A Domain-Specific Language for Multimedia Service Function Chains based on Virtualization of Sensors

Franklin Jordan Ventura Quico
fventuraq@midiacom.uff.br
Fluminense Federal University
Niterói, RJ, Brazil

Anselmo L. E. Battisti
anselmo@midiacom.uff.br
Fluminense Federal University
Niterói, RJ, Brazil

Débora Muchaluat-Saade
debora@midiacom.uff.br
Fluminense Federal University
Niterói, RJ, Brazil

Flavia C. Delicato
fdelicato@ic.uff.br
Fluminense Federal University
Niterói, RJ, Brazil

ABSTRACT

Virtualization is a widely used technology that can abstract the complexity of heterogeneous environments, such as the Internet of Things (IoT) and multimedia systems. Multimedia sensors are an important data source in the Internet of Things (IoT), which brings the Internet of Media Things (IoMT) paradigm. Based on virtualization and IoMT, the concept of a multimedia Virtual Network Function (multimedia VNF) has been adopted to denote the virtualized representation of devices and also software components that process multimedia streams. In many scenarios, multiple processes must be applied to multimedia streams in a predefined sequence, thus creating the concept of multimedia Service Function Chain (multimedia SFC). Few efforts have been made in the literature to create a description language to support the definition of multimedia SFCs. In order to fill this gap, we propose a Domain Specific Language (DSL) called L-PRISM. This DSL can be used as a conceptual base for developers to implement and virtualize multimedia applications using multimedia VNFs. We also present a Proof of Concept (PoC) that uses L-PRISM to run multimedia SFCs. Our DSL and PoC were evaluated by software developers, and the results show that adopting L-PRISM facilitates the definition and deployment of multimedia SFCs based on multimedia VNFs.

KEYWORDS

IoMT, IoT, VNF, SFC, DSL, L-PRISM

1 INTRODUCTION

Virtualization is a concept that has paved the way for numerous research studies in Cloud Computing, 5G, and IoT [14, 24]. This technology not only reduces the costs of implementing and managing infrastructures, but also serves as a fundamental pillar in paradigms such as Network Function Virtualization (NFV) [16, 19]. The adoption of NFV and its Virtual Network Function (VNF) components has proved to be effective in reducing the consumption of computational and network resources. Furthermore, virtualization increases the speed and efficiency of resource provisioning in cloud and edge environments, facilitating the management and orchestration of these resources [24].

Many investigations combine NFV technology with multimedia stream processing. In [4], the authors explore the concept of virtual devices that abstract the heterogeneity of multimedia sensors and introduce virtual multimedia sensors (VMS), which we will refer to as multimedia VNFs in our work. These VMSs enable multimedia stream processing using lightweight virtualization. Furthermore, the authors demonstrated the feasibility of using chained VMSs to create complex multimedia stream processing pipelines. However, this characteristic was not fully explored in that work. Meanwhile, in [11], the authors investigated the joint load balancing and autoscaling of multimedia VNFs in edge or cloud environments. In this context, virtualization concepts like NFV have been actively integrated into different studies to process multimedia streams.

The Service Function Chain (SFC) is a sequence of sorted VNFs associated with a Service Level Agreement (SLA) [5]. By chaining multimedia functions, it is possible to create complex multimedia stream processing pipelines. When the VNFs in an SFC are tailored for multimedia processing, they are referred to as multimedia SFCs.

Despite the potential of multimedia SFCs in various fields [9], we face the challenge of implementing and managing them easily, securely, and efficiently. The components used to create multimedia SFCs are typically developed by different entities using multiple technologies [14]. Thus, to create a multimedia SFC, users must have a moderate level of knowledge of these technologies. Moreover, the interoperability of these components sometimes is often limited.

To address these challenges, it is necessary to provide a layer that encapsulates the technical details of multimedia SFCs. There are different alternatives for this, among which Domain-Specific Languages (DSLs) stand out [18]. Unlike other approaches, such as metamodels or conventional programming languages, DSLs offer greater flexibility and simplicity.

DSLs are language specifications designed to describe the characteristics, syntax, and behavior of systems in specific fields [15]. These languages enable intuitive integration and management by providing an abstraction layer encapsulating technical details. This allows users to focus on business logic and specific functionalities. In addition, DSLs facilitate the integration of new or external components more efficiently and less technically.

Although there have been significant advances in the literature to facilitate the definition and deployment of NFV services, such as

In: Proceedings of the Brazilian Symposium on Multimedia and the Web (WebMedia'2024). Juiz de Fora, Brazil. Porto Alegre: Brazilian Computer Society, 2024.

ISSN 2966-2753

the TOSCA-NFV metamodel [22] or ETSI GS NFV-SOL [10], there is still a considerable gap specifically regarding multimedia SFCs. The need to address multimedia SFCs separately arises from their unique nature and challenges. Some of these challenges are (i) managing large-volume and high-speed data flows, (ii) synchronizing multiple formats and media types, and (iii) meeting requirements for real-time processing, which are not as critical in traditional NFVs [17]. Furthermore, multimedia SFCs often require greater flexibility in terms of scalability and adaptation of dynamic changes in resource demand. Therefore, although existing models, such as TOSCA-NFV, provide means to describe SFCs in general, the unique characteristics of multimedia SFCs demand a specialized approach that directly addresses their specific requirements and inherent challenges.

In this work, our aim is to fill this gap by proposing a DSL called Language for Programming IoT Sensors for Multimedia (L-PRISM), which serves as a conceptual foundation for describing multimedia SFCs, particularly in the emerging field of the IoMT. The decision to develop L-PRISM, instead of extending existing solutions like TOSCA-NFV, is driven by the unique demands of multimedia SFCs, such as the inherent complexity in their definition and configuration, the need for flexibility in their components and adaptability to specific environments.

