WebMedia
30º SIMPÓSIO BRASILEIRO DE SISTEMAS MULTIMÍDIA E WEB 2024

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

30th Brazilian Symposium on Multimedia and Web Systems
Juiz de Fora - MG, Brazil
Thursday, October 17th, 2024 - 8:30 AM

L-PRISM

Authors:
Franklin Jordan Ventura Quico (***Presenter***)
Anselmo L. E. Battisti
Débora Muchaluat-Saade
Flávia Coimbra Delicato

Laboratório MídiaCom

uff Universidade Federal Fluminense

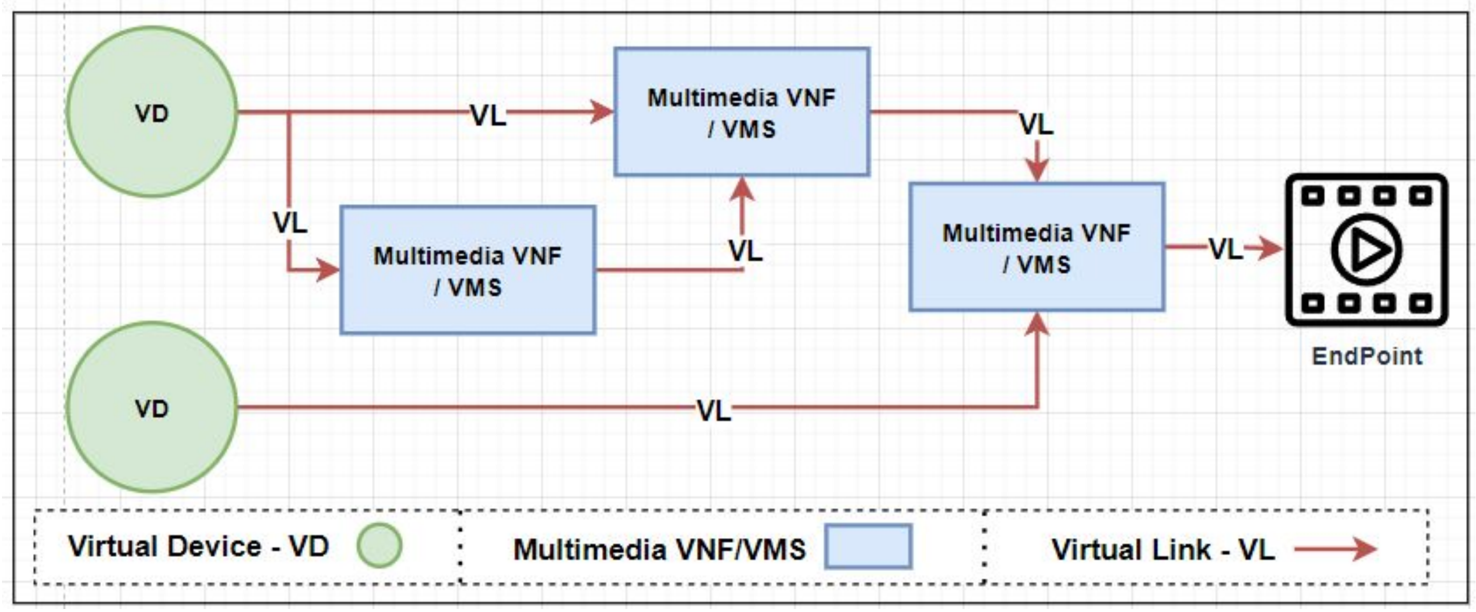
Instituto de Computação

SUMMARY

- Introduction
- Goal
- L-PRISM Proposal
- ALFA 2.0
- Evaluation
- Conclusion

INTRODUCTION - Multimedia SFC

A Service Function Chain (SFC) is an **ordered sequence of** network functions (**VNFs**) through which data traffic flows.



Multimedia Service Function Chain
Multimedia SFC

INTRODUCTION - DSL

A Domain-Specific Language (DSL) is a language specifically designed to **describe** and **manage** aspects within a specific domain.

Advantages of a DSL:

- ***Increased productivity***: **Simplifies** and **accelerates** development in the specific domain.
- ***Readability***: Uses domain-specific **terminology**, making the code easier to understand.
- ***Fewer errors***: Reduces common errors by focusing on a narrow area.
- ***Better maintainability***: Facilitates system evolution and adaptation.

GOALS

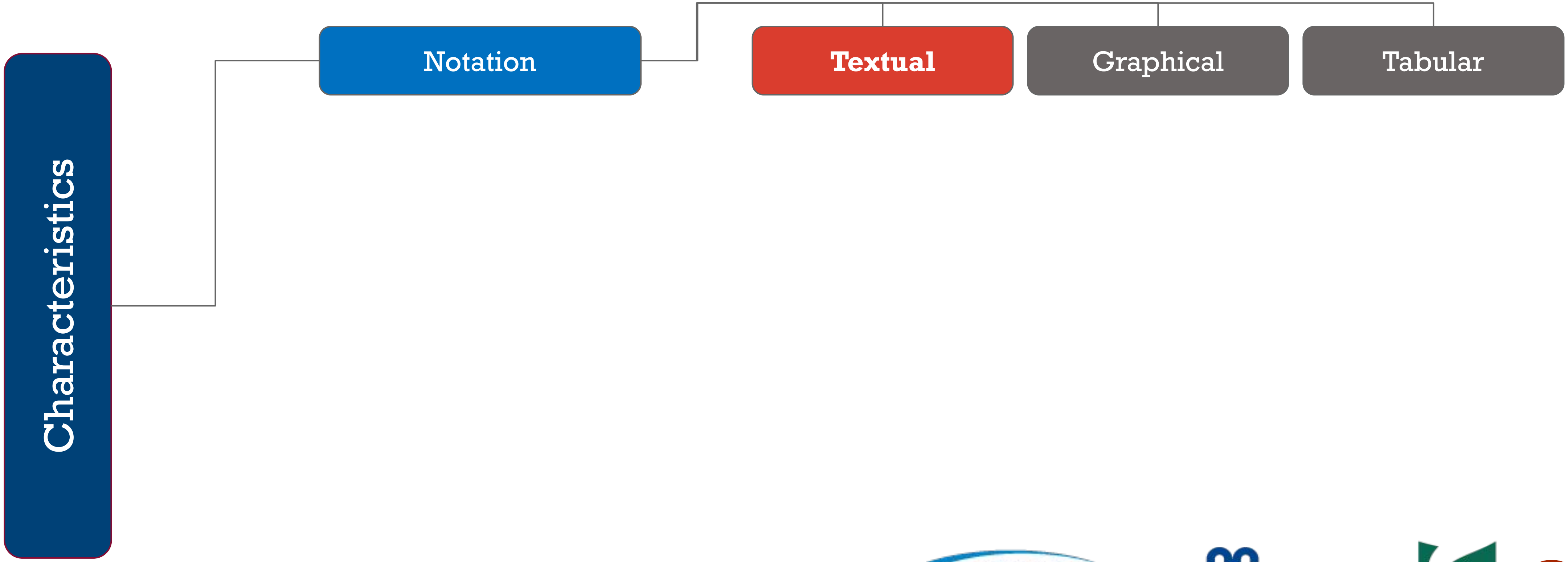
Propose a Domain-Specific Language (DSL) for creating Multimedia Service Function Chains based on Virtualization of Sensors.

CONTRIBUTIONS

- L-PRISM makes it possible to **describe** a multimedia SFC based on a sequence of multimedia VNFs.
- L-PRISM defines the necessary structures for **registering** 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.

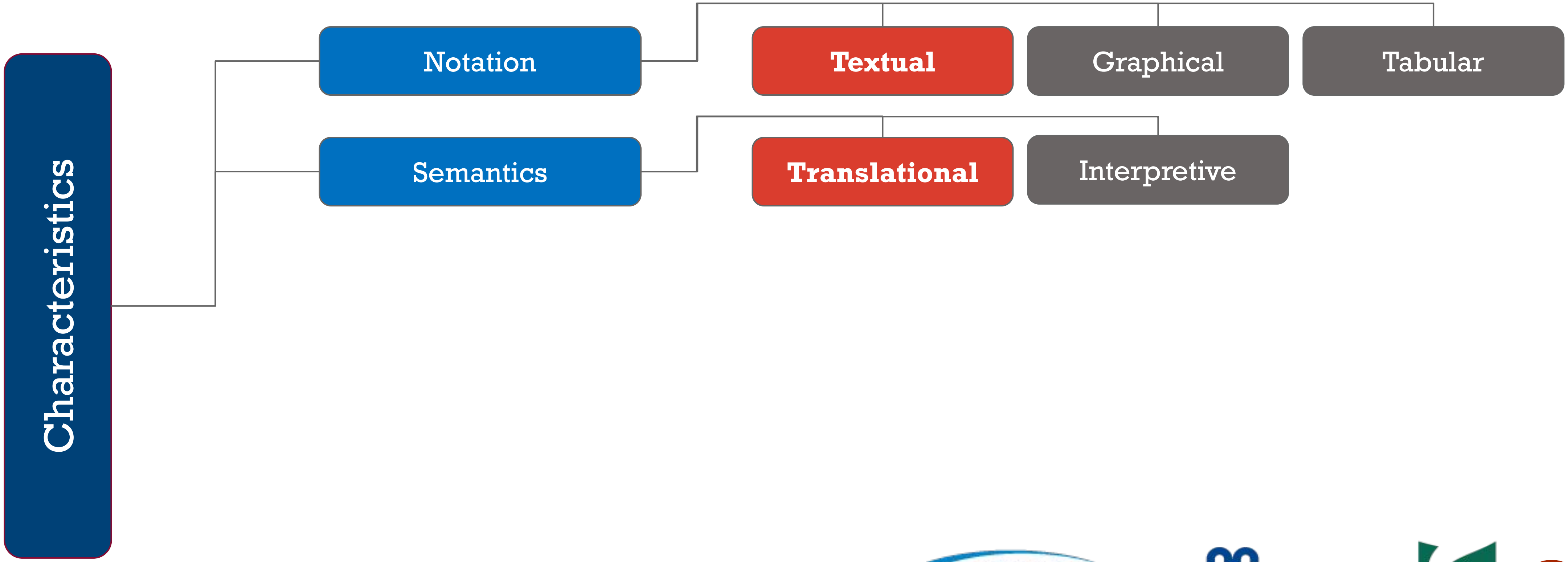
L-PRISM Proposal

- The implementation of L-PRISM follows the compilation-based DSL implementation approach since we focus on defining a high-level language, which will be translated into a low-level language to be compiled.



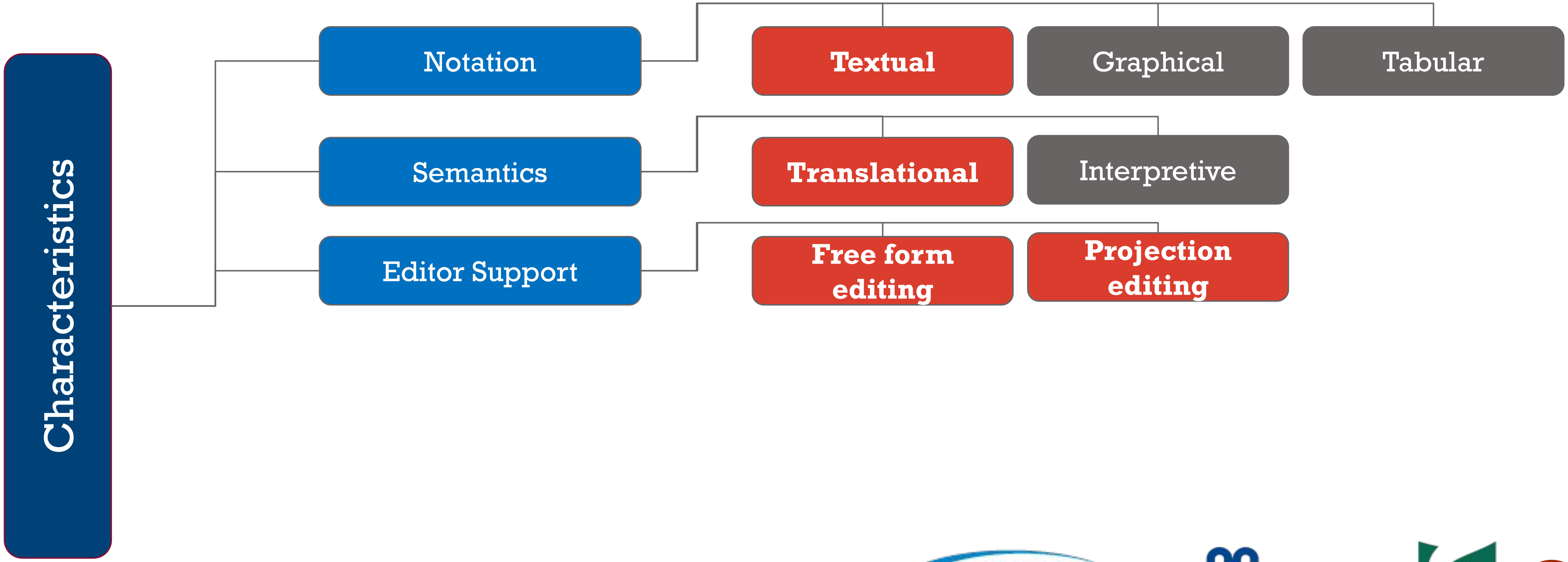
L-PRISM Proposal

- The implementation of L-PRISM follows the compilation-based DSL implementation approach since we focus on defining a high-level language, which will be translated into a low-level language to be compiled.



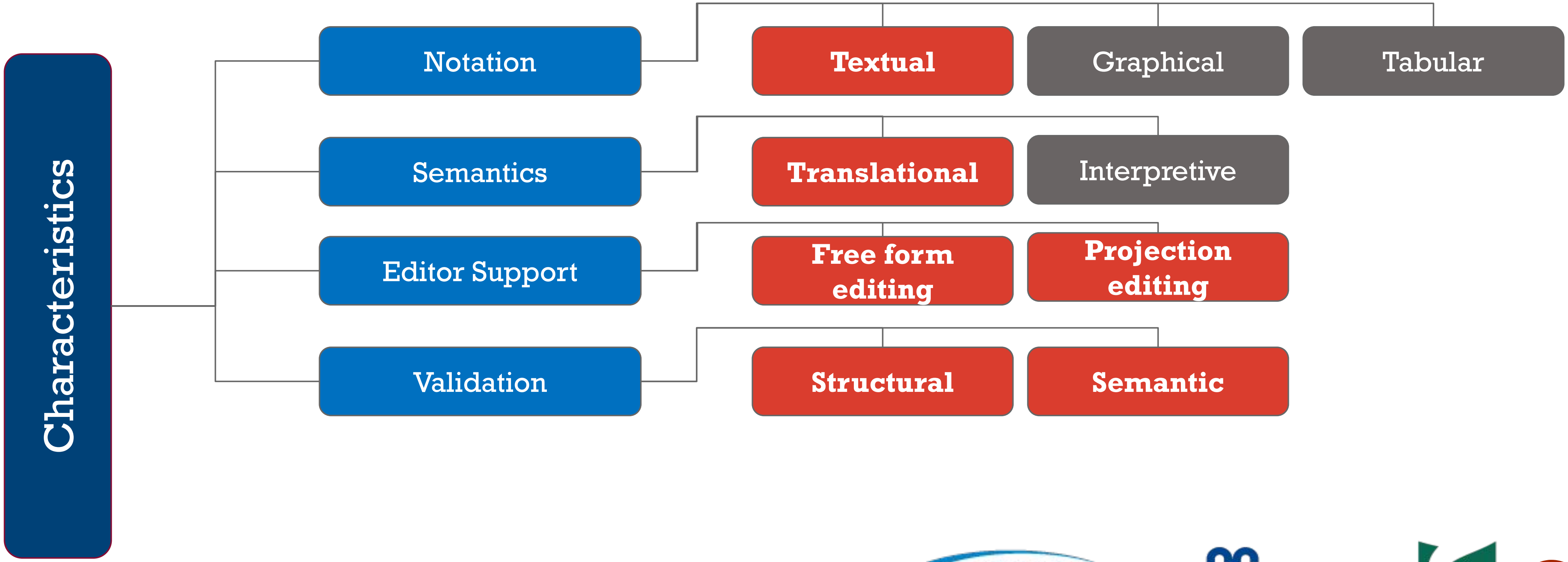
L-PRISM Proposal

- The implementation of L-PRISM follows the compilation-based DSL implementation approach since we focus on defining a high-level language, which will be translated into a low-level language to be compiled.



