Segment Routing for Chaining Micro-Services at Different Programmable Network Levels

Bertrand Mathieu, Olivier Dugeon, **Joël Roman Ky**

Orange Innovation

Philippe Graff, Thibault Cholez

Université de Lorraine, Inria

14/03/2024

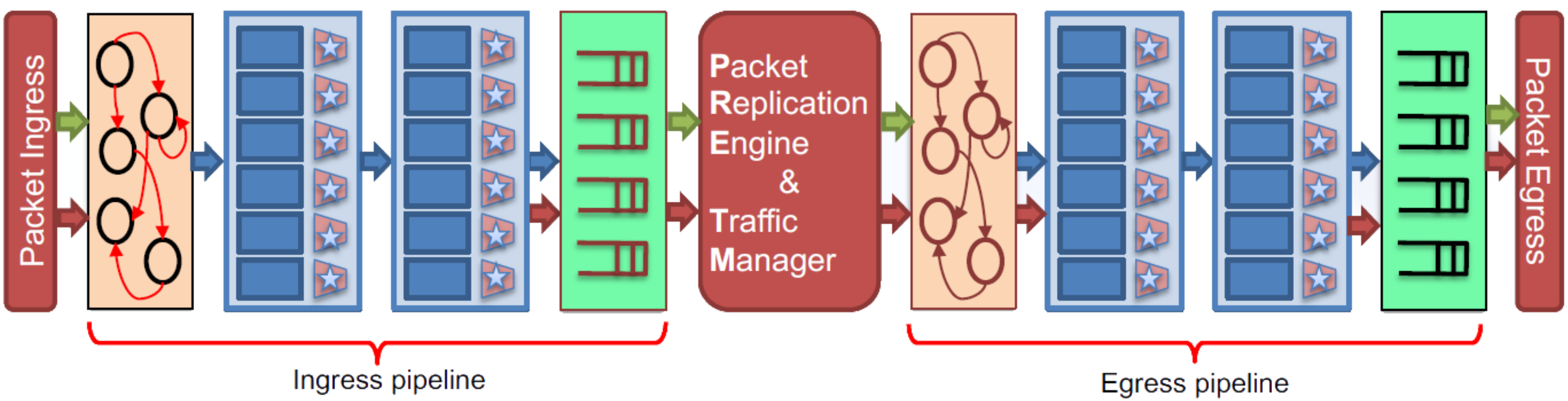
Context and problem statement

- Global end-to-end services can be offered to end-users by chaining several micro-services, eventually developed by different providers
- Network programmability has become increasingly important in network architectures, first with NFV (Network Function Virtualization) and later with P4 (Programming Protocol-independent Packet Processors)
- However, NFV micro-services can only chained with other NFV micro-services and currently no chaining for P4 modules

=> Need for a multi-level and multi-technology chaining of micro-services

P4 : Brief concepts

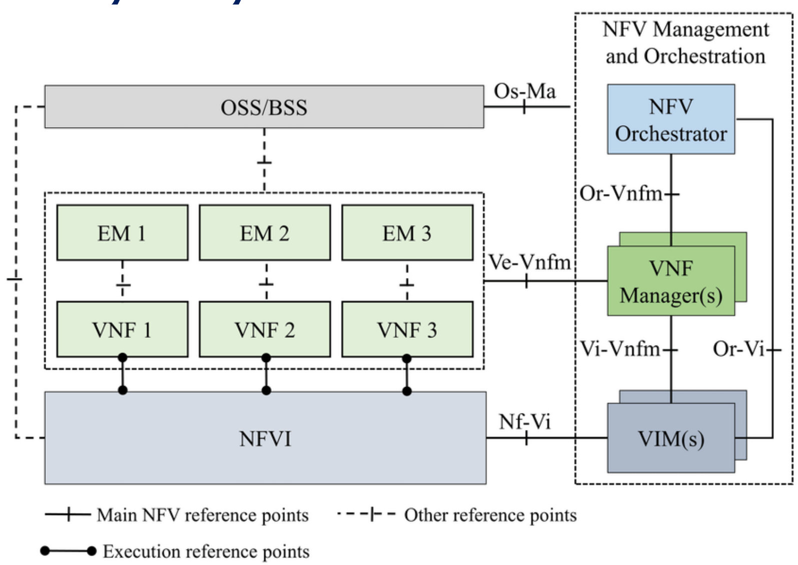
- Open-source consortium (p4.org)
- Data plane programmability : deploy software into P4-based equipment (e.g., switch)
- Parsing of packet information
- Packet processing based on match-action tables, remotely configurable by interfaces (controllers)
- P4 hardware chipset (e.g. Intel/Tofino, AMD/Pensando)