Our DSL includes a data model that provides detailed structures for specifying the critical properties of multimedia SFCs. This model enables the specification of the type, formats, and resolutions of data streams that multimedia VNFs can process and send, but it also considers scenarios where a multimedia VNF can handle its streams in various formats and qualities. This aspect is crucial, as the computational and network requirements associated with a multimedia VNF are variable and depend on the characteristics of the multimedia stream to be processed.

This information is vital for developers of multimedia SFCs, as it provides the necessary details to understand and utilize multimedia VNFs. In turn, this will enable developers to specify their multimedia SFCs efficiently. Additionally, at the application level, our data model can facilitate the effective orchestration and management of multimedia SFCs, optimizing their lifecycle.

Furthermore, having a dedicated DSL presents several advantages, such as customization, facilitated implementation, and maintenance of its various components. The main contributions of this work are:

- L-PRISM makes it possible to describe a multimedia SFC based on a sequence of multimedia VNFs.
- L-PRISM defines the necessary structures for detailed logging of multimedia VNFs.
- L-PRISM makes using multimedia VNFs developed by third parties easier, so developers of multimedia VNF-based solutions do not need to have advanced knowledge about the technologies or tools used for developing the components of a multimedia SFC.

The remainder of this paper is organized as follows. Section 2 describes the background and related work. Sections 3 and 4 present L-PRISM and its metamodel. Section 5 describes our PoC. In Section 6, we present an evaluation of our work. Section 7 brings our main conclusions and future work.

2 BACKGROUND AND RELATED WORK

Traditional sensors, such as thermostats and presence detectors, typically produce discrete data in a readable format. However, multimedia sensors produce continuous, complex and large-volume data, which require high processing rates, massive storage, wide bandwidth, low latency, and high energy consumption [17]. These characteristics make their operation different from that of traditional sensors. One strategy to handle this complexity is to decouple physical sensors from multimedia stream consumers [4].

Different languages, metamodels, and DSLs have been developed, which have helped develop systems and architectures, such as, YANG [20], TOSCA-NFV [22] and ETSI GS NFV-SOL [10].

YANG [20] is a data modeling language used to describe network configurations and telecommunication services. YANG can be used for NETCONF-based operations, including configuration, data state, remote procedure calls (RPCs), and notifications. YANG models can be translated into XML syntax, allowing applications that use XML parsers to operate and manipulate YANG models [6]. However, YANG does not have features required to describe multimedia SFCs.

A specific data model that works with virtualization technologies is TOSCA-NFV [22], which allows describing, deploying, and managing applications and services based on NFV. TOSCA-NFV describes in detail the components of its solutions and the interactions and dependencies between them. TOSCA is a modeling language that uses the template concept to describe the components of a workload using a topology template, and their relationships using a relationship template [22]. It is crucial to note that TOSCA-NFV focuses primarily on traditional virtualization systems and does not align well with lightweight virtualization platforms, such as Docker and Kubernetes.

Mininet-NFV [8] presents an advanced framework for NFV orchestration, facilitating the implementation and operation of VNF-based network services. This framework is built upon Mininet, a tool widely recognized for its capability in agile experimentation with networks, SDN and NFV, and extensively used for prototyping, testing, and implementing NFV solutions. Additionally, Mininet-NFV supports parameterized TOSCA-NFV templates, using virtual link descriptors to define detailed and flexible network configurations.

ETSI GS NFV-SOL 001 [10] adapts TOSCA-NFV to meet the specification requirements of ETSI GS NFV IFA 011 [1] and 014 [2] for Virtualized Network Function Descriptors (VNFDs), Network Service Descriptors (NSDs), and Physical Network Function Descriptors (PNFDs). Like TOSCA-NFV, it specifies requirements for the management and orchestration of VNFs.

Although very related to our proposal, neither TOSCA-NFV nor ETSI NFV-SOL specifically addresses the requirements of multimedia applications and systems, particularly concerning multimedia SFCs and their components. Therefore, while our work is based on the principles of TOSCA-NFV, it stands out by addressing the necessary aspects for the definition, orchestration, and management of multimedia SFCs implemented on lightweight virtualization platforms.

Table 1 compares related languages, templates, and data models that support the definition of SFCs. The last row of the table presents L-PRISM. The meaning of each column in the table are as follows:

- **IoMT:** Focus on Internet of Media Things applications.
- **Edge Computing:** Tailored for edge computing environments.
- **Light Virtualization:** Support for lightweight virtualization platforms.

Table 1: Comparison between related work.

Related Work	IoMT	Edge Computing	Light Virtualization
YANG [20]	-	-	-
TOSCA-NFV [22]	-	✓	-
Mininet-NFV [8]	-	✓	✓
ETSI NFV-SOL [10]	-	✓	✓
L-PRISM	✓	✓	✓

Our previous work [3, 4] proposed an architecture, which allows the execution and management of multimedia VNFs based on lightweight virtualization in an edge-cloud environment. However, at that time, we did not explore multimedia SFCs or provided a domain-specific language to describe multimedia SFCs, which would make the creation and maintenance of multimedia SFCs easier. In this work, we fill in this gap by simplifying the definition of multimedia SFCs through the development of a DSL.

3 L-PRISM

This section presents L-PRISM, a DSL that facilitates the description of multimedia SFCs based on multimedia VNFs. L-PRISM is mainly based on the architecture proposed in [3] and the TOSCA-NFV metamodel [22]. L-PRISM was developed following the stages proposed by Negm et al. [18], detailed in the following sections.

3.1 Domain Analysis

An analysis of multimedia applications in virtualized environments was carried out. It allowed us to understand and abstract the main characteristics that influence the operation of these types of application. The characteristics represent configurable aspects such as the allocation of computing and network resources, communication requirements, etc.