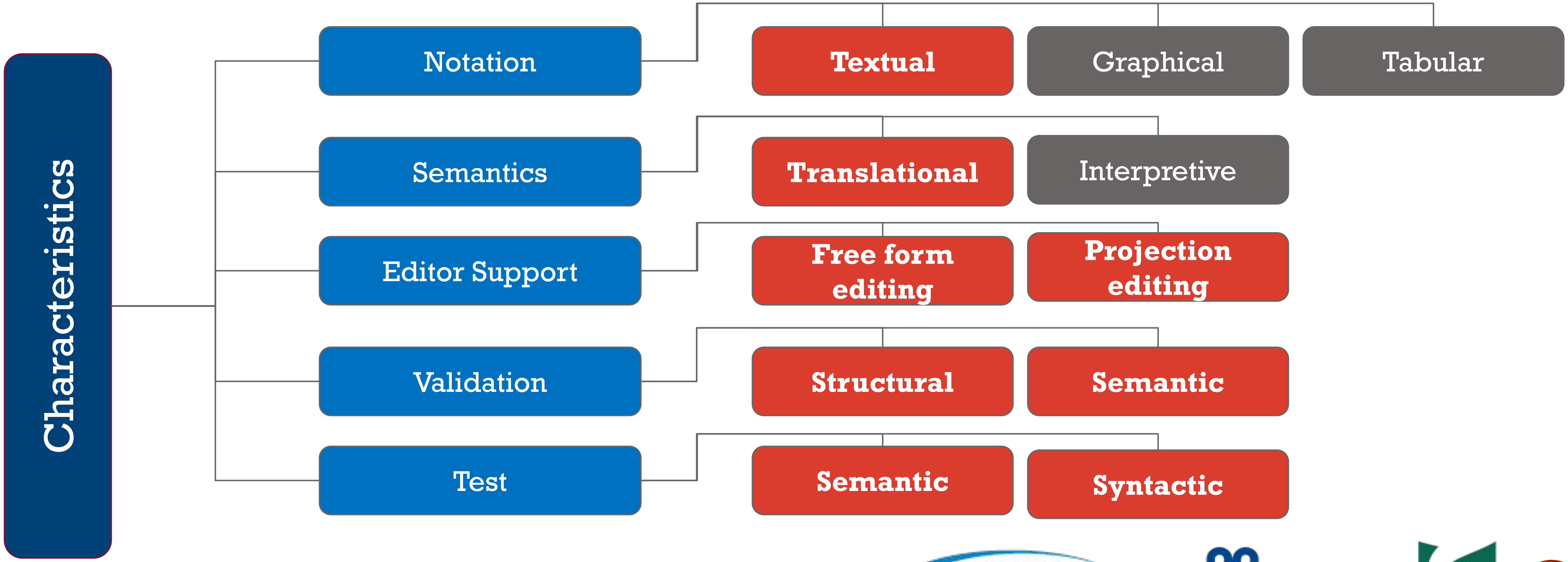
L-PRISM Proposal

- The implementation of L-PRISM follows the compilation-based DSL implementation approach since we focus on defining a high-level language, which will be translated into a low-level language to be compiled.



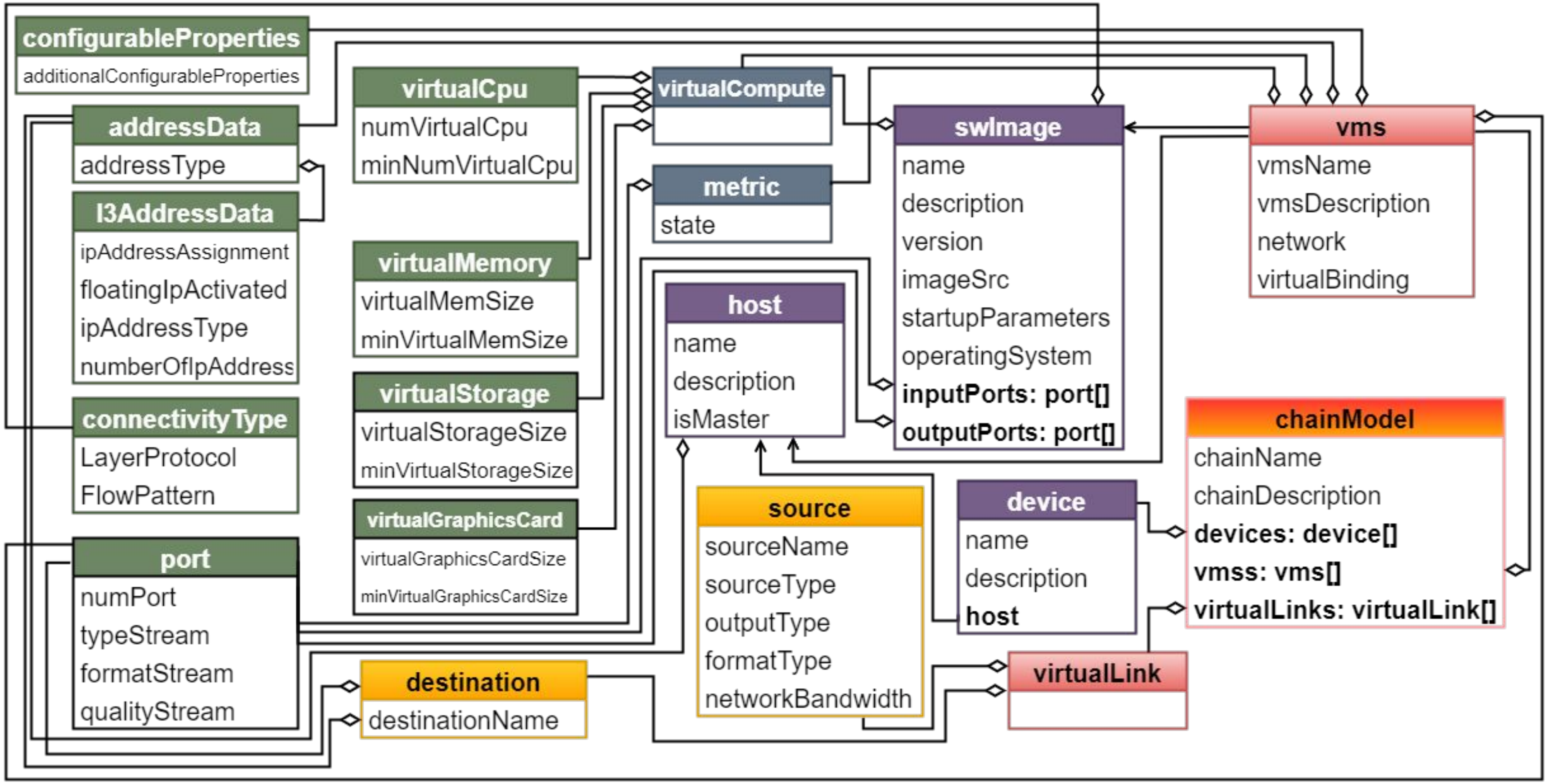
L-PRISM Proposal

- The implementation of L-PRISM follows the compilation-based DSL implementation approach since we focus on defining a high-level language, which will be translated into a low-level language to be compiled.

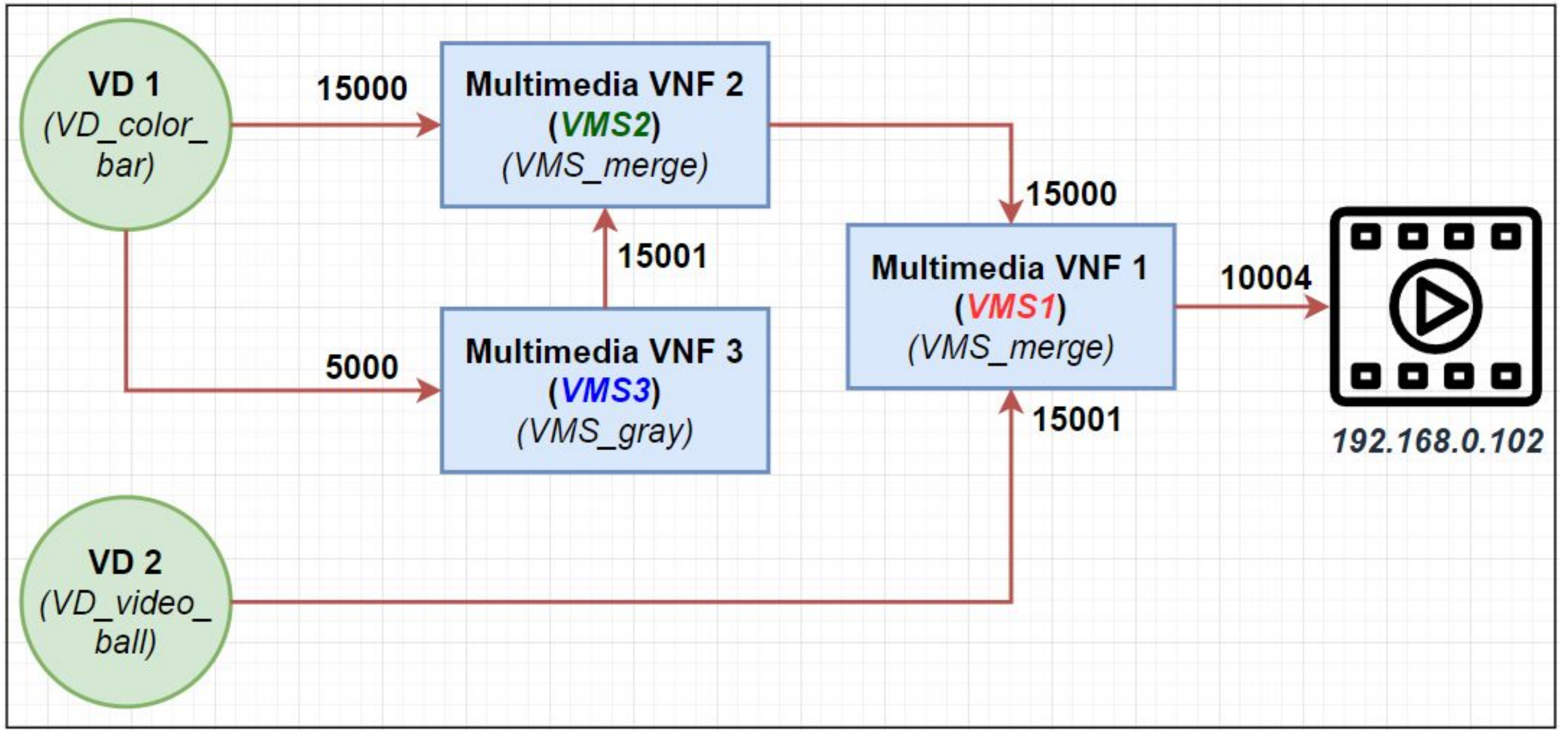


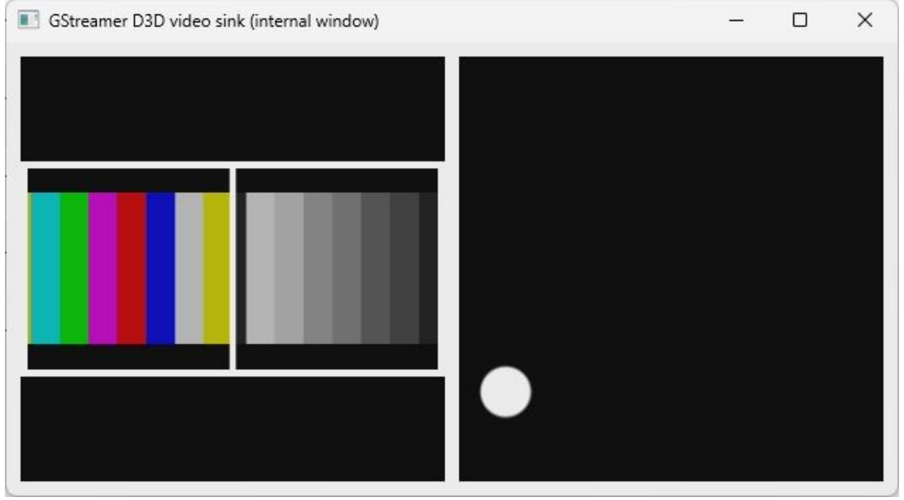
L-PRISM METAMODEL

- L-PRISM follows a model-based approach and is based on the analysis and design of the multimedia application.