From https://events19.linuxfoundation.org/wp-content/uploads/2017/12 Tutorial-P4-and-P4Runtime-Technical-Introduction-and-Use-Cases-for-Service-Providers-Carmelo-Cascone-Open-Networking-Foundation.pdf

NFV : Brief concepts

- Defined at ETSI
- Services, aka VNF (Virtual Network Functions), hosted on standard virtualized infrastructure, in order to ease their lifecyle (install, configure, remove, etc.)
- Remove the necessity of dedicated hardware, provided by vendors
- Operators only buy the micro-services to the vendors



From https://www.researchgate.net/figure/MANO-architecture-overview_fig2_342571996

Why a 2-levels programmability ?

- NFV is available for offering network services (VNFs):
  - Mostly for control plane functions
  - Running in standard machines
  - Services can be complex, with difficult processing tasks, as developed in common programming language.
    - But not suited for line-rate function, requiring very low latency

- P4 is emerging as a data plane programmable solution :
  - Processing packet per packet as defined by the P4 module, ensuring line-rate processing
  - Running in network hardware (e.g., switch)
    - But P4 services can not be complex and include tricky processing tasks

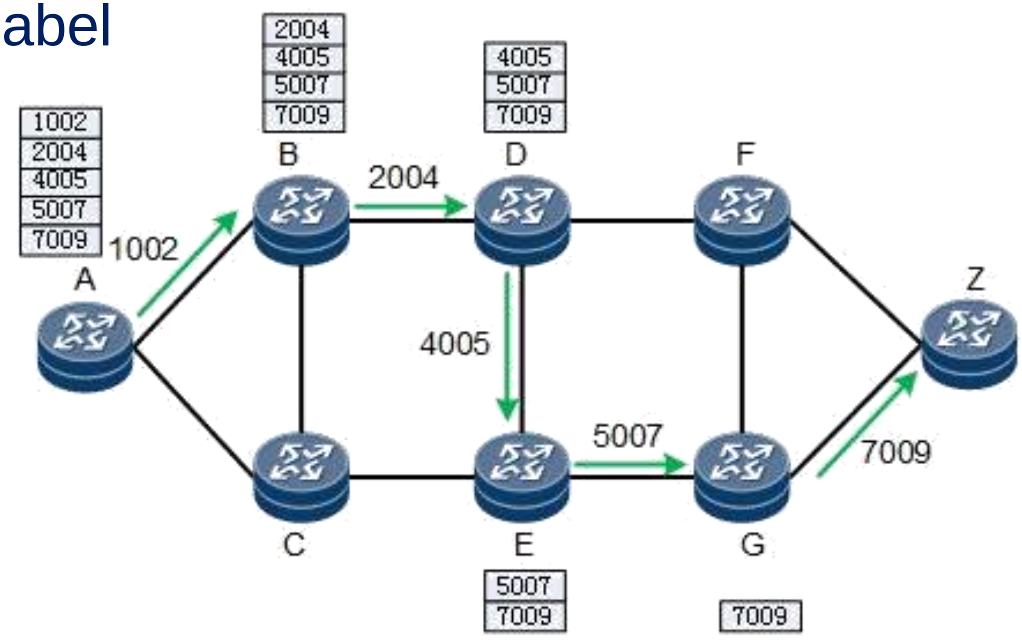
Why a 2-levels programmability ?