In addition, an analysis of the interaction between multimedia applications was performed. These interactions can be modeled as directed graphs, representing complex multimedia applications. We describe these complex applications as *multimedia SFCs* and their components as *multimedia VNFs*.

3.2 L-PRISM Design

L-PRISM was created following the concepts presented in [4] as a reference. Thus, the design of L-PRISM considers the implementation of *multimedia SFCs* in distributed environments composed of edge and cloud nodes. In addition, L-PRISM was coined to operate on lightweight virtualization platforms, such as Docker and Kubernetes. Figure 1 shows an example of a multimedia SFC. The main components of a multimedia SFC are:

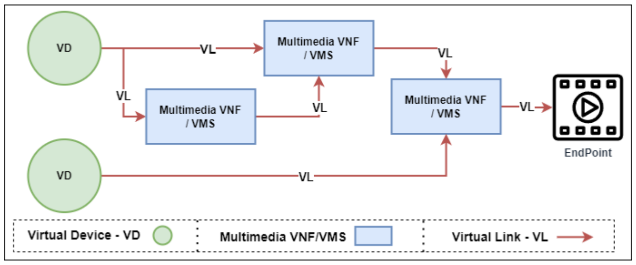


Figure 1: Example of a multimedia SFC.

- *Virtual Device (VD)* is the virtual representation of a physical multimedia device.
- *Multimedia VNF / VMS* is a function that process the multimedia streams. Unlike VDs, instances of multimedia VNFs are created exclusively for each multimedia SFC.
- *Virtual Links (VL)* provide the communication between (i) VDs and multimedia VNFs, (ii) different multimedia VNFs and (iii) multimedia VNFs and endpoints (user applications).

L-PRISM is a flexible language that can be used in compiled and interpreted scenarios. In both cases, the user describes the multimedia SFC using our proposed language. In a compilation-based scenario, the compiler will create a low-level code that will be processed to execute the multimedia SFCs. In an interpreted-based scenario, which is the case of our PoC, an interpreter runs over the code and creates/instantiates the multimedia SFCs.

The L-PRISM language is based on YAML to provide readability, flexibility, cross-platform support, integration with existing tools, and a fast learning curve [23]. These features make L-PRISM intuitive and easy to use, and in turn, YAML is tailored to our specific needs and helps to make our DSL easily accepted by developers.

L-PRISM follows a model-driven approach. Figure 2 presents the L-PRISM metamodel, defining its components, attributes, and the relationships between them. For example, the *swImage* (Image of the multimedia VNF) structure in the metamodel of our DSL allows for detailed description of the type, formats, and resolutions of multimedia streams that a multimedia VNF can process and emit. This information at the application level can be stored in a database, which can then be accessed by developers of multimedia SFCs. This will enable them to understand the characteristics of multimedia VNFs generally and efficiently specify each component of their multimedia SFCs.

For an extensive overview of L-PRISM, visit our website[1].

4 THE L-PRISM METAMODEL

This section summarizes the L-PRISM metamodel. Table 2 presents the attributes of a *chainModel*, which describes a multimedia SFC. It should be noted that attributes ending with an (*) are mandatory.

Table 3 describes the attributes of the *device* component, which is a virtual representation of a physical device (camera or microphone). One advantage of virtualized devices is that a single multimedia stream captured from a camera/microphone can be reused multiple times and sent to different destinations.

[1] https://fventuraq.github.io/lprism.html

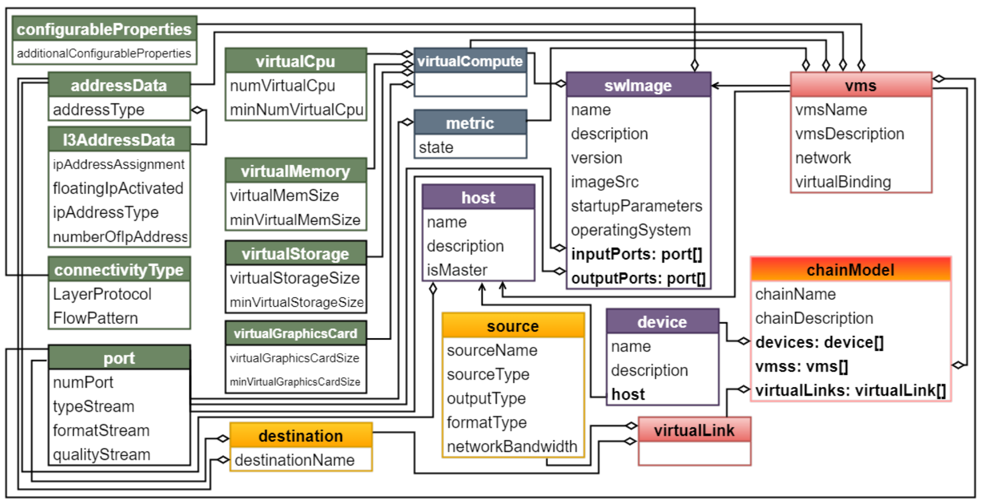


Figure 2: Class diagram of the L-PRISM metamodel.

Table 2: Attributes of a *chainModel* in L-PRISM

Name	Type	Description
chain-Name*	String	Name of the multimedia SFC.
chain-Description	String	Multimedia SFC description.
devices*	map	List of virtual devices (*device*) that compose the multimedia SFC.
vmss*	map	List of multimedia VNFs (*vms*) that compose the multimedia SFC.
virtual-Links*	map	List of *virtualLinks* between *device* and multimedia VNF (*vms*).