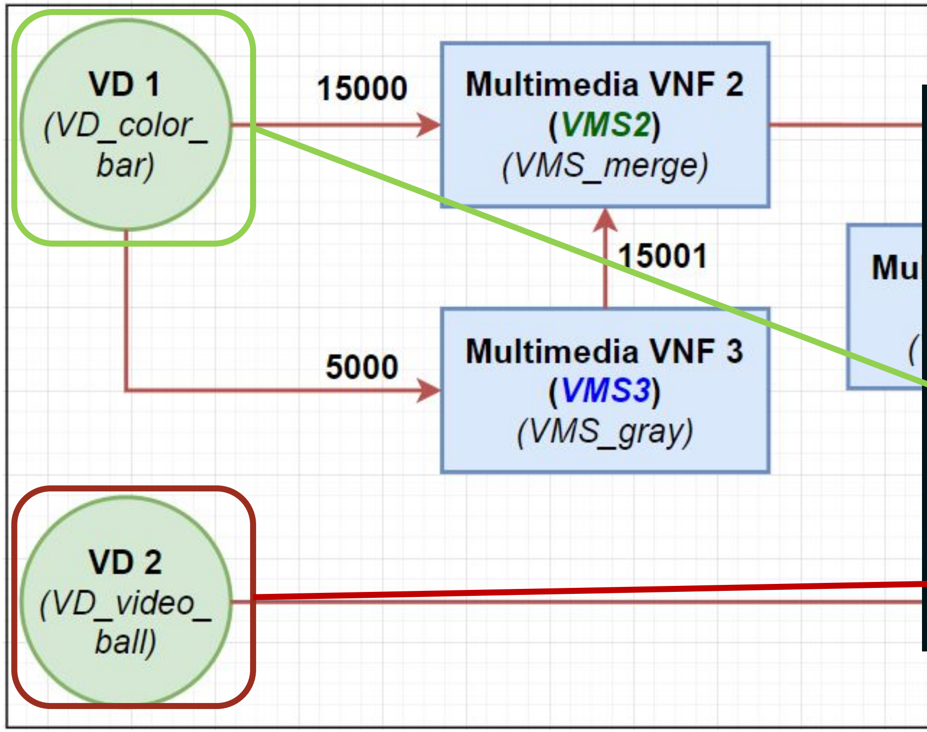


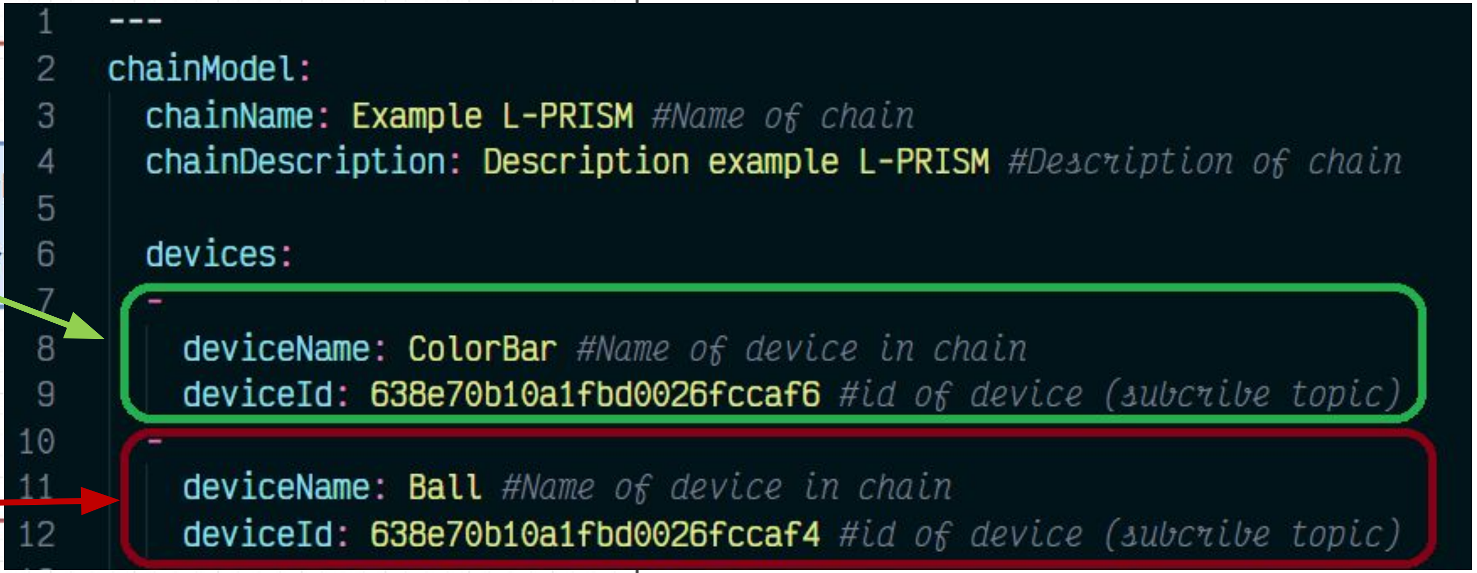
L-PRISM - EXAMPLE





L-PRISM - EXAMPLE

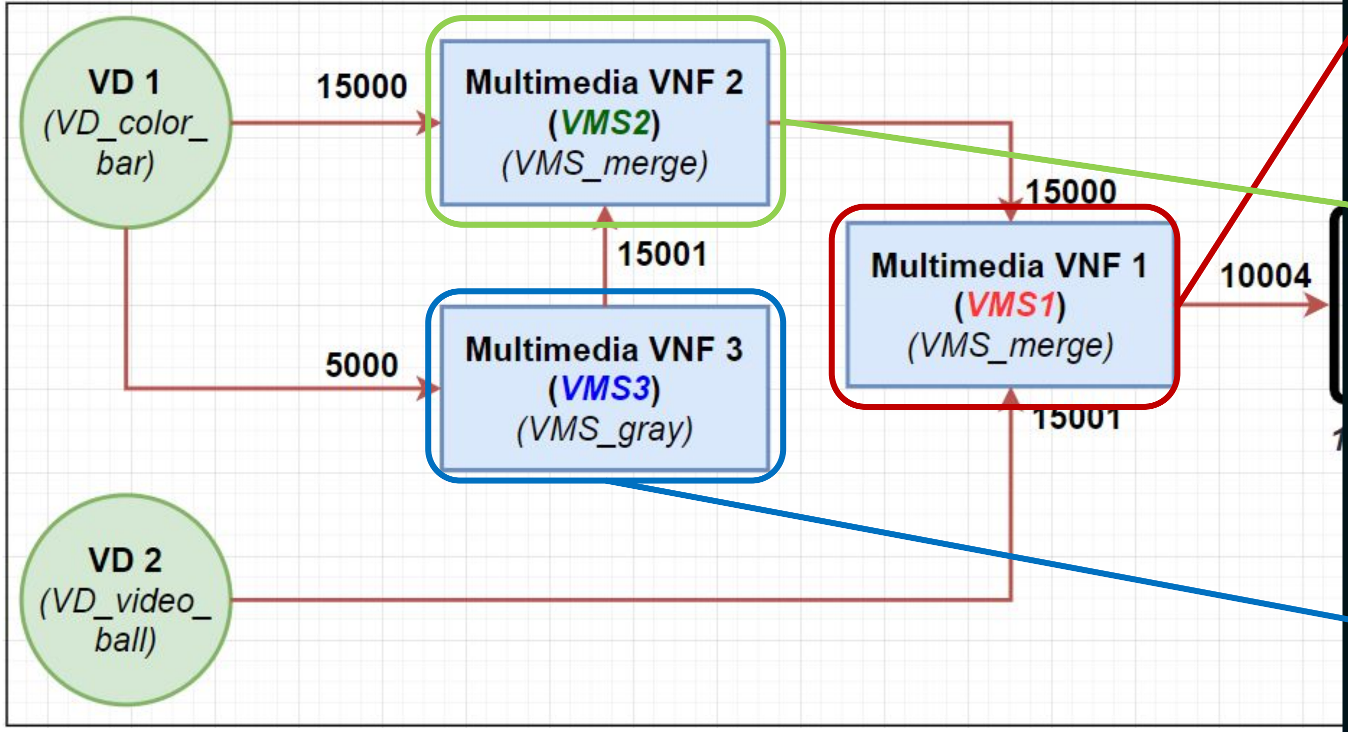


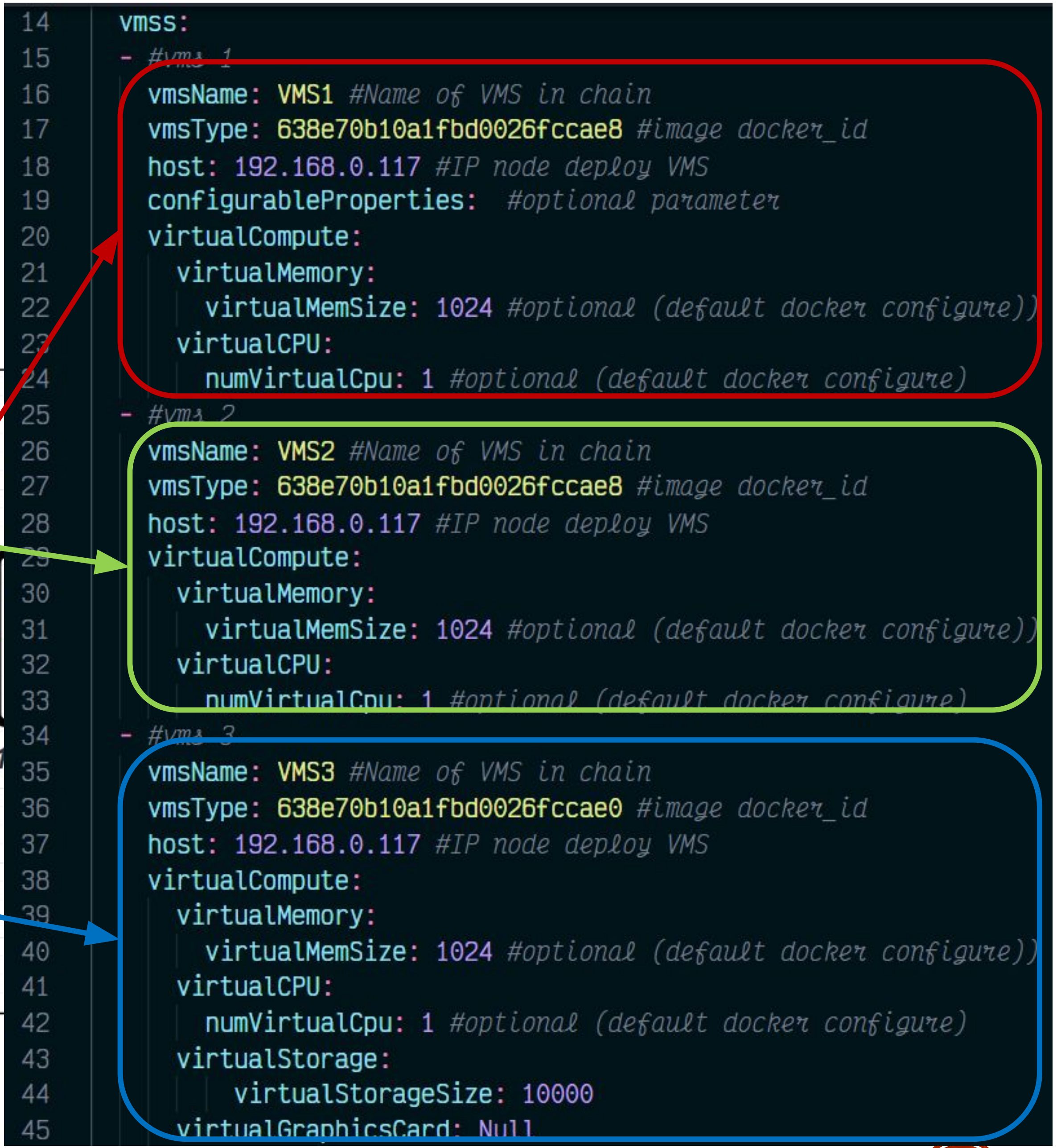


```yaml
---
chainModel:
  chainName: Example L-PRISM #Name of chain
  chainDescription: Description example L-PRISM #Description of chain

  devices:
    -
      deviceName: ColorBar #Name of device in chain
      deviceId: 638e70b10a1fbd0026fccaf6 #id of device (subcribe topic)
    -
      deviceName: Ball #Name of device in chain
      deviceId: 638e70b10a1fbd0026fccaf4 #id of device (subcribe topic)
```