- NFV and P4 exhibits their own advantages and limitations related to their environment :
  - execution time, resource consumption, computational task, protocol stack layer, configuration, migration, etc.
- Depending on the complexity and requirements of the micro-service, developers should consider to develop it either as a VNF or a P4 module

=> The global end-to-end service could benefit of the best of the two levels if chaining micro-services at the 2 levels (NFV and P4) would be feasible

SR-MPLS Concept

- Segment Routing leverages the source routing paradigm, where a packet carries the path to reach its destination in its header
- SR-MPLS is mostly used in networks to route packets between nodes (i.e. IP routers).
- Segment Identifier (SID) identifies a node, a link or a service that is reachable in the network. For SR-MPLS, SID are represented by a MPLS label



From https://support.huawei.com/enterprise/en/doc/EDOC110027856/3ee10304/understanding-segment-routing-mpls

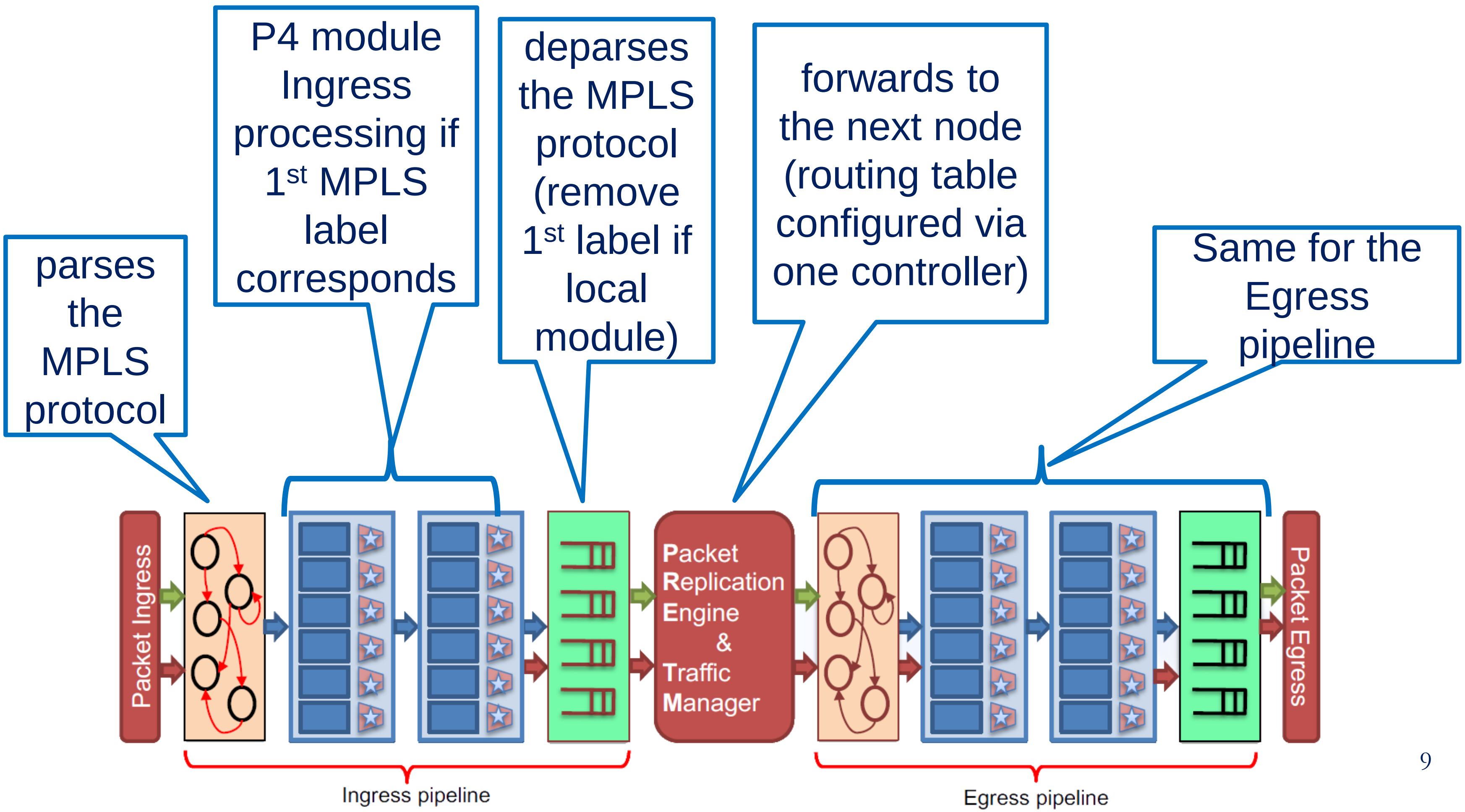
SR for chaining P4 & NFV services

- Proposal to use SR-MPLS to chain micro-services :