Table 3: Attributes of *device* in L-PRISM

Name	Type	Description
device-Name*	String	Unique name of a virtual device that works as an *ID* in a *virtualLink*.
deviceId*	String	Identifies a virtualized *device*. This index helps any element of the multimedia SFC to subscribe to a *device*. When a *device* receives a subscription, it replicates its stream and sends it to the new subscriber. It should be noted that virtualized devices are not created in the multimedia SFC. They are initialized separately and can be part of different multimedia SFCs.

Table 4 presents the attributes of a *vms* (multimedia VNF) component. A multimedia VNF within a multimedia SFC is used to process one or multiple multimedia streams and generate one or more output results. The result of a multimedia VNF can be sent to one or more destinations, including an end-user application.

Table 4: Attributes of multimedia VNF (*vms*) in L-PRISM

Name	Type	Description
vmsName*	String	The name of a multimedia VNF must be unique in a multimedia SFC. This data type works as an *ID* in a *virtualLink*.
vms-Description	String	Multimedia VNF description.
vmsType*	lPrism.-artifacts.-vms.-swImage	The container images used for creating the multimedia VNF.
startup-Parameters	lPrism.-datatype.-vms.-configurable-Properties	Data type defined initially by TOSCA-NFV; it describes the initial configuration properties of the multimedia VNF.
host*	lPrism.-nodes.-vms.host	network node that will host the multimedia VNF. The list of available nodes must be shared with the developer of the multimedia SFC.
virtual-Compute	lPrism.-capabilities.-vms.-virtual-Compute	Computational properties assigned to the multimedia VNF.

Tables 5 and 6 present the *source* and *destination* element attributes, respectively. The interconnections between two elements (virtual device or VNF) of a multimedia SFC are described by *virtualLink*. This element provides information on two other components, the source point (*source*) from which the stream is sent and the destination point (*destination*) where the stream will be received.

Table 5: Attributes of *virtualLink-source* in L-PRISM.

Name	Type	Description
source-Name*	String	source of a multimedia stream. This name must be identical to that configured in a *deviceName* or a *vmsName*.
source-Type*	String	type of the source node of the virtual link, which can be *vms* or *device*.
output-Type*	String	Describes the type of multimedia stream to be sent. Multimedia applications usually process one type of stream but may produce a different output type.
format-Type	String	The multimedia stream type such as MP4, AVI, MPEG.
network-Bandwidth	lPrism.data-type.vms.-network-Bandwidth	Bandwidth assigned to the connection. Within a host, it is not necessary to specify this attribute, as connections are efficiently managed internally. However, for connections between different hosts on the network, it is vital to properly establish and adjust the bandwidth according to the characteristics of the multimedia stream.

Table 6: Attributes of *virtualLink-destination* in L-PRISM.

Name	Type	Description
destina-tionName*	String	Which component of the multimedia SFC will receive the stream. This data must be identical to the one configured in a *vmsName* of one of the *vms* used in the same multimedia SFC.
Address	lPrism.-datatype.-vms.-addressData	Complex data type initially defined by TOSCA-NFV and adapted in L-PRISM, which describes a network address.
Port*	lPrism.-datatype.-vms.-listPorts	Describes the port where the multimedia stream will be received.
inputType*	String	Describes the type of multimedia stream that the destination of the connection accepts.

Source Code 1 gives a simple example of a multimedia SFC described using L-PRISM. Figure 3 shows the visual representation of that SFC. It is composed by one *device*, one *vms* and two *virtualLinks*. It has one video source (VD 2) that is handled by a VNF (Multimedia VNF 1) that sends the video stream to the final application (192.168.0.101 Port 1002). The next section presents our PoC.

Source Code 1: A multimedia SFC described using L-PRISM.

```
chainModel:
    chainName: multimedia SFC test
    chainDescription: Video Flux
    devices: #List of devices, in this example 1 device
    -
        deviceName: VD 2
        deviceId: 638e70b10a1fbd0026fccaf4
    vmss: #List of multimedia VNFs, in this example 1 VNF
    -
        vmsName: Multimedia VNF 1
        vmsDescription: Multimedia VNF 1 description
        vmsType: 638e70b10a1fbd0026fccae8
        host: 192.168.0.117 # IP
        virtualCompute:
          virtualMemory:
              size: 1024
          virtualCPU:
              numCpu: 1
    virtuallinks:
    - #virtualLink 1 ("VD 2" -> "Multimedia VNF 1")
        source:
          sourceName: VD 2
          sourceType: device
          outputType: video
        destination:
          destinationName: Multimedia VNF 1
          address: #define for orchestrator
          port: 5000 # port of multimedia VNF
          inputType: video
    - #virtualLink 2 (final connection)
        source:
          sourceName: Multimedia VNF 1
          sourceType: vms
          outputType: video
        destination:
          destinationName: 192.168.0.101
          address: 192.168.0.101
          port: 10002
          inputType: video
```

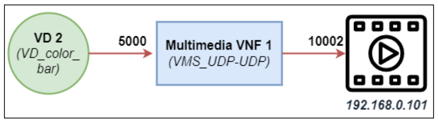


Figure 3: Multimedia Chain example.

5 ALFA 2.0: PROOF OF CONCEPT (POC)

This section describes the integration of L-PRISM into a platform named ALFA [3][2]. ALFA uses containers to run VMSs and virtual devices thus allowing language independence. Our PoC named ALFA 2.0 is released under the MIT License. The source code is available[3].

5.1 Database

ALFA uses a non-relational database (mongoDB) to store information about VMSs, Virtual Devices, Edge Nodes, VMS types, and Device types. To extend its functionality, we added the L-PRISM components, such as *chain model*, to the database. The chain model

[2]Source code available at https://github.com/midiacom/alfa
[3]Source code of alfa 2.0 https://github.com/fventuraq/alfa

stores information about each created multimedia SFC, such as which VMSs are used within a multimedia SFC, which devices are used, and the connections between them.