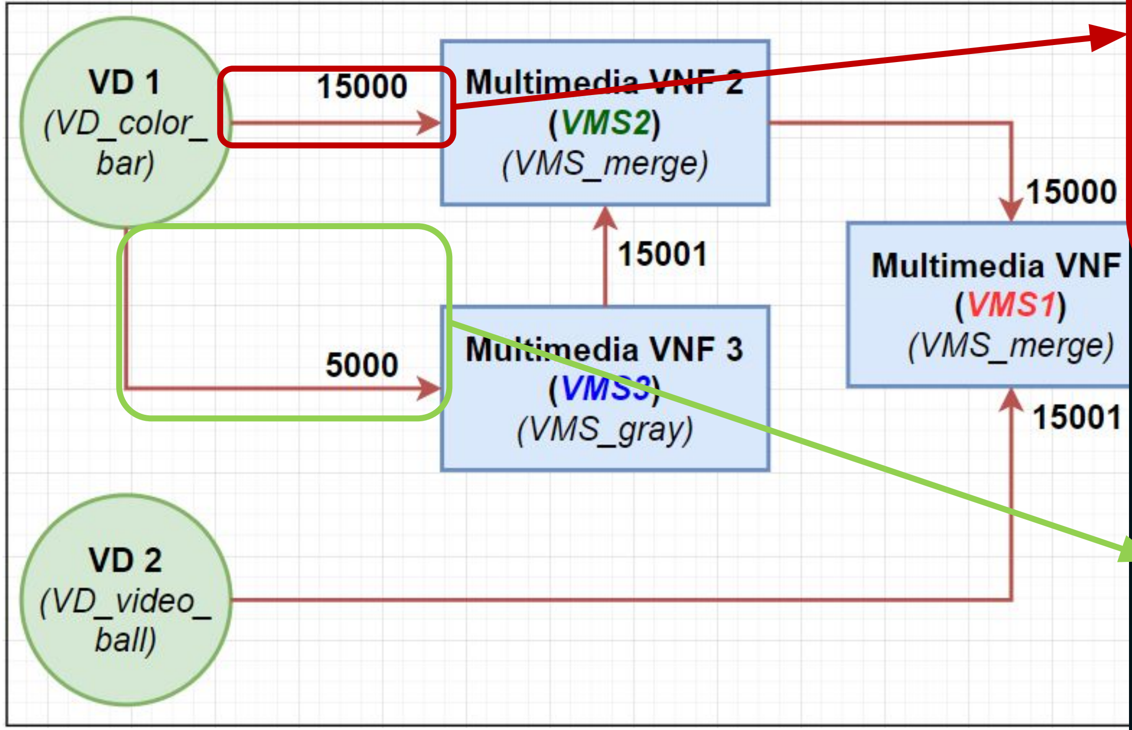
L-PRISM - EXAMPLE

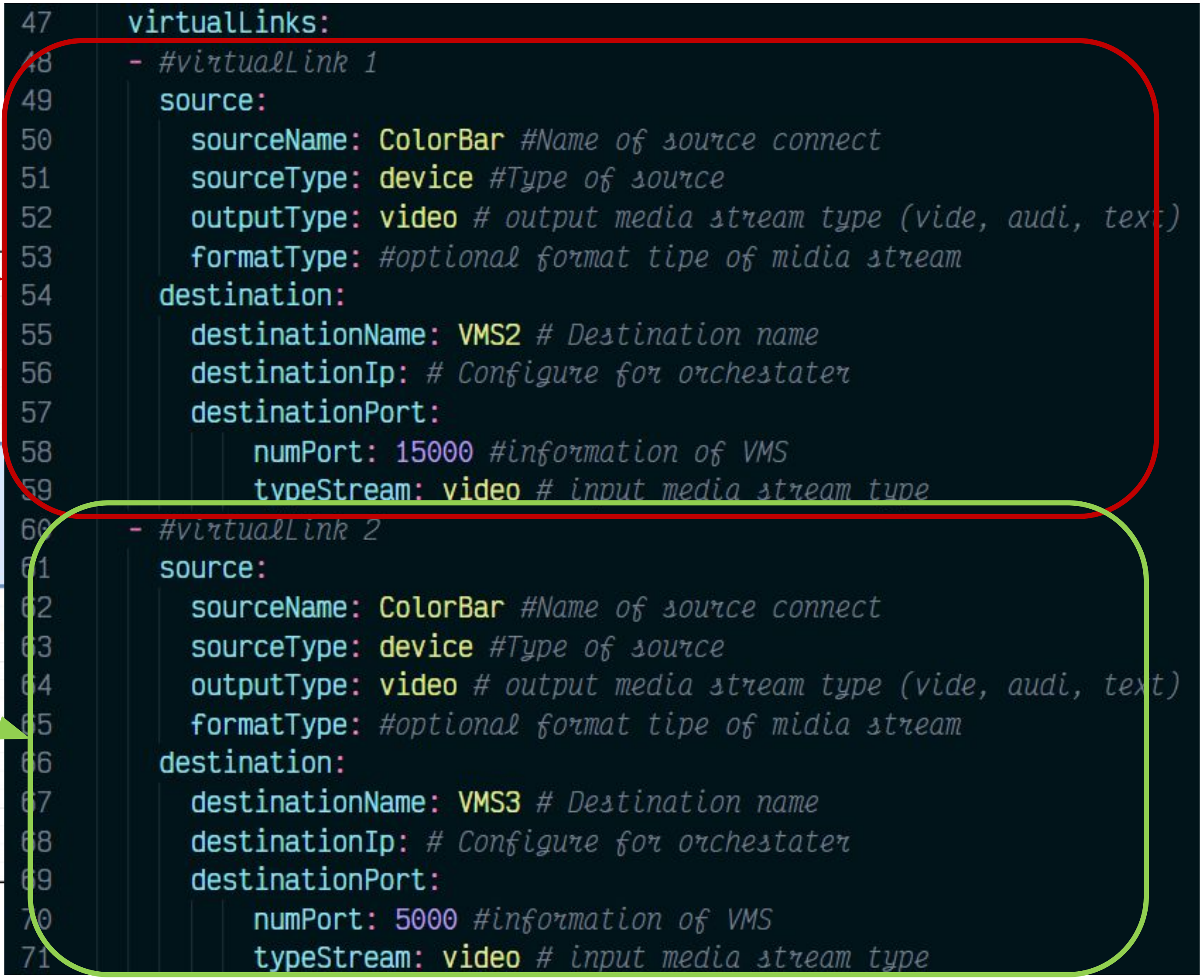




```yaml
vmss:
- #vms 1
  vmsName: VMS1 #Name of VMS in chain
  vmsType: 638e70b10a1fbd0026fccae8 #image docker_id
  host: 192.168.0.117 #IP node deploy VMS
  configurableProperties:  #optional parameter
  virtualCompute:
    virtualMemory:
      virtualMemSize: 1024 #optional (default docker configure))
    virtualCPU:
      numVirtualCpu: 1 #optional (default docker configure)
- #vms 2
  vmsName: VMS2 #Name of VMS in chain
  vmsType: 638e70b10a1fbd0026fccae8 #image docker_id
  host: 192.168.0.117 #IP node deploy VMS
  virtualCompute:
    virtualMemory:
      virtualMemSize: 1024 #optional (default docker configure))
    virtualCPU:
      numVirtualCpu: 1 #optional (default docker configure)
- #vms 3
  vmsName: VMS3 #Name of VMS in chain
  vmsType: 638e70b10a1fbd0026fccae0 #image docker_id
  host: 192.168.0.117 #IP node deploy VMS
  virtualCompute:
    virtualMemory:
      virtualMemSize: 1024 #optional (default docker configure))
    virtualCPU:
      numVirtualCpu: 1 #optional (default docker configure)
    virtualStorage:
        virtualStorageSize: 10000
    virtualGraphicsCard: Null
```

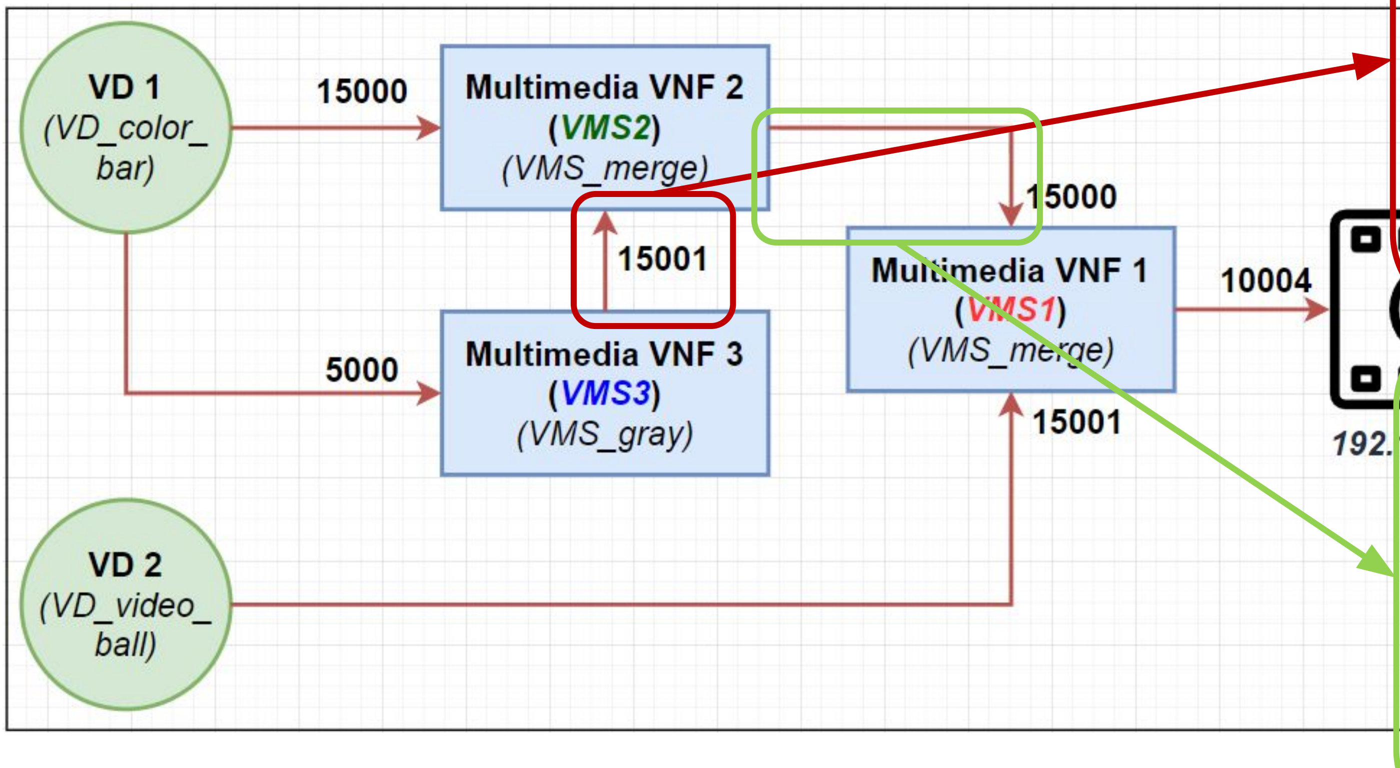
L-PRISM - EXAMPLE

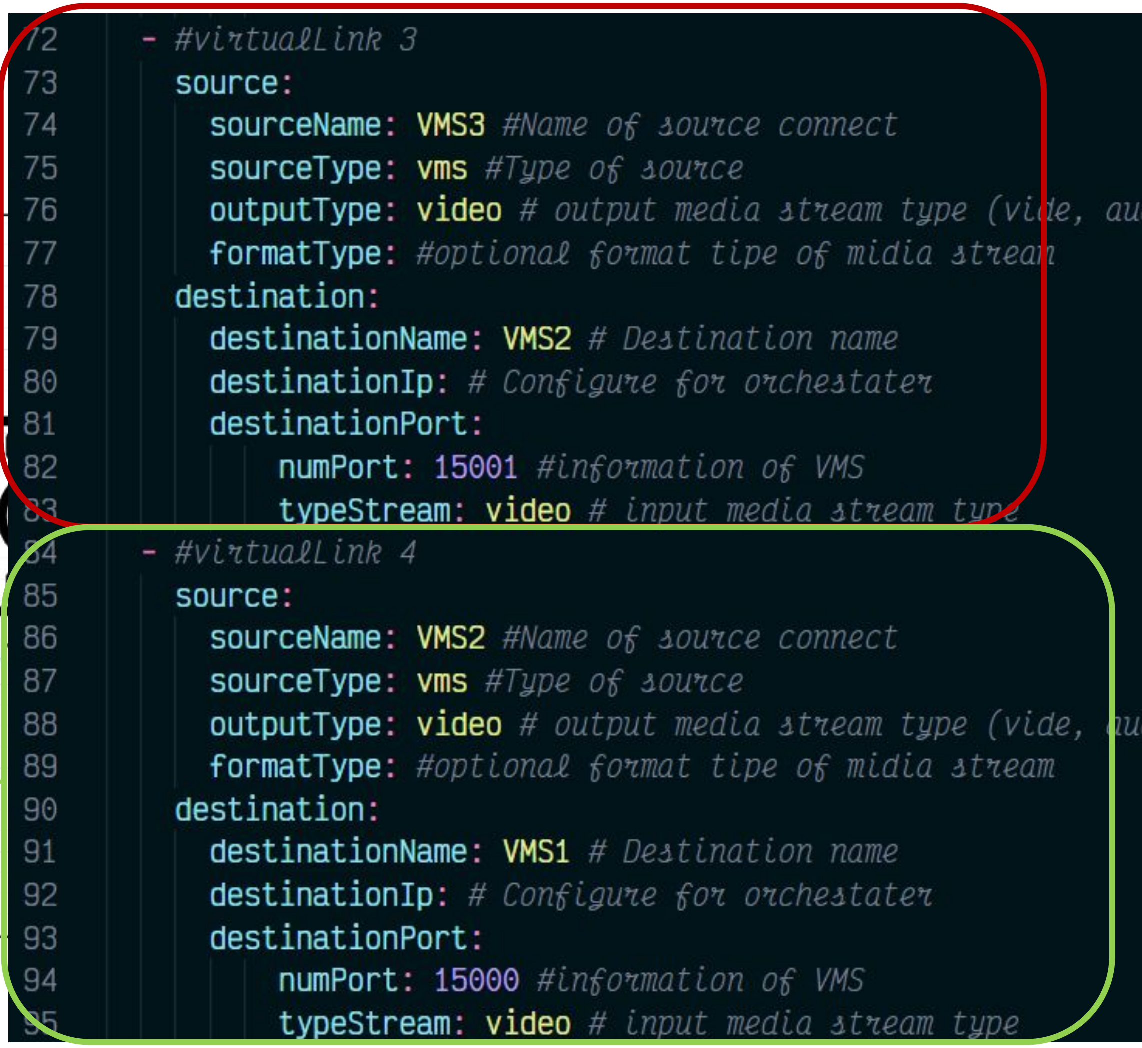




```yaml
virtualLinks:
  - #virtualLink 1
    source:
      sourceName: ColorBar #Name of source connect
      sourceType: device #Type of source
      outputType: video # output media stream type (vide, audi, text)
      formatType: #optional format tipe of midia stream
    destination:
      destinationName: VMS2 # Destination name
      destinationIp: # Configure for orchestater
      destinationPort:
          numPort: 15000 #information of VMS
          typeStream: video # input media stream type
  - #virtualLink 2
    source:
      sourceName: ColorBar #Name of source connect
      sourceType: device #Type of source
      outputType: video # output media stream type (vide, audi, text)
      formatType: #optional format tipe of midia stream
    destination:
      destinationName: VMS3 # Destination name
      destinationIp: # Configure for orchestater
      destinationPort:
          numPort: 5000 #information of VMS
          typeStream: video # input media stream type
```