- Define a SID for each micro-service, being a P4 module or a VNF.
- Stack the SR-MPLS labels as the order to execute the micro-services to compose the global service
- Route the MLPS packet according to the MPLS label to reach to next micro-service to be executed for the given packet, either to the next P4 node or to the NFV node hosting the VNF

SR in P4-based network switch

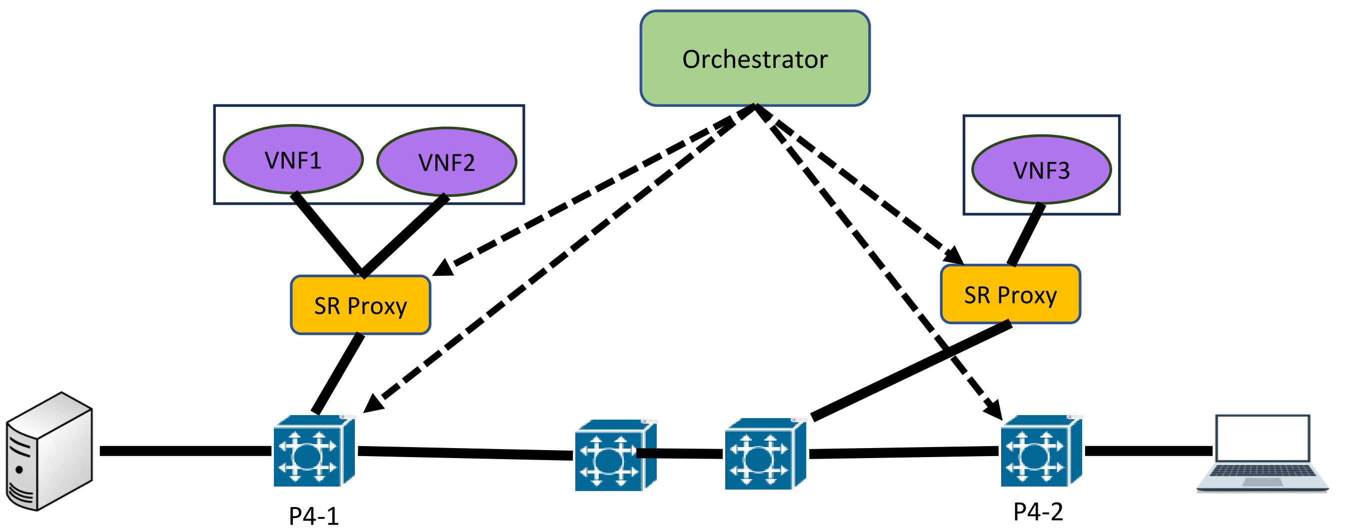
- MPLS being a network protocol, the P4 switch can easily manage it.