In addition to these changes, some attributes were modified and added to the already defined components, such as VMS, VD, Edge Node, VMS types, and Device type. These new attributes helps ALFA 2.0 to support the L-PRISM specification. With these changes, it is possible to create, and manage multimedia SFCs defined using L-PRISM inside ALFA 2.0.

5.2 API

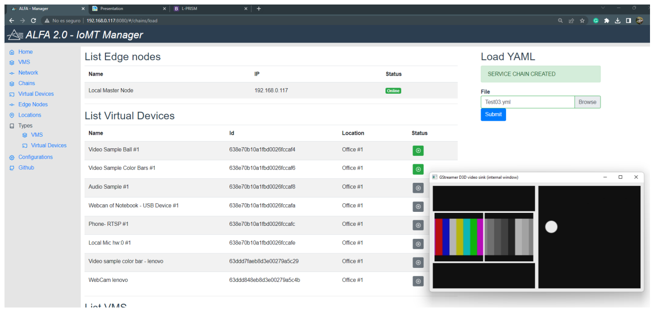
The ALFA implementation already has an API that is used to manage the entities responsible for executing the VMSs. This API did not support multimedia SFC, thus a new endpoint was added to the ALFA 2.0 API. This new endpoint receives and processes a YAML file that contains the multimedia SFC described using L-PRISM.

To manage multimedia SFCs, multiple tasks must be performed. The new endpoint executes commands using the Docker API and the ALFA API to create VMSs. After that, the virtual devices are linked to the VMSs, and interconnections between VMSs are created.

5.3 Web Client

ALFA API allows different software to interact with the entity of the platform. The ALFA implementation already has a Web Client to manage the VMSs. Thus, we extended the original web-based client by adding a new component that allows the upload of a file with the multimedia SFC description using the L-PRISM language.

Figure 4 shows the web client interface used to send the file with the multimedia SFC description. The Web client presents the VMSs, Virtual Devices, and nodes running in the environment. The figure also shows the result of the multimedia stream processed by the created multimedia SFC.



List Edge nodes

Name	IP	Status
Local Master Node	192.168.0.117	Online

SERVICE CHAIN CREATED

File: Test03.yml

List Virtual Devices

Name	Id	Location	Status
Video Sample Ball #1	638e70b10a1fbd0026fccaf4	Office #1	
Video Sample Color Bars #1	638e70b10a1fbd0026fccaf6	Office #1	
Audio Sample #1	638e70b10a1fbd0026fccaf8	Office #1	
Webcan of Notebook - USB Device #1	638e70b10a1fbd0026fccafa	Office #1	
Phone- RTSP #1	638e70b10a1fbd0026fccafc	Office #1	
Local Mic hw:0 #1	638e70b10a1fbd0026fccafe	Office #1	
Video sample color bar - lenovo	63ddd7faeb8d3e00279a5c29	Office #1	
WebCam lenovo	63ddd848eb8d3e00279a5c4b	Office #1	

Figure 4: Web interface for executing L-PRISM files in ALFA 2.0.

6 EVALUATION

This section describes the L-PRISM evaluation. The experiment was carried out with computer science students. We asked them to create multimedia SFCs with L-PRISM and also using a manual deployment.

6.1 Goal Question Metric

The experiment aims to evaluate L-PRISM during the development of multimedia SFCs. The Goal Question Metric (GQM) methodology was adopted in the process [12]. Table 7 presents our goals.

Table 7: Evaluation goals.

Goal	Description	Perspective
G1	Analyze the application engineering process with and without L-PRISM to evaluate the efficiency and productivity of developing multimedia SFC.	Efficiency and Productivity
G2	Evaluate the comprehensibility of L-PRISM to analyze if variables, attributes, and structures are understandable for the participants.	Usability

For $G1$, six questions related to efficiency and productivity in the process of developing multimedia SFCs were raised. Table 8 presents the G1 questions and the metrics related to them.

Table 8: Questions and metrics for goal *G1*.

#Q	Description	Metrics
Q1	Is the application engineering process using L-PRISM effective in terms of time for developing multimedia SFC, compared to the traditional approach (ALFA)?	M1 - Development effort
Q2	Does the developer claim that using L-PRISM makes it easier to understand the functional and non-functional requirements of the multimedia SFC?	M2 - Understanding of requirements
Q3	Does the developer claim that using L-PRISM helps create multimedia SFC?	M3 - Perceived ease of use
Q4	Does the developer claim that L-PRISM is useful to create multimedia SFC?	M4 - Perceived utility
Q5	Does the developer claim that using L-PRISM makes it easier to reuse the multimedia SFC created with L-PRISM to create new multimedia SFC?	M5 - Perceived reuse
Q6	Is the process of modifying multimedia SFC faster with L-PRISM compared with the traditional method (ALFA)?	M6 - Reuse effort

The questions for $G1$ were adapted from the Technology Acceptance Model (TAM) [21]. The goal was to quantitatively evaluate the multimedia SFC development process using L-PRISM and the traditional method. Metrics $M2$, $M3$, $M4$ and $M5$ were measured using the Likert scale (1 - strongly disagree to 5 - strongly agree) [13], and for metrics $M1$ and $M6$, the time in minutes required to perform each task was requested.

For goal $G2$, seven questions ($Q1 - Q7$) related to Cognitive Dimensions of Notations (CDN) [7] were used. This approach helped us evaluate L-PRISM from a usability point of view. Table 9 presents the proposed questions for $G2$ and the respective metrics. Those questions should be answered using the Likert scale, for $Q3$ and $Q4$ (1 - strongly disagree to 5 - strongly agree) and for *Q1, Q2, Q5, Q6* and $Q7$ (1 - very difficult to 5 - very easy) [13].