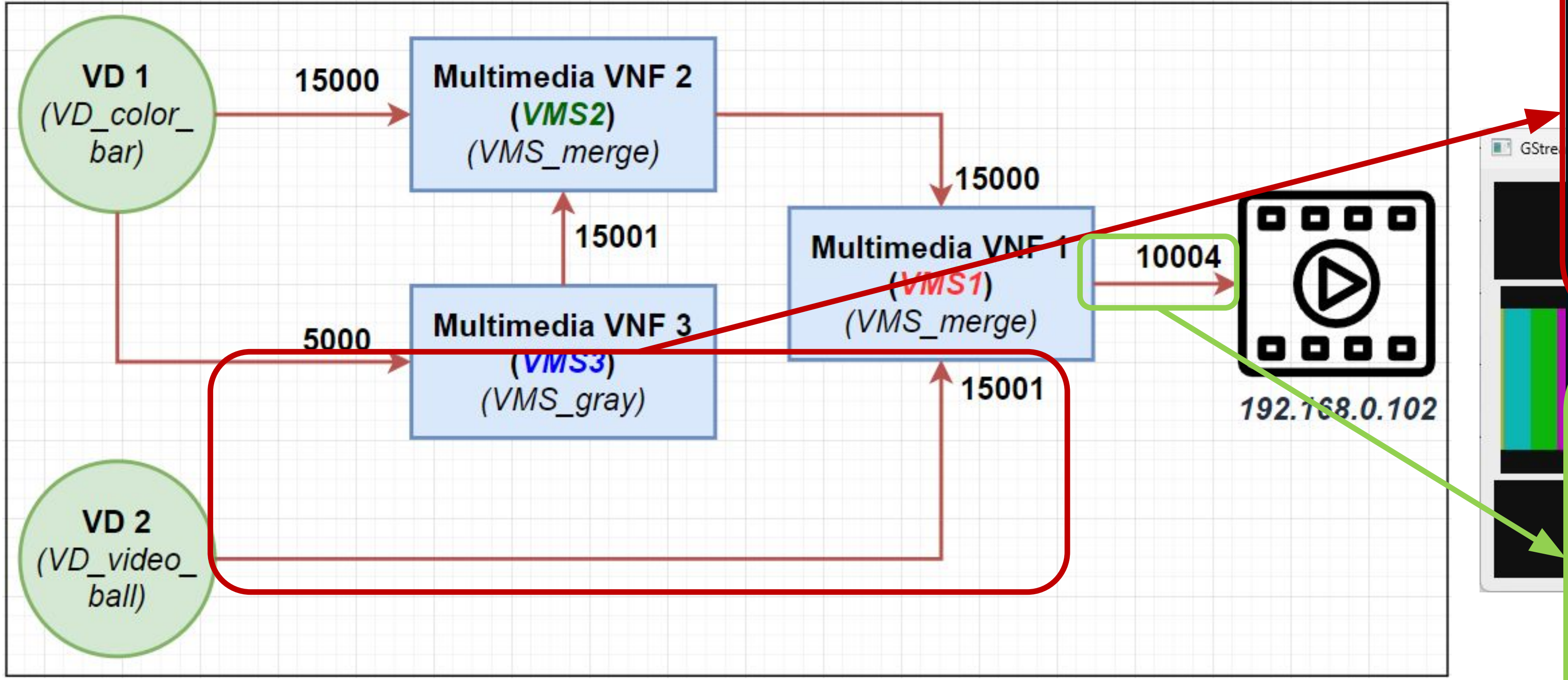
L-PRISM - EXAMPLE

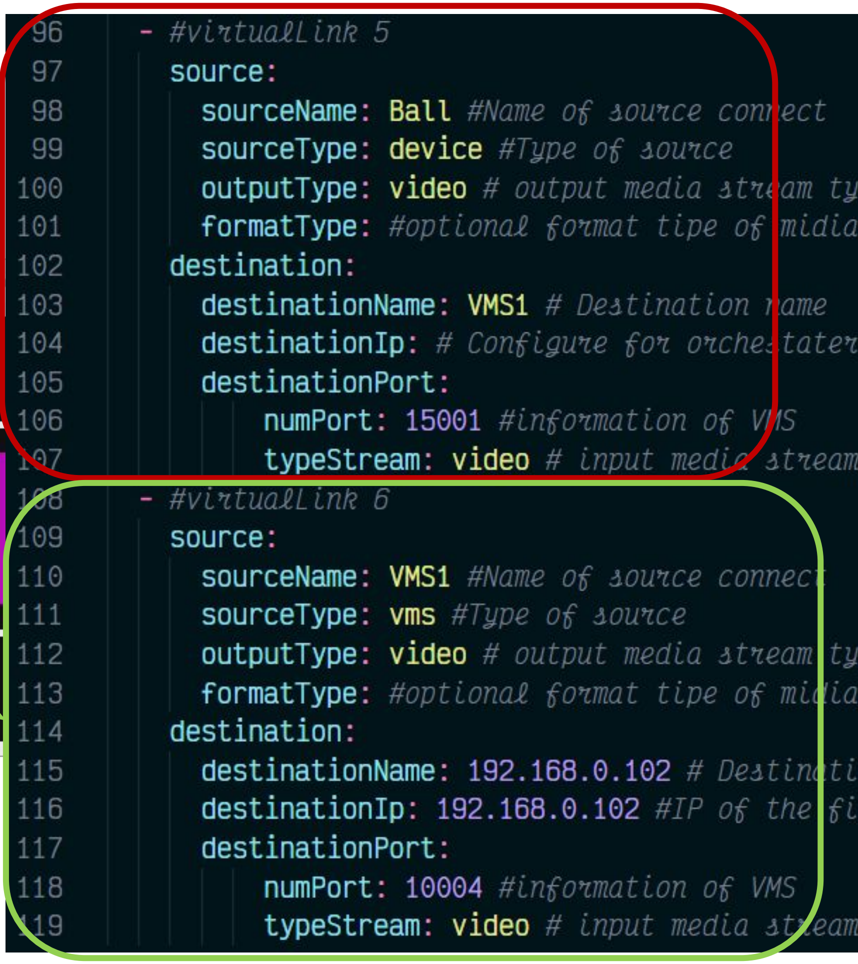




```
- #virtualLink 3
  source:
    sourceName: VMS3 #Name of source connect
    sourceType: vms #Type of source
    outputType: video # output media stream type (vide, au
    formatType: #optional format tipe of midia stream
  destination:
    destinationName: VMS2 # Destination name
    destinationIp: # Configure for orchestater
    destinationPort:
        numPort: 15001 #information of VMS
        typeStream: video # input media stream type
- #virtualLink 4
  source:
    sourceName: VMS2 #Name of source connect
    sourceType: vms #Type of source
    outputType: video # output media stream type (vide, au
    formatType: #optional format tipe of midia stream
  destination:
    destinationName: VMS1 # Destination name
    destinationIp: # Configure for orchestater
    destinationPort:
        numPort: 15000 #information of VMS
        typeStream: video # input media stream type
```

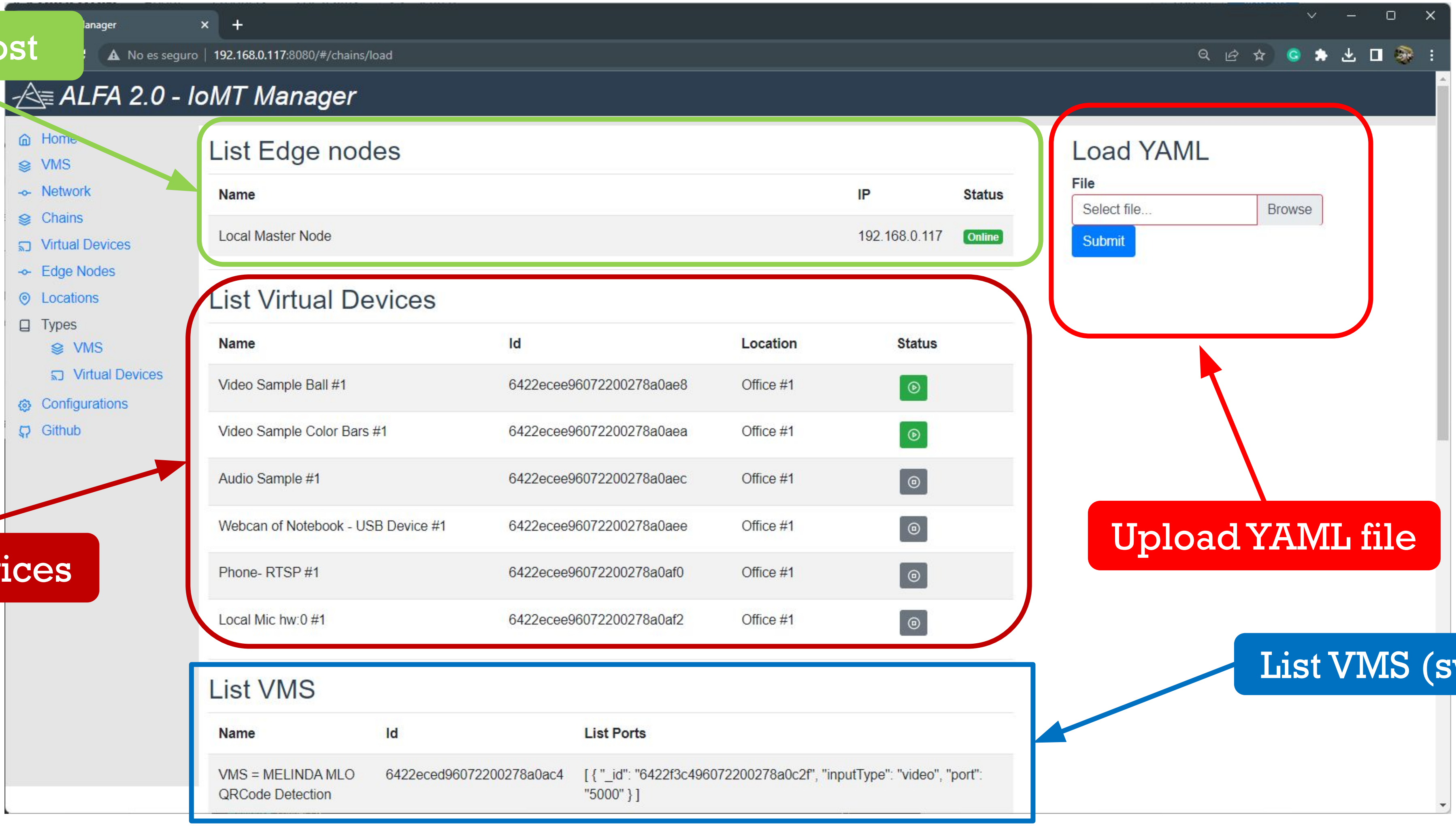
L-PRISM - EXAMPLE





```
- #virtualLink 5
  source:
    sourceName: Ball #Name of source connect
    sourceType: device #Type of source
    outputType: video # output media stream ty
    formatType: #optional format tipe of midia
  destination:
    destinationName: VMS1 # Destination name
    destinationIp: # Configure for orchestater
    destinationPort:
        numPort: 15001 #information of VMS
        typeStream: video # input media stream
- #virtualLink 6
  source:
    sourceName: VMS1 #Name of source connect
    sourceType: vms #Type of source
    outputType: video # output media stream ty
    formatType: #optional format tipe of midia
  destination:
    destinationName: 192.168.0.102 # Destinati
    destinationIp: 192.168.0.102 #IP of the fi
    destinationPort:
        numPort: 10004 #information of VMS
        typeStream: video # input media stream
```

ALFA 2.0



List Edge nodes

Name	IP	Status
Local Master Node	192.168.0.117	Online

List Virtual Devices

Name	Id	Location	Status
Video Sample Ball #1	6422ecee96072200278a0ae8	Office #1	▶
Video Sample Color Bars #1	6422ecee96072200278a0aea	Office #1	▶
Audio Sample #1	6422ecee96072200278a0aec	Office #1	◎
Webcan of Notebook - USB Device #1	6422ecee96072200278a0aee	Office #1	◎
Phone- RTSP #1	6422ecee96072200278a0af0	Office #1	◎
Local Mic hw:0 #1	6422ecee96072200278a0af2	Office #1	◎

List VMS

Name	Id	List Ports
VMS = MELINDA MLO QRCode Detection	6422eced96072200278a0ac4	[{ "_id": "6422f3c496072200278a0c2f", "inputType": "video", "port": "5000" }]

List of host

List of Devices

Upload YAML file

List VMS (swImage)

EVALUATION - GOALS

Goals of the experiment

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 subjects.	**Usability**

EVALUATION – G1

Questions and Metric for the goal *G1*

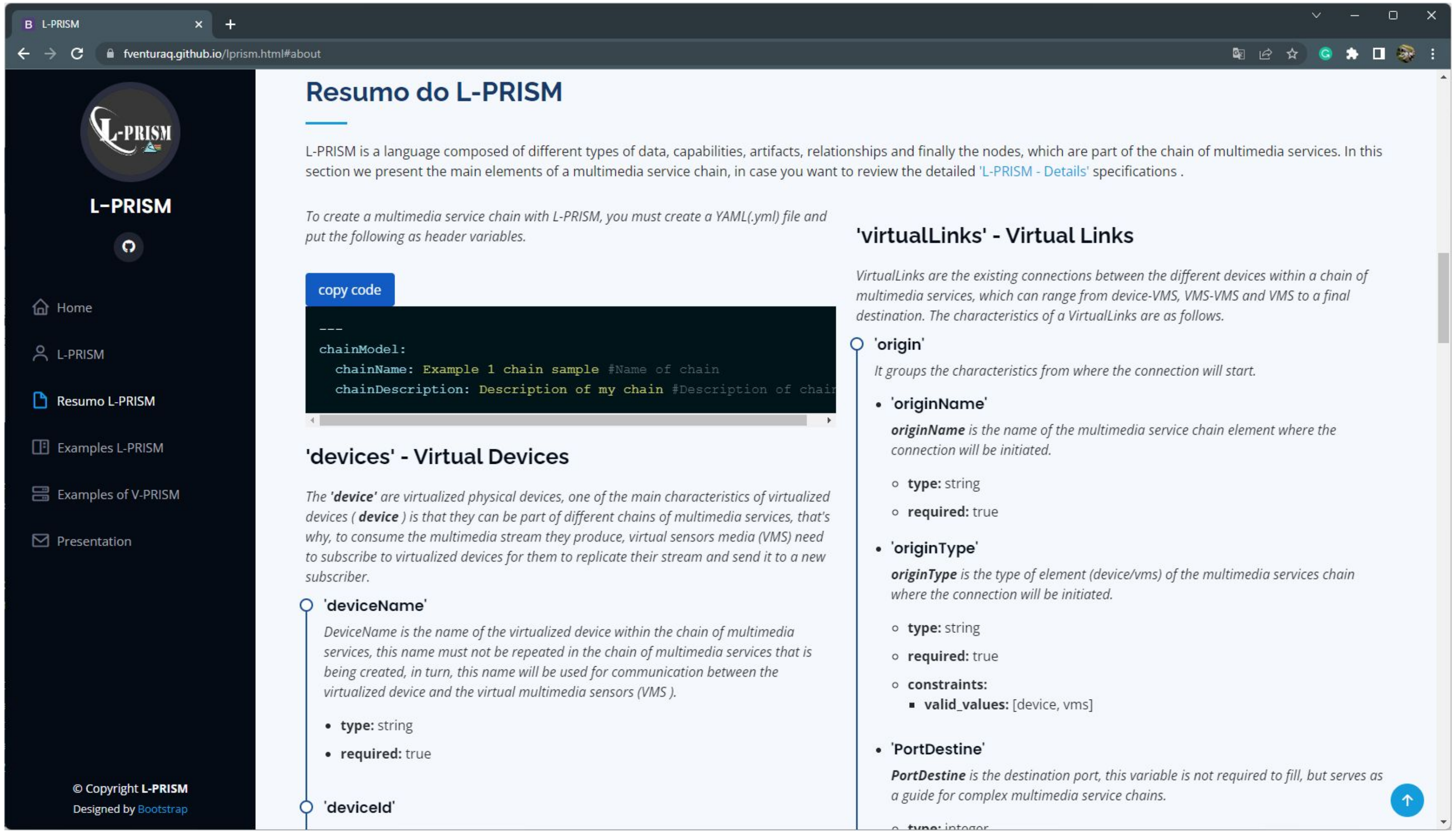
Quest.	Description	Metric (TAM)
Q1	Is the application engineering process using L-PRISM effective in terms of time for developing multimedia SFC based on multimedia VNF, compared to the traditional approach (V-PRISM)?	M1 - Development effort
Q2	Does the developer claim that using L-PRISM makes it easier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

EVALUATION – G2

Questions and Metric for the goal *G2*

Quest.	Description	Metric (CDN)
Q1	How easy is it to visualize or find the various components of L-PRISM while creating or changing a multimedia application?	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	In general, do the elements and attributes of L-PRISM represent well a multimedia SFC?	M4 - Closeness of Mapping
Q5	Is it easy to understand the data types and structures in L-PRISM?	M5 - Role Expressiveness
Q6	There are structures and data types in L-PRISM that can be closely related, and changes to one can affect the other. Are those dependencies visible?	M6 - Hidden dependencies
Q7	Does L-PRISM generally seem easy or difficult understand (for example, when changing different elements of a multimedia SFC)?	M7 - Hard mental operations

EVALUATION - EXPERIMENT (TRAINING)



Resumo do L-PRISM

L-PRISM is a language composed of different types of data, capabilities, artifacts, relationships and finally the nodes, which are part of the chain of multimedia services. In this section we present the main elements of a multimedia service chain, in case you want to review the detailed 'L-PRISM - Details' specifications .

To create a multimedia service chain with L-PRISM, you must create a YAML(.yml) file and put the following as header variables.

```
---
chainModel:
  chainName: Example 1 chain sample #Name of chain
  chainDescription: Description of my chain #Description of chai
```

'devices' - Virtual Devices

*The **'device'** are virtualized physical devices, one of the main characteristics of virtualized devices (**device**) is that they can be part of different chains of multimedia services, that's why, to consume the multimedia stream they produce, virtual sensors media (VMS) need to subscribe to virtualized devices for them to replicate their stream and send it to a new subscriber.*

'deviceName'

DeviceName is the name of the virtualized device within the chain of multimedia services, this name must not be repeated in the chain of multimedia services that is being created, in turn, this name will be used for communication between the virtualized device and the virtual multimedia sensors (VMS).

- **type:** string
- **required:** true

'deviceId'

'virtualLinks' - Virtual Links

VirtualLinks are the existing connections between the different devices within a chain of multimedia services, which can range from device-VMS, VMS-VMS and VMS to a final destination. The characteristics of a VirtualLinks are as follows.

'origin'

It groups the characteristics from where the connection will start.