SR Proxy for VNFs

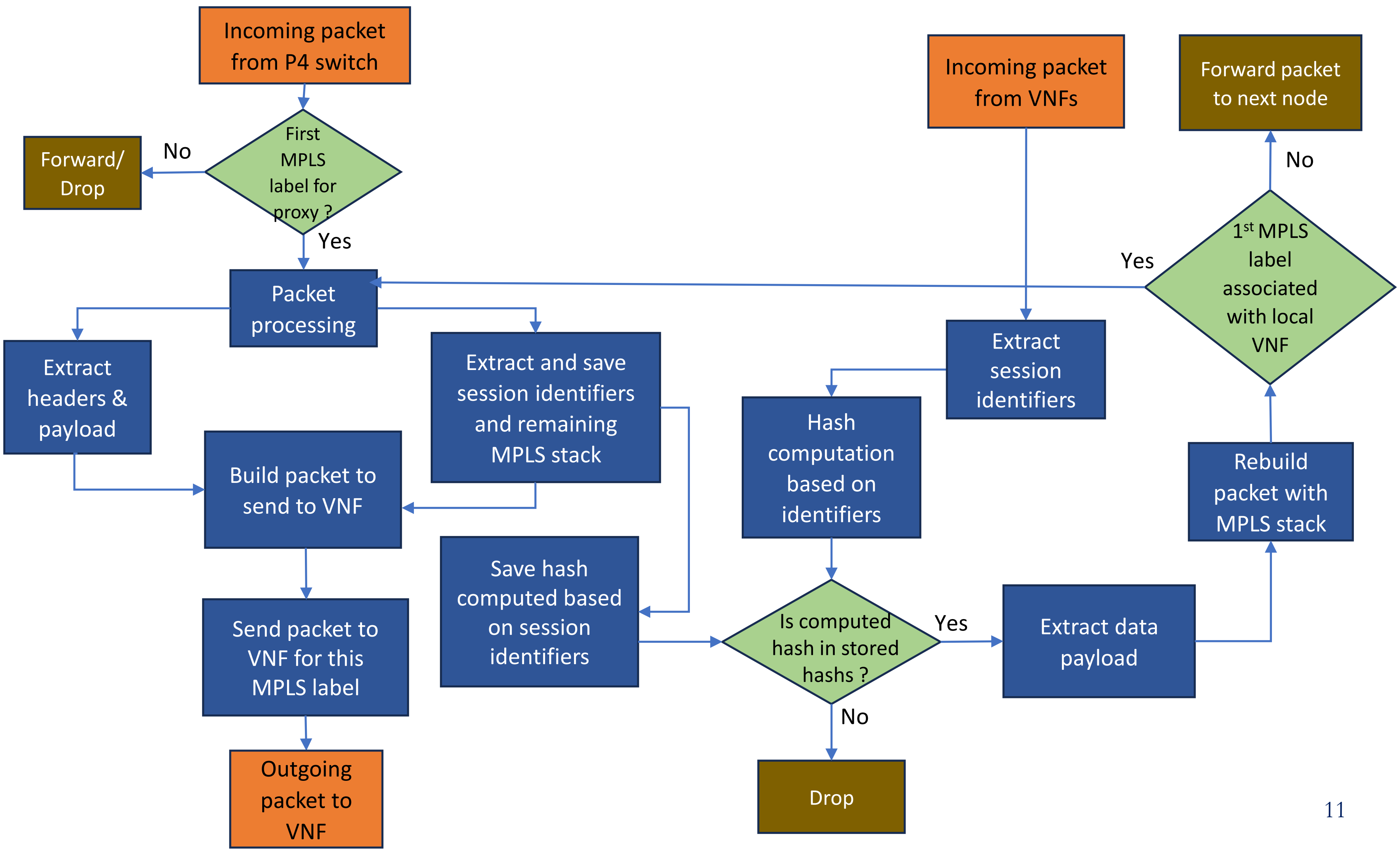
- But VNFs, running in computing machines, do not include MPLS in their protocol stack
- One option could be to integrate MPLS into the protocol stack but our objective being to provide an architecture requiring no modification of the current VNFs for a seamless integration, we

=> Define a proxy, intermediate between network MPLS-enabled nodes and VNFs to chain the micro-services