Table 9: Questions and metrics for goal *G2*.

#Q	Description	Metrics
Q1	How easy is it to visualize or find the different elements and attributes of L-PRISM when creating or changing a multimedia SFC?	M1 - Visibility
Q2	How easy is modifying a multimedia SFC with L-PRISM?	M2 - Viscosity
Q3	Is the L-PRISM language too verbose to specify a multimedia SFC?	M3 - Diffuseness
Q4	How well do the L-PRISM elements and attributes represent a multimedia SFC?	M4 - Closeness of Mapping
Q5	How easy is it to understand the structures and data types of L-PRISM?	M5 - Role Expressiveness
Q6	There are structures and data types in L-PRISM that can be closely related, and changes in one can affect the other. Are these dependencies easy to be seen?	M6 - Hidden dependencies
Q7	Does L-PRISM generally seem easy or difficult to understand (for example, when changing different elements of a multimedia SFC)?	M7 - Hard mental operations

6.2 Procedure

For the evaluation of L-PRISM, four tasks ($T1 - T4$) were proposed, which had to be carried out using our L-PRISM proposal and a traditional method using our previous implementation with manual configuration through the web client interface. For the tasks to be completed, additional material was prepared and is available at the L-PRISM website[4]. The site contains information about L-PRISM and different examples of how to implement multimedia SFCs with L-PRISM and the traditional method used for these tests.

The types of components available for this experiment were two virtual devices. The first device produces a video of a white sphere bouncing on a black box, which we called *VD_video_ball*, and the second device produces a video of colored bars, which we called *VD_color_bar*. Three types of multimedia VNFs were also made available, namely (i) a multimedia VNF that transforms a color video to grayscale called *VMS_gray*, (ii) a multimedia VNF that only transfers UDP-UDP data and does not apply any changes to the original stream, called *VMS_udp*, and (iii) a multimedia VNF that merges video-type media streams, grabbing two multimedia streams and transforming them into one, called *VMS_merge*. The tasks are detailed as follows.

6.2.1 *T*1. This task is to develop a simple multimedia SFC, where *VMS_gray* will need to subscribe to the *VD_color_bar* to process the multimedia stream and publish the result to a point within the network (*IP* and *PORT*).

[4]https://fventuraq.github.io/lprism.html

6.2.2 *T*2. This task compares a raw multimedia stream with the same processed multimedia stream. To perform this task, it will be necessary to use *VMS_merge* that will receive the multimedia stream processed by *VMS_gray* and subscribe to *VD_color_bar* to receive the original stream, *VMS _gray* will need to subscribe to *VD_color_bar* in order to process this multimedia stream and send it to *VMS_merge*. Finally, *VMS_merge* will publish the result to a network point (*IP* and *PORT*).

6.2.3 *T*3. This task consists of adding the *VD_video_ball* multimedia stream to *T*2. It is necessary to create a new *VMS_merge* that will receive the result of *T*2 through one of its ports, and the other port will subscribe to *VD_video_ball*. Finally, it will publish its result in a given network point (*IP* and *PORT*). The visual result of *T*3 is presented in the bottom-right part of Figure 4, while Figure 5 illustrates the corresponding multimedia SFC.

6.2.4 *T*4. This task consists of replicating *T*3, changing the final destination port to where the final stream is sent. The idea is to be able to visualize two identical multimedia SFCs, created in *T*3 and *T*4, running at the same time.

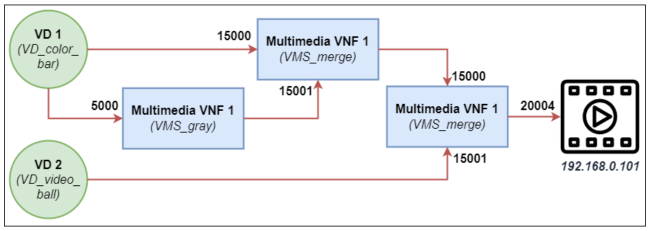


Figure 5: Model of task 3.

6.3 Participants

In order to carry on user experiments, the consent of each participant was requested through a Free and Informed Consent Form (FICF) and all collected data was anonymized. All responses were collected through a Google Forms questionnaire. No personal or sensible data was collected from participants, except their gender and age. Computer science students were invited to participate. Fifteen participants aged 20-39 years accepted our invitation. They were 14 men and 1 woman. Their academic degree was: seven were undergraduate, three were graduate, and five post-graduate students. We asked their level of experience on XML, JSON and YAML (from 1 (no experience) to 5 (a lot of experience)). The median answers were 4 for XML, 5 for JSON, and 3 for YAML. We built a testbed with edge nodes in our lab to run the experiments. Eight participants did it physically in the lab and seven participants did it remotely. All of them were observed all the time during the experiment, even the remote ones.

6.4 Analysis of Results

The results of our evaluation are presented in Figure 6. Our evaluation focuses on determining whether L-PRISM is efficient, productive, and easy to use.

Questions $Q2$ to $Q5$ of G1 are related to the productivity of L-PRISM. Figure 6(a) shows that the results of these questions are positive and the responses expected to validate L-PRISM were positive. With this, we can conclude that the $G1$ goal was achieved.

Questions $Q1$ and $Q6$ are related to efficiency. To answer $Q1$, Figure 6(b) shows the average time in minutes it took for participants to complete each task with both methods (L-PRISM and Traditional), with a confidence interval of 90%. It can be seen that Tasks 1, 2, and 3 have similar times, although the learning curve to learn a new language is much greater than the use of an intuitive interface, L-PRISM is equivalent to the traditional method for new developers of multimedia SFCs.

In the case of Task 4, L-PRISM has proven to be superior to the traditional method. This is because developers came to understand L-PRISM better with the previous tasks, and most developers perceived that they could reuse the code from Task 3 for Task 4. This task also answers $Q6$ of $G1$, which focuses on evaluating if L-PRISM allows the reuse of an already created multimedia SFC. As it can be seen in Figure 6(b), using L-PRISM, participants completed Task 4 in an average time 77.8% shorter than with the traditional method, due to L-PRISM code reuse.

The answers to the G2 questions related to the cognitive dimensions are presented in Figure 6(c). It should be noted that, except for $Q3$ about language diffuseness (verbosity), where a negative response was expected, all the others had positive feedback from the subjects. As it can be seen, the responses for Q3 are somewhat ambiguous. A subsequent consultation was made with the participants asking about this result and the conclusion was that the question was not very clear for them, hence the disparity in the responses.

As mentioned above, the objectives of our evaluation were to demonstrate that L-PRISM is efficient, productive, and easy to use. Moreover, based on the results obtained, we can conclude that goals $G1$ and $G2$ were achieved.

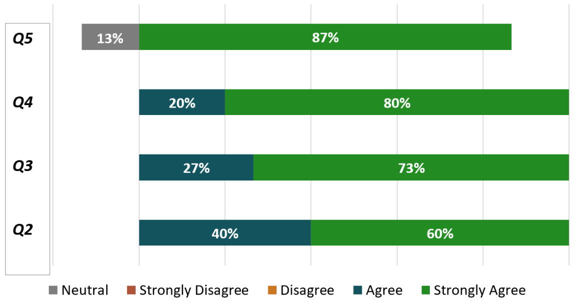
7 CONCLUSION

This work presented L-PRISM, a DSL to specify multimedia SFCs based on multimedia VNFs. We also extended the ALFA platform with the ability to execute SFCs described with L-PRISM. This extension is available in our GitHub[5] repository, where the installation guide and all necessary resources for installing, executing, and testing our work are provided.

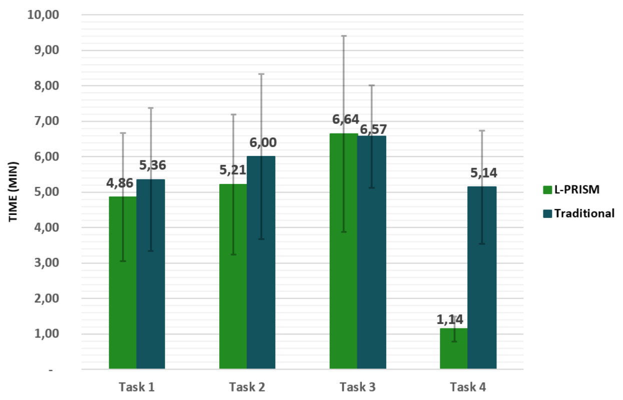
We evaluated L-PRISM, and the results confirmed that our proposal language is efficient, productive, and easy to use. One observation we made was that the learning curve was shorter for the traditional model, as it focused on learning to use a specific application. In contrast, the learning curve for L-PRISM largely depended on the user's prior experience with programming languages; those with more experience adapted to our language more quickly and, therefore, completed tasks faster.

Finally, the practical applicability of L-PRISM will mainly depend on multimedia VNFs, as these are the core components of multimedia SFCs and are responsible for processing multimedia streams. L-PRISM can be used to deploy applications such as virtual/augmented reality, live streaming, surveillance systems, and more.

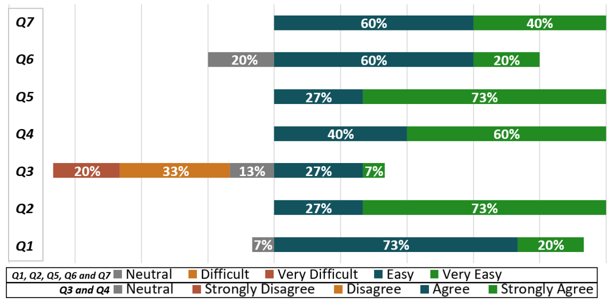
[5]Source code of alfa 2.0 https://github.com/fventuraq/alfa



(a) Participant Responses for Questions Q2-Q5 of G1.



(b) Time needed per task in L-PRISM and Traditional methods.



(c) Participant Responses for G2 Questions.

Figure 6: Evaluation results.

As future work, we intend to develop a graphical tool to create/edit SFCs designed with L-PRISM, offering an intuitive user interface to visualize and modify their topology. We will also propose the integration of resource allocation and scaling algorithms into ALFA 2.0.

ACKNOWLEDGMENT

This work was supported by CAPES, CAPES-Print, CNPq, FAPERJ and INCT-ICoNIoT.

REFERENCES

[1] ETSI GS NFV-IFA 011. 2023. Network Functions Virtualisation (NFV) Release 4; Management and Orchestration; VNF Descriptor and Packaging Specification. https://docbox.etsi.org/ISG/NFV/open/Publications_pdf/Specs-Reports/NFV-IFA%20011v4.5.1%20-%20GS%20-%20VNF%20Packaging%20Spec.pdf

[2] ETSI GS NFV-IFA 014. 2021. Network Functions Virtualisation (NFV) Release 4; Management and Orchestration; Network Service Templates Specification. https://docbox.etsi.org/ISG/NFV/open/Publications_pdf/Specs-Reports/NFV-IFA%20014v4.2.1%20-%20GS%20-%20Network%20Service%20Templates%20Spec.pdf

[3] Anselmo Luiz Éden Battisti, Débora Christina Muchaluat-Saade, and Flávia C. Delicato. 2020. V-PRISM: An Edge-Based IoT Architecture to Virtualize Multimedia Sensors. In *2020 IEEE 6th World Forum on Internet of Things (WF-IoT)*. 1–6. https://doi.org/10.1109/WF-IoT48130.2020.9221199

[4] Anselmo Luiz Éden Battisti, Debora Christina Muchaluat-Saade, and Flavia C Délicato. 2021. Enabling Internet of Media Things with edge-based virtual multimedia sensors. *IEEE Access* 9 (2021), 59255–59269.