- 'originName'

  ***originName** is the name of the multimedia service chain element where the connection will be initiated.*

  - **type:** string
  - **required:** true
- 'originType'

  ***originType** is the type of element (device/vms) of the multimedia services chain where the connection will be initiated.*

  - **type:** string
  - **required:** true
  - **constraints:**
    - **valid_values:** [device, vms]
- 'PortDestine'

  ***PortDestine** is the destination port, this variable is not required to fill, but serves as a guide for complex multimedia service chains.*

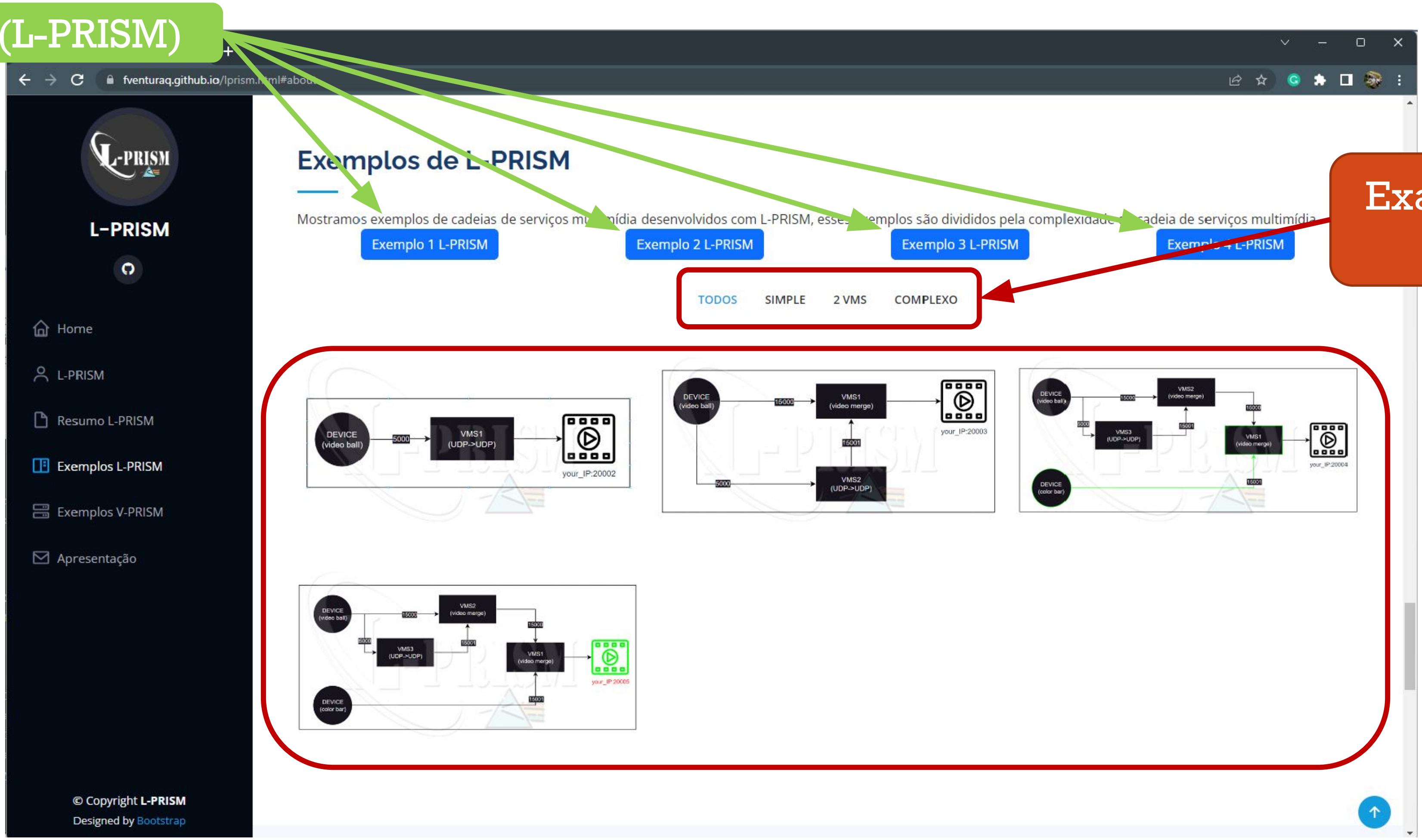
  - **type:** integer

https://fventuraq.github.io/lprism.html

EVALUATION - EXPERIMENT (TRAINING)

List examples (L-PRISM)

Example for type (L-PRISM)



fventuraq.github.io/lprism.html#about

L-PRISM

- Home
- L-PRISM
- Resumo L-PRISM
- Exemplos L-PRISM
- Exemplos V-PRISM
- Apresentação

© Copyright **L-PRISM**
Designed by Bootstrap

Exemplos de L-PRISM

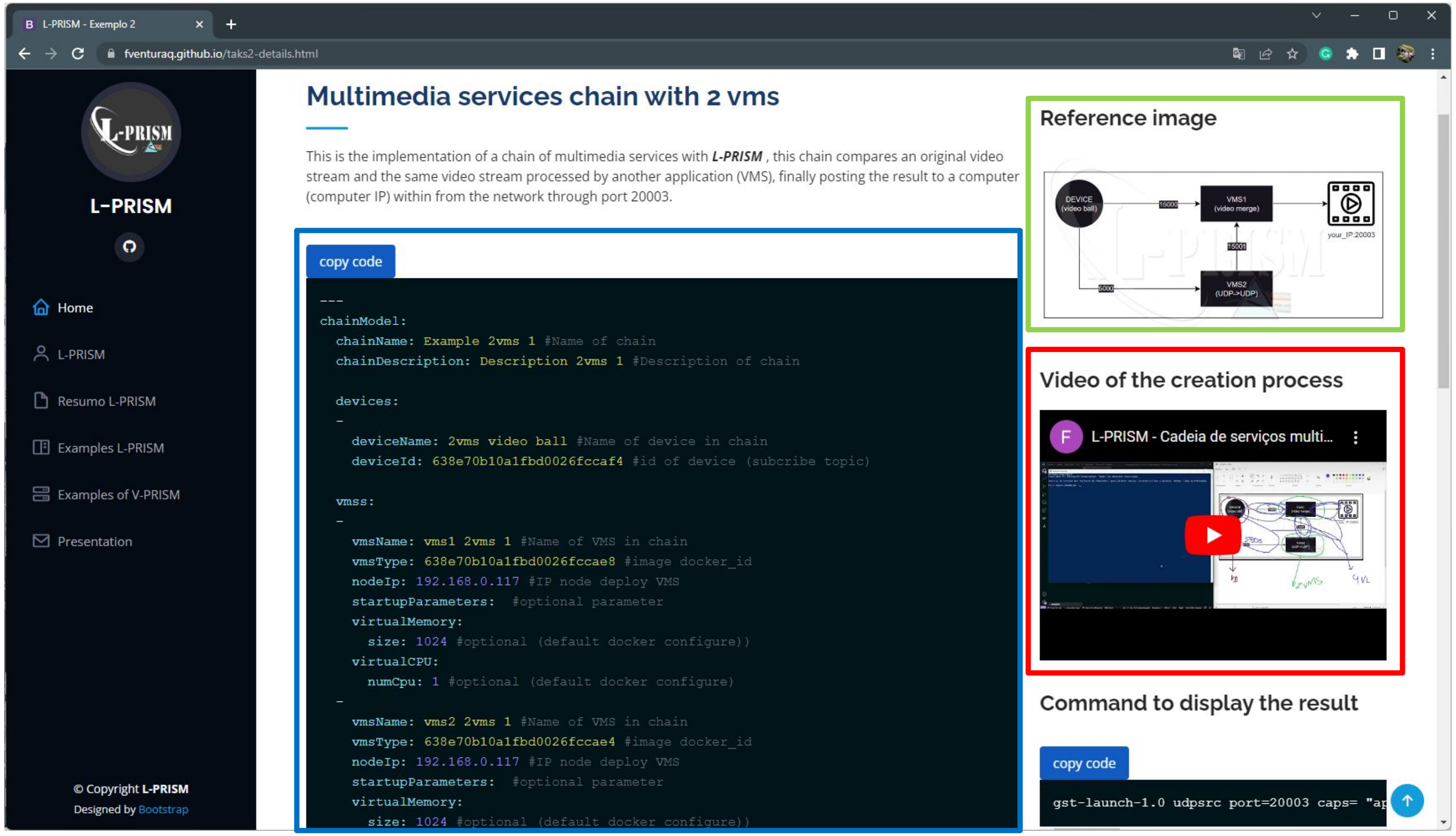
Mostramos exemplos de cadeias de serviços multimídia desenvolvidos com L-PRISM, esses exemplos são divididos pela complexidade da cadeia de serviços multimídia

Exemplo 1 L-PRISM | Exemplo 2 L-PRISM | Exemplo 3 L-PRISM | Exemplo 4 L-PRISM

TODOS | SIMPLE | 2 VMS | COMPLEXO

https://fventuraq.github.io/lprism.html

EVALUATION - EXPERIMENT (TRAINING)



Multimedia services chain with 2 vms

This is the implementation of a chain of multimedia services with ***L-PRISM*** , this chain compares an original video stream and the same video stream processed by another application (VMS), finally posting the result to a computer (computer IP) within from the network through port 20003.

```
---
chainModel:
  chainName: Example 2vms 1 #Name of chain
  chainDescription: Description 2vms 1 #Description of chain

  devices:
  -
    deviceName: 2vms video ball #Name of device in chain
    deviceId: 638e70b10a1fbd0026fccaf4 #id of device (subcribe topic)

  vmss:
  -
    vmsName: vms1 2vms 1 #Name of VMS in chain
    vmsType: 638e70b10a1fbd0026fccae8 #image docker_id
    nodeIp: 192.168.0.117 #IP node deploy VMS
    startupParameters:  #optional parameter
    virtualMemory:
      size: 1024 #optional (default docker configure))
    virtualCPU:
      numCpu: 1 #optional (default docker configure)
  -
    vmsName: vms2 2vms 1 #Name of VMS in chain
    vmsType: 638e70b10a1fbd0026fccae4 #image docker_id
    nodeIp: 192.168.0.117 #IP node deploy VMS
    startupParameters:  #optional parameter
    virtualMemory:
      size: 1024 #optional (default docker configure))
```

Reference image

Video of the creation process

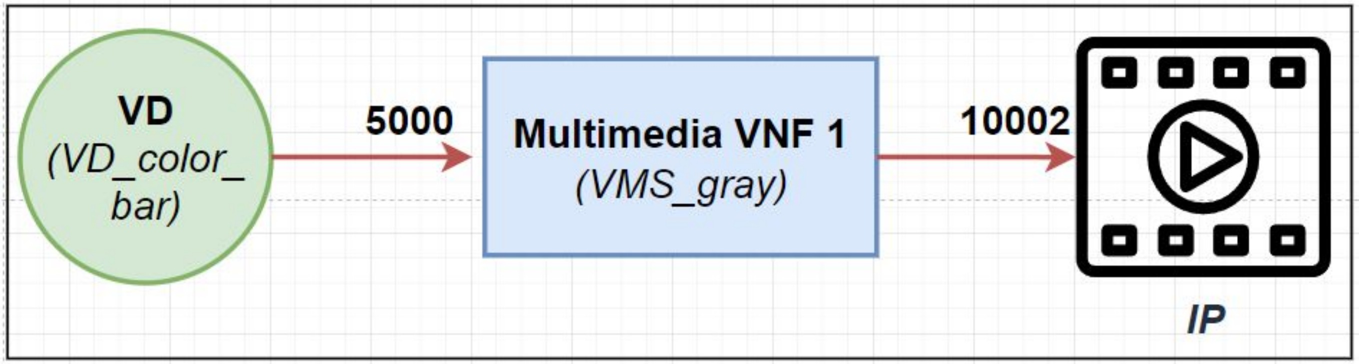
Command to display the result

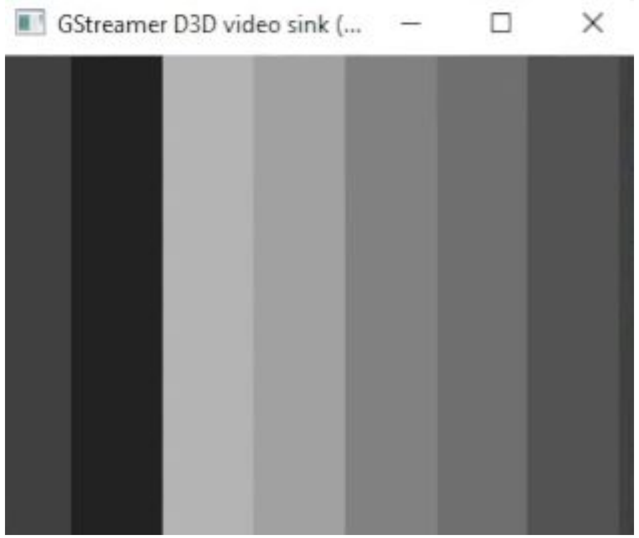
```
gst-launch-1.0 udpsrc port=20003 caps= "ap
```

https://fventuraq.github.io/lprism.html

EVALUATION - EXPERIMENT (EXECUTION)

Task 1: Create a chain of multimedia services that receive a multimedia stream (color video), transform the video into grayscale and finally publish the result to a computer within the network on port 10002.

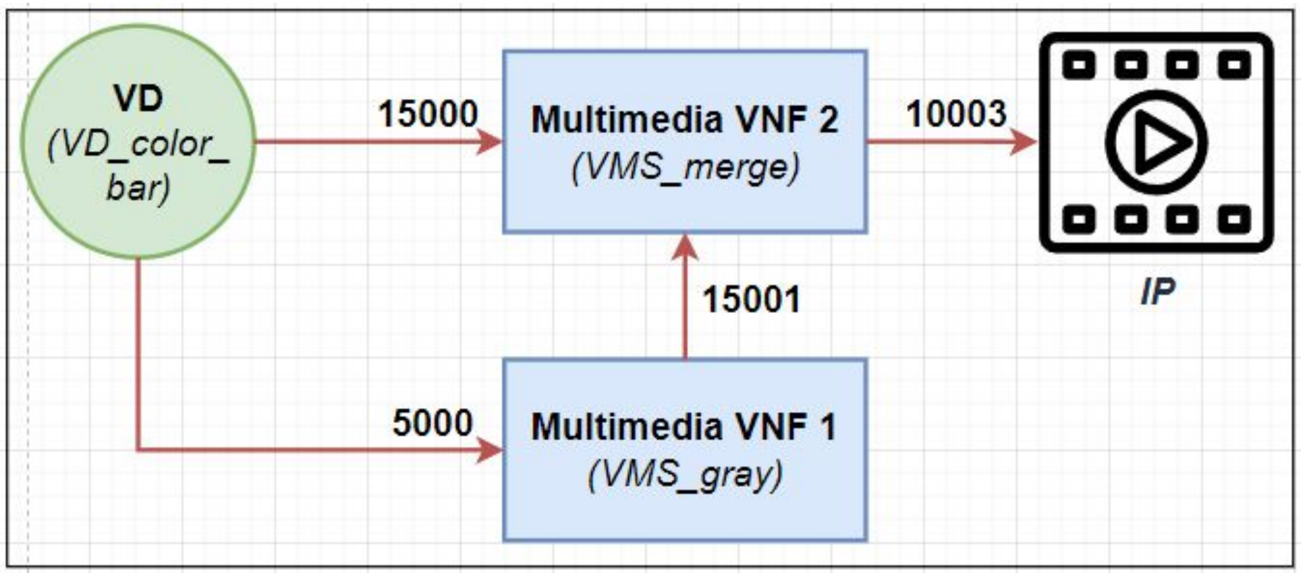


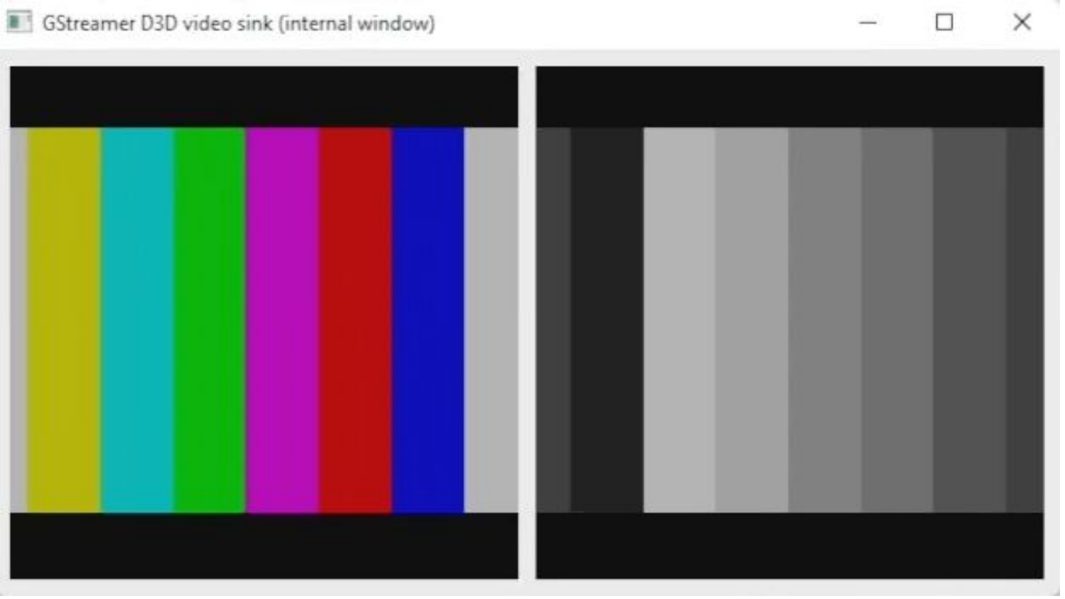


https://fventuraq.github.io

EVALUATION - EXPERIMENT (EXECUTION)

Task 2: Create a chain of multimedia services that compare an original video stream and the same greyscale-transformed video stream, finally publish the result to a computer within the network on port 10003.

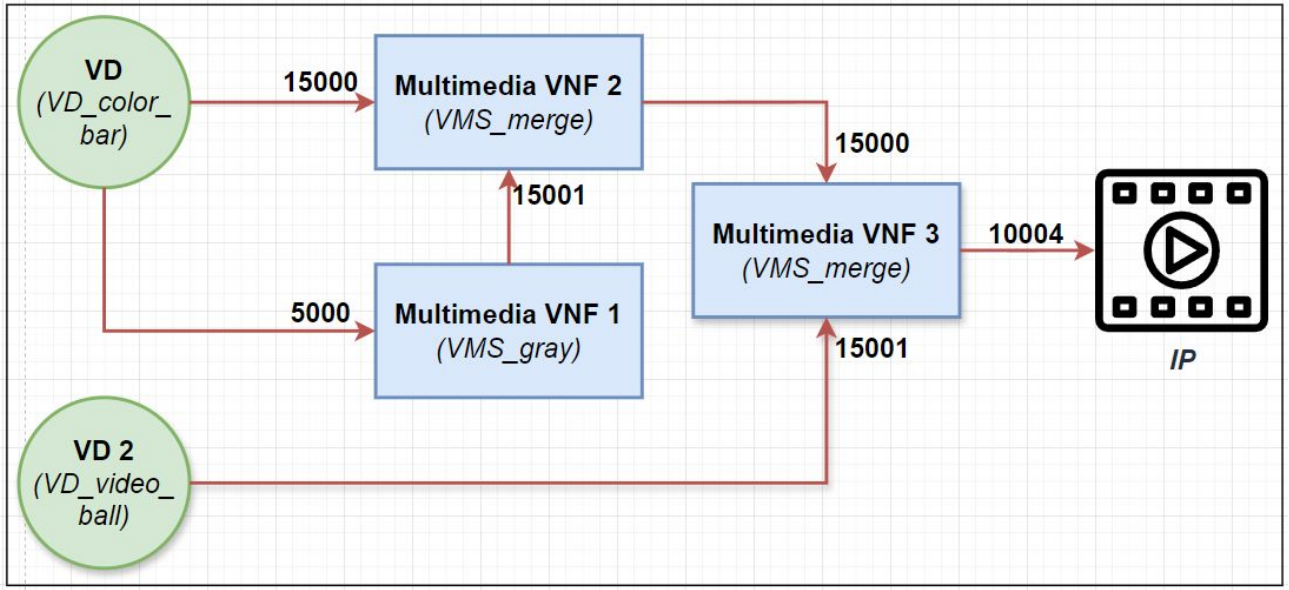


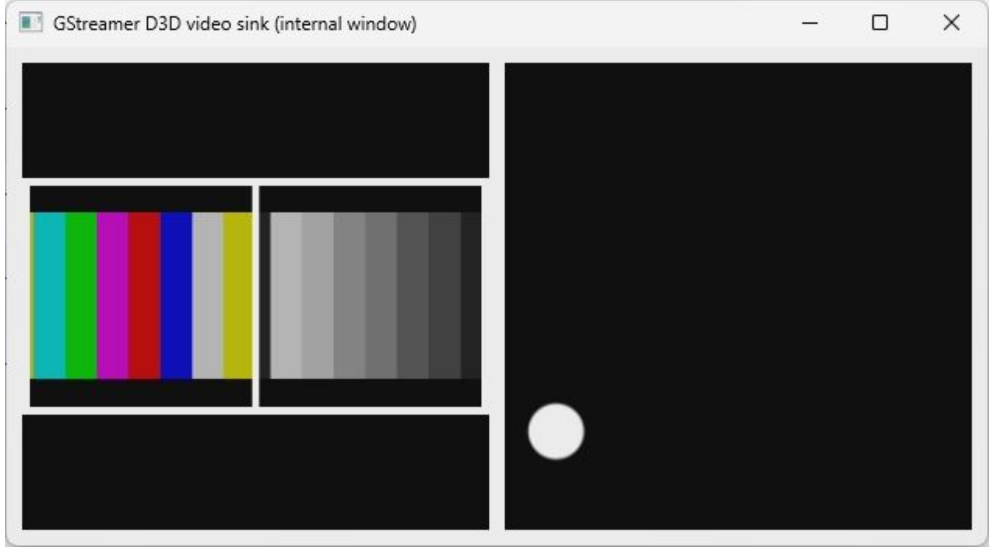


https://fventuraq.github.io

EVALUATION - EXPERIMENT (EXECUTION)

Task 3: Add a different video stream (video ball) to ***task 2***, group it with the result of task 2 and finally publish the result to a computer within the network on port 10004..

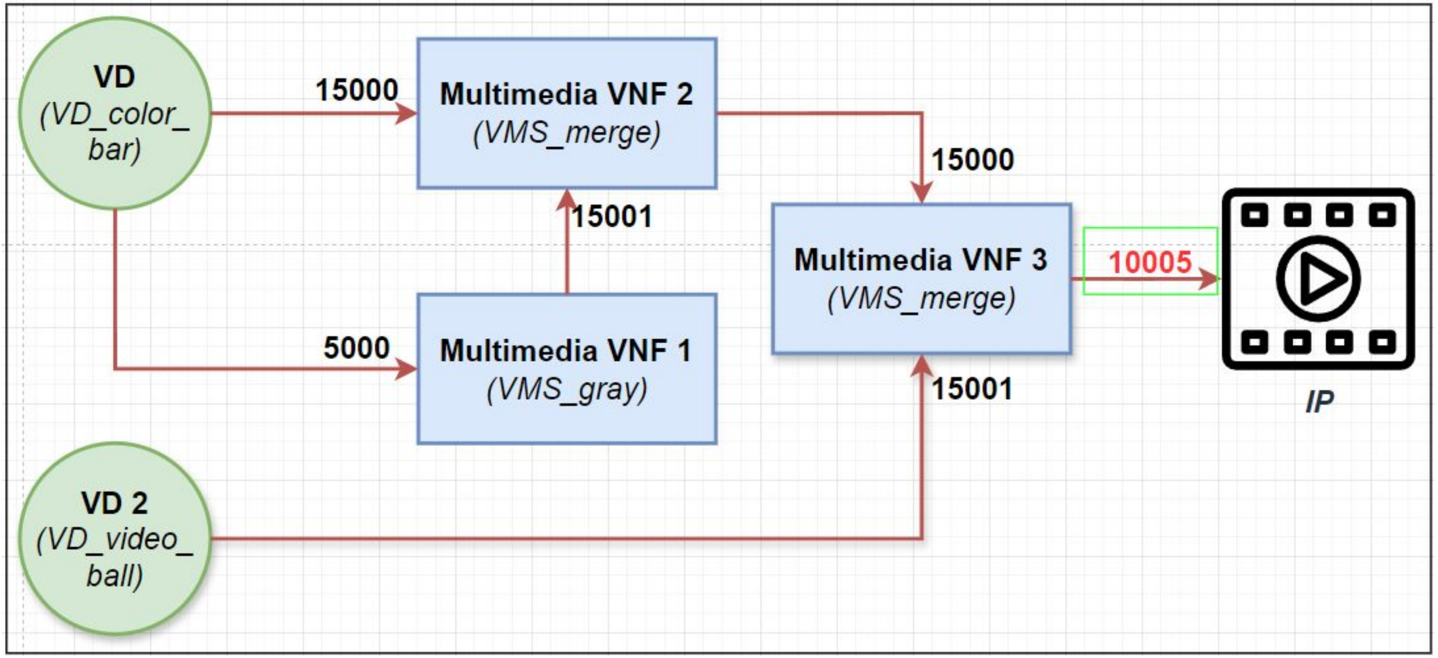


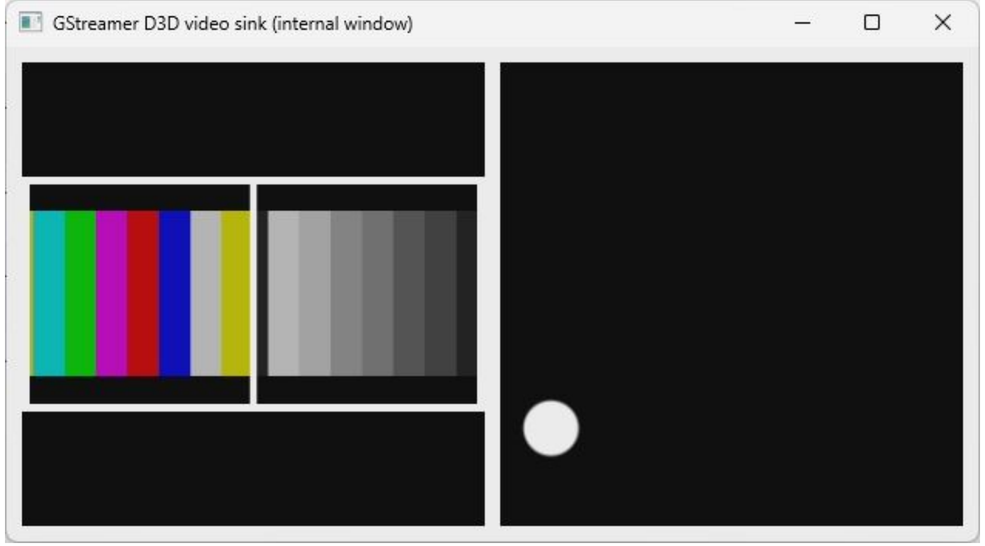


https://fventuraq.github.io

EVALUATION - EXPERIMENT (EXECUTION)

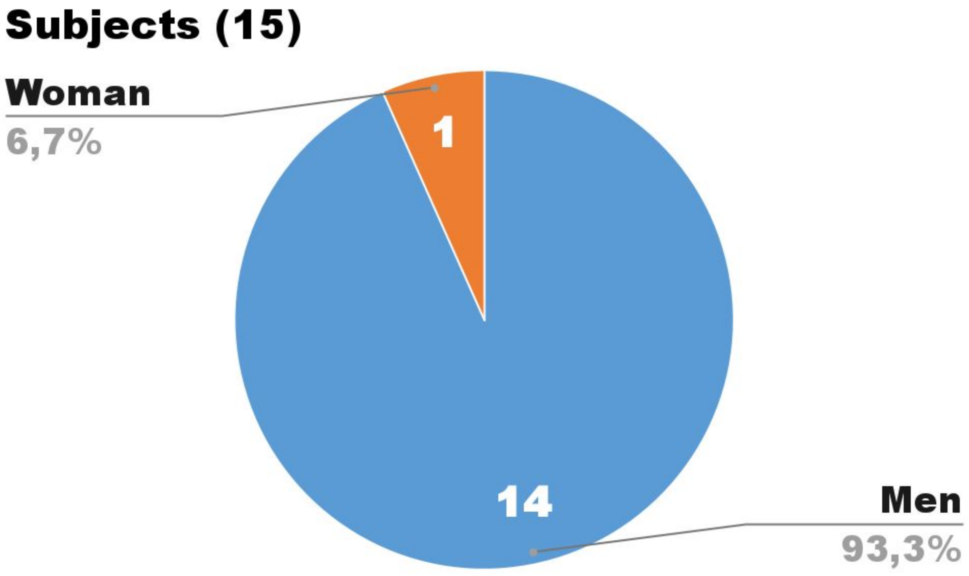
Task 4: Replicate ***task*** 3 and post the result on a computer inside the network through port 10005.