SR Proxy for VNFs

- The proxy has the following algorithm



Implementation & Demonstrator

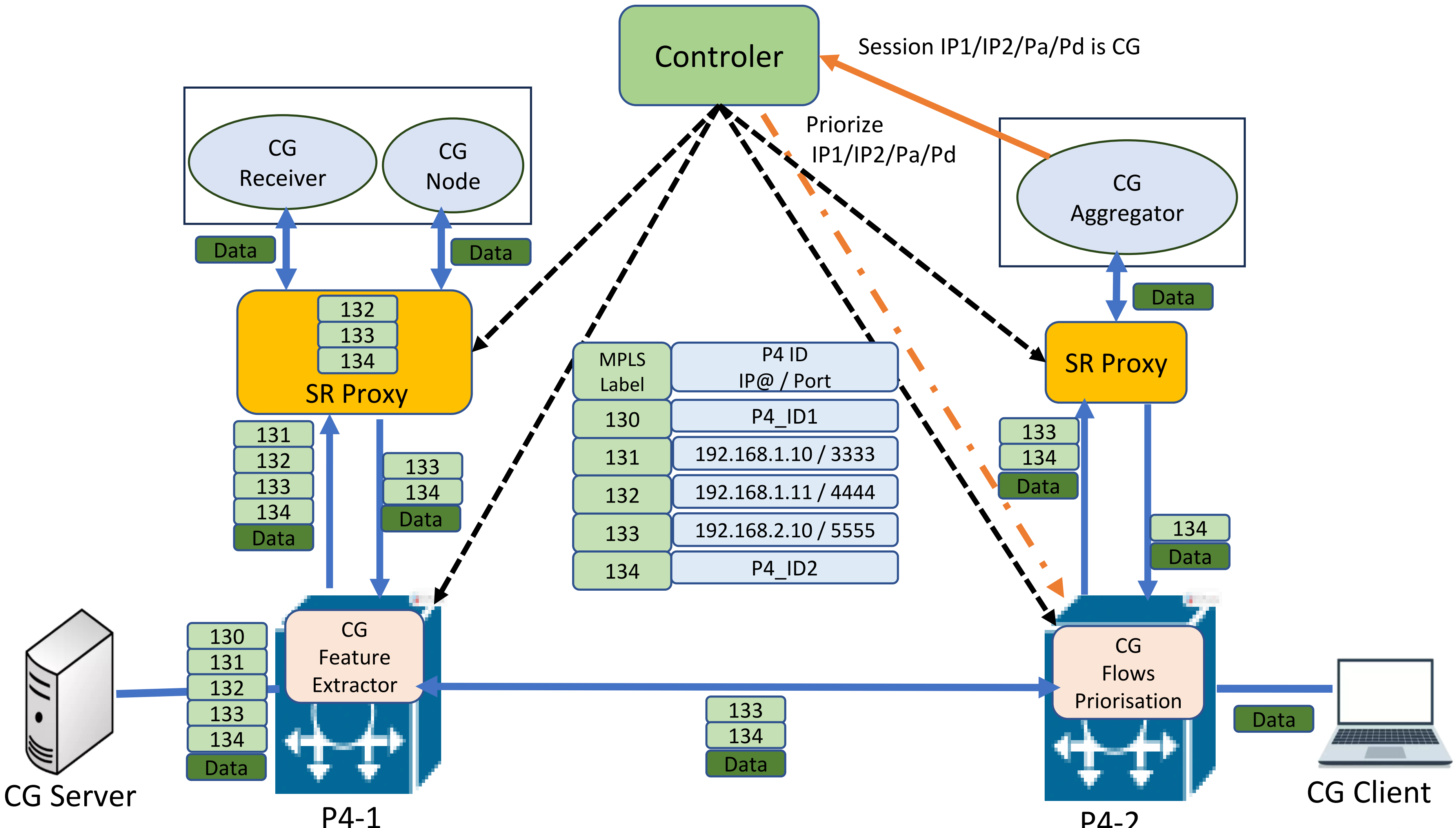
- Global service : Identify cloud gaming sessions and priorize them when detected for ensuring low-latency requirements

- 2 P4 modules :
  - 1 for session features computation (packet size, direction, inter-arrival time, etc.)
  - 1 for for traffic priorisation
- 3 VNFs
  - 1 module for receiving session data from the 1st P4 module and format them as needed
  - 1 or more modules for processing the data, with a AI module
  - 1 module for aggregating for the results and make a decision
- The SR-Proxy between the P4 nodes and VNFs
- A global controler to configure the routing tables based on the order of micro-services chaining

Implementation & Demonstrator

- For the P4 modules :
  - Running in a Edgecore DCS 800 Wedge 100BF-32X switch, having 32 QSFP28 ports supporting each 100 GbE, with a P4 Intel/Tofino chipset
  - Modules developed with the TNA (Tofino Native Architecture) SDE (Software Development Environment) v9.9.1
- For the VNF micro-services
  - Software running in common linux-based computing machines, developed in Python 3.7, with pandas, pytorch, