in NFV infrastructures," in *Proceedings of the 1st Conference on Network Softwarization (NetSoft)*. IEEE, Apr. 2015, pp. 1–9.

[7] H. Moens and F. De Turck, "VNF-P : A Model for Efficient Placement of Virtualized Network Functions," in *Proceedings of the 10th International Conference on Network and Service Management (CNSM 2014)*. IEEE, nov 2014, pp. 418–423.

[8] B. Jennings and R. Stadler, "Resource management in clouds: Survey and research challenges," *Journal of Network and Systems Management*, vol. 23, no. 3, pp. 567–619, 2015.

[9] R. Guerzoni, R. Trivisonno, I. Vaishnavi, Z. Despotovic, A. Hecker, S. Beker, and D. Soldani, "A Novel Approach to Virtual Networks Embedding for SDN Management and Orchestration," in *Proceedings of the 14th Network Operations and Management Symposium (NOMS 2014)*. IEEE, may 2014, pp. 1–7.

[10] S. Pandey, L. Wu, S. M. Guru, and R. Buyya, "A Particle Swarm Optimization-Based Heuristic for Scheduling Workflow Applications in Cloud Computing Environments," in *24th IEEE International Conference on Advanced Information Networking and Applications*. IEEE, 2010, pp. 400–407.

[11] O. Biran, A. Corradi, M. Fanelli, L. Foschini, A. Nus, D. Raz, and E. Silvera, "A stable network-aware vm placement for cloud systems," in *12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*. IEEE, May 2012, pp. 498–506.

[12] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, "Data structures for disjoint sets," in *Introduction to Algorithms (Second ed.)*. MIT Press, pp. 498–524.

[13] R. P. Stanly, *Enumerative combinatorics, Volume 2*. Cambridge University Press, 1999.