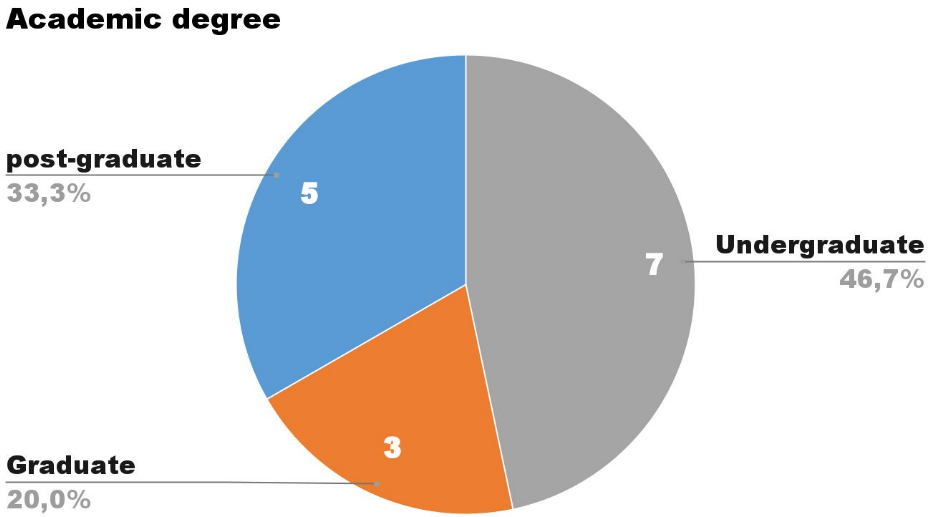


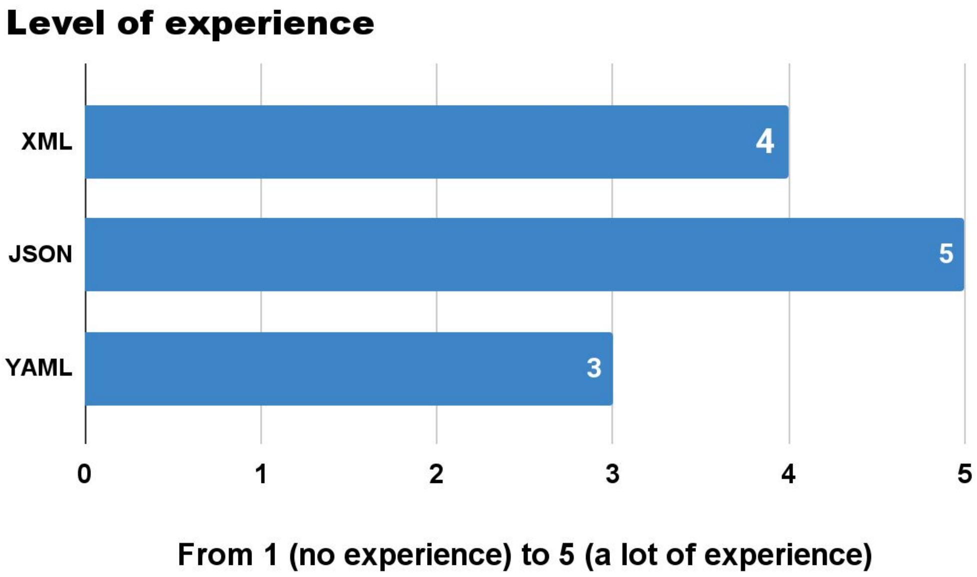


https://fventuraq.github.io

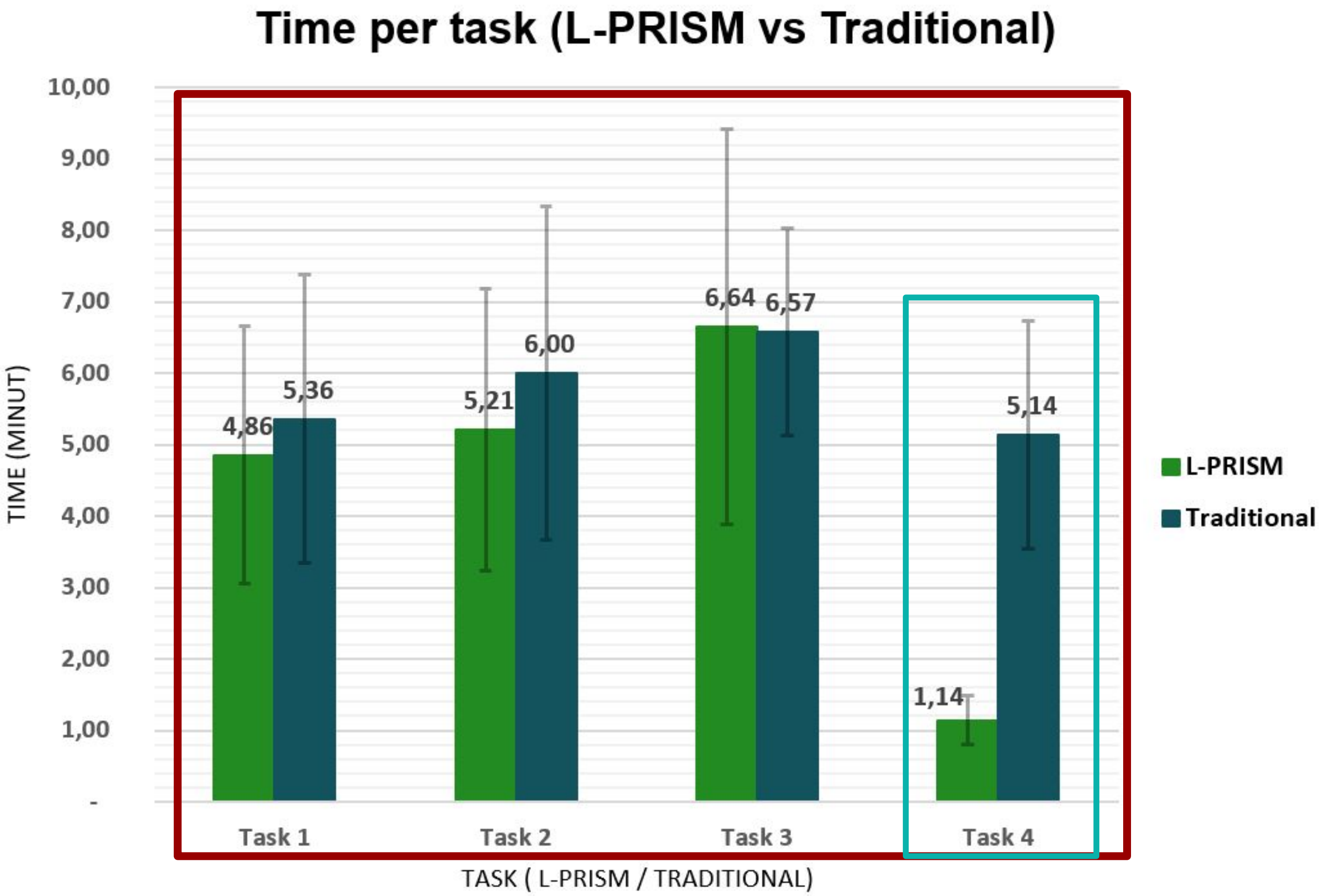
EVALUATION - SUBJECTS







EVALUATION - RESULT OF G1

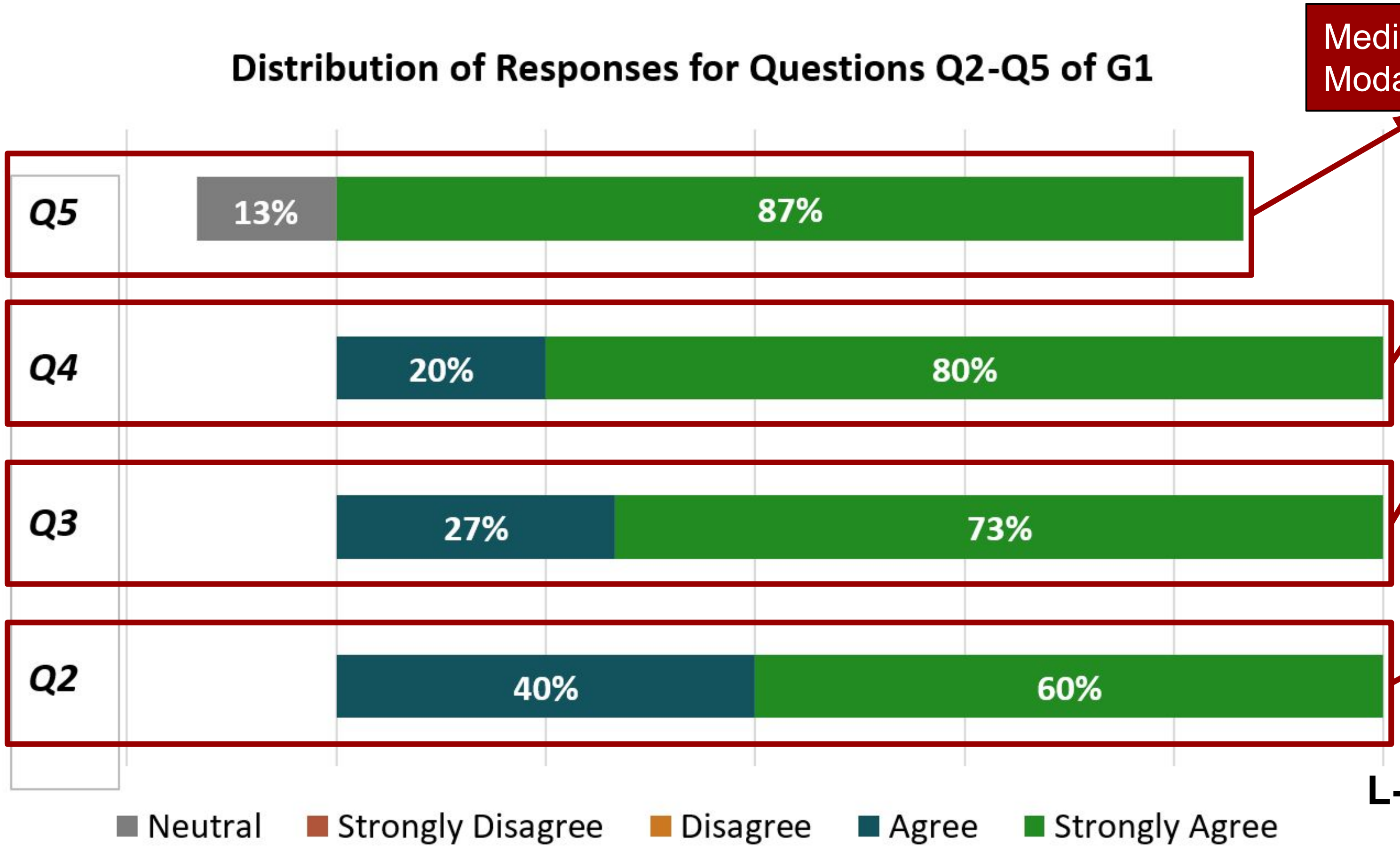


M1 - Development effort
L-PRISM: 17.85 min
Traditional: 23.07 min

M6 - Reuse effort
L-PRISM: 1.14 min
Traditional: 5.14 min

*So we can conclude that L-PRISM is more **productive** the more it is used.*

EVALUATION - RESULT OF G1



Distribution of Responses for Questions Q2-Q5 of G1

Median = 5
Moda = 5

M5 - Perceived reuse

Median = 5
Moda = 5

M4 - Perceived utility

Median = 5
Moda = 5

M3 - Perceived ease of use

Median = 5
Moda = 5

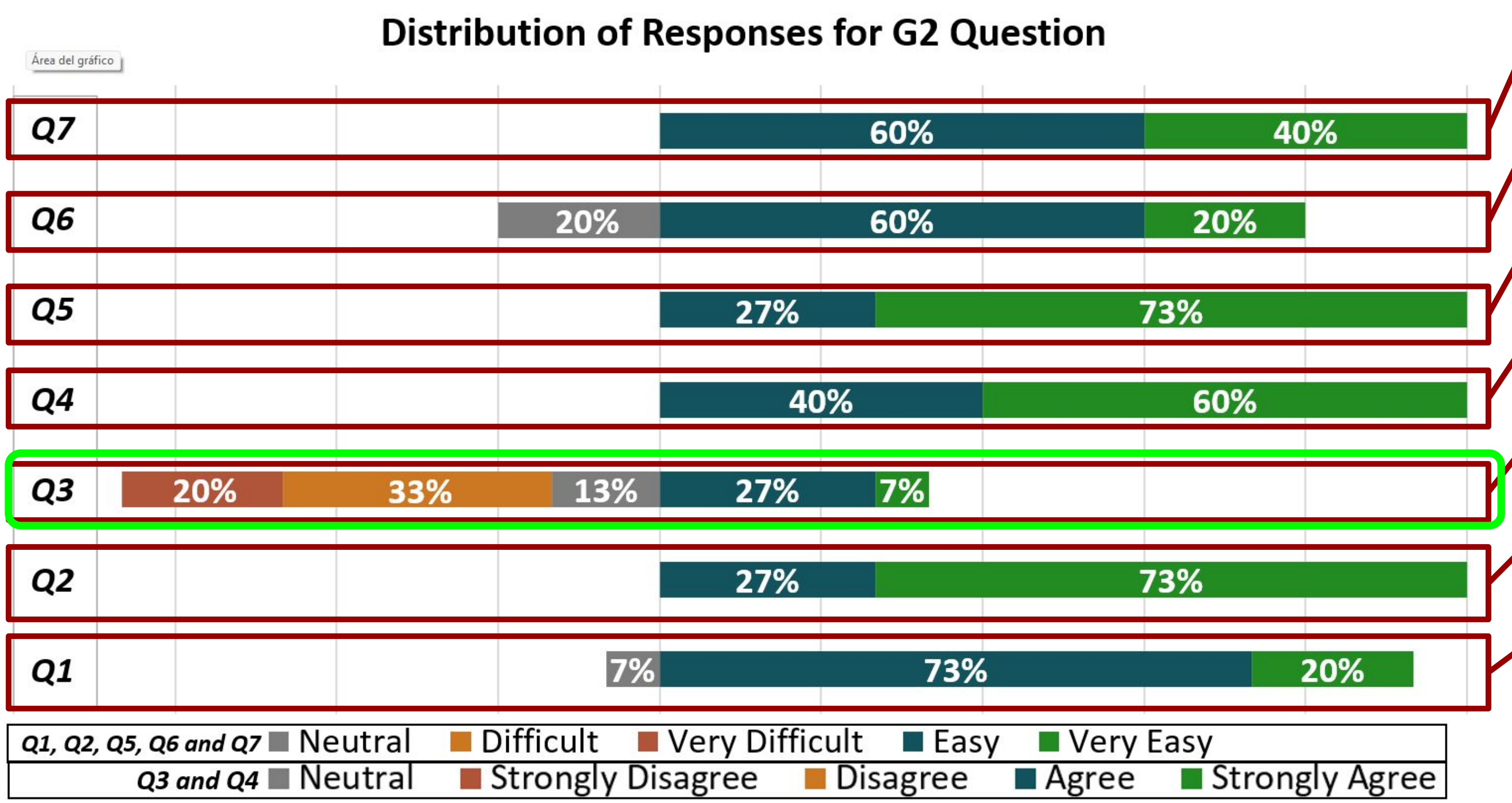
M2 - Understanding of requirements.

Median Q2 to Q5 of G1 = 5

L-PRISM is excellent from the point of view of ***efficiency***

Then it is possible to conclude that the goal G1 was achieved.

EVALUATION - RESULT OF G2



Median = 4
Moda = 4
M7 - Hard mental operations

Median = 4
Moda = 4
M6 - Hidden dependencies

Median = 5
Moda = 5
M5 - Role Expressiveness

Median = 5
Moda = 5
M4 - Closeness of Mapping

Median = 3
Moda = 2
M3 - Diffuseness

Median = 5
Moda = 5
M2 - Viscosity

Median = 4
Moda = 4
M1 - Visibility

Median (+) G2 = 5

L-PRISM is good from the point of view of ***usability***

Then it is possible to conclude that the goal G2 was achieved.

CONCLUSION

- In summary, L-PRISM, our Domain Specific Language (DSL), has been shown to be highly **efficient**, **productive**, and **easy to use** for creating Multimedia SFC based on multimedia VNF. These features make it a valuable tool for developing multimedia services based on virtualization.

- 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.

FUTURE WORK

- Develop a framework that facilitates the import of SFCs designed with L-PRISM, offering an intuitive **graphical interface** to visualize and modify their **topology**.
- Propose the integration of resource **allocation** and **scaling algorithms** into *ALFA 2.0*.

Thank you so much.

Muito obrigado.

Muchas gracias.

https://github.com/fventuraq/alfa

https://fventuraq.github.io/lprism

fventuraq@midiacom.uff.br