[5] Deval Bhamare, Raj Jain, Mohammed Samaka, and Aiman Erbad. 2016. A survey on service function chaining. *Journal of Network and Computer Applications* 75 (Nov. 2016), 138–155. https://doi.org/10.1016/j.jnca.2016.09.001

[6] Martin Björklund. 2010. YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF). RFC 6020. https://doi.org/10.17487/RFC6020

[7] Alan F Blackwell and Thomas RG Green. 2000. A Cognitive Dimensions questionnaire optimised for users. In *PPIG*, Vol. 13. Citeseer.

[8] José Castillo-Lema, Augusto Venâncio Neto, Flávio de Oliveira, and Sergio Takeo Kofuji. 2019. Mininet-NFV: Evolving Mininet with OASIS TOSCA NVF profiles Towards Reproducible NFV Prototyping. In *2019 IEEE Conference on Network Softwarization (NetSoft)*. 506–512. https://doi.org/10.1109/NETSOFT.2019.8806686

[9] Mario Di Mauro, Giovanni Galatro, Maurizio Longo, Fabio Postiglione, and Marco Tambasco. 2021. Comparative Performability Assessment of SFCs: The Case of Containerized IP Multimedia Subsystem. *IEEE Transactions on Network and Service Management* 18, 1 (2021), 258–272. https://doi.org/10.1109/TNSM.2020.3044232

[10] GS NFV-SOL 001 ETSI. 2022. Network Functions Virtualisation (NFV) Release 4; Protocols and Data Models; NFV descriptors based on TOSCA specification. https://www.etsi.org/deliver/etsi_gs/NFV-SOL/001_099/001/04.03.01_60/gs_NFV-SOL001v040301p.pdf

[11] A.H. Ghorab, A. Kusedghi, M. A. Nourian, and A. Akbari. 2020. Joint VNF Load Balancing and Service Auto-Scaling in NFV with Multimedia Case Study. In *2020 25th International Computer Conference, Computer Society of Iran (CSICC)*. 1–7. https://doi.org/10.1109/CSICC49403.2020.9050122

[12] Heiko Koziolek. 2008. Goal, question, metric. In *Dependability metrics*. Springer, 39–42.

[13] Rensis Likert. 1932. A technique for the measurement of attitudes. *Archives of psychology* (1932).

[14] Yuyi Mao, Changsheng You, Jun Zhang, Kaibin Huang, and Khaled B. Letaief. 2017. A Survey on Mobile Edge Computing: The Communication Perspective. *IEEE Comm. Surveys and Tutorials* 19, 4 (2017), 2322–2358. https://doi.org/10.1109/COMST.2017.2745201 arXiv:1701.01090

[15] Marjan Mernik, Jan Heering, and Anthony M Sloane. 2005. When and how to develop domain-specific languages. *ACM computing surveys (CSUR)* 37, 4 (2005), 316–344.

[16] Rashid Mijumbi, Joan Serrat, Juan Luis Gorricho, Niels Bouten, Filip De Turck, and Raouf Boutaba. 2016. Network function virtualization: State-of-the-art and research challenges. *IEEE Communications Surveys and Tutorials* 18, 1 (2016), 236–262. https://doi.org/10.1109/COMST.2015.2477041 arXiv:1509.07675

[17] Ali Nauman, Yazdan Ahmad Qadri, Muhammad Amjad, Yousaf Bin Zikria, Muhammad Khalil Afzal, and Sung Won Kim. 2020. Multimedia Internet of Things: A comprehensive survey. *Ieee Access* 8 (2020), 8202–8250.

[18] Eman Negm, Soha Makady, and Akram Salah. 2019. Survey on domain specific languages implementation aspects. *International Journal of Advanced Computer Science and Applications* 10, 11 (2019).

[19] Guto Leoni Santos, Diego de Freitas Bezerra, Élisson da Silva Rocha, Leylane Ferreira, André Luis Cavalcanti Moreira, Glauco Estácio Gonçalves, Maria Valéria Marquezini, Ákos Recse, Amardeep Mehta, Judith Kelner, Djamel Sadok, and Patricia Takako Endo. 2022. Service Function Chain Placement in Distributed Scenarios: A Systematic Review. *Journal of Network and Systems Management* 30, 1 (2022), 1–39. https://doi.org/10.1007/s10922-021-09626-4

[20] Jürgen Schönwälder, Martin Björklund, and Phil Shafer. 2010. Network configuration management using NETCONF and YANG. *IEEE communications magazine* 48, 9 (2010), 166–173.

[21] Priyanka Surendran et al. 2012. Technology acceptance model: A survey of literature. *International Journal of Business and Social Research* 2, 4 (2012), 175–178.

[22] OASIS TOSCA. 2017. TOSCA Simple Profile for Network Functions Virtualization (NFV) Version 1.0, Committee Specification Draft 04. https://docs.oasis-open.org/tosca/tosca-nfv/v1.0/csd04/tosca-nfv-v1.0-csd04.html

[23] YAML. [n. d.]. YAML Ain't Markup Language (YAML™) version 1.2. https://yaml.org/spec/1.2/spec.html. Accessed: Jun 29, 2024.

[24] Bo Yi, Xingwei Wang, Keqin Li, Sajal k. Das, and Min Huang. 2018. A comprehensive survey of Network Function Virtualization. *Computer Networks* 133 (2018), 212–262. https://doi.org/10.1016/j.comnet.2018.01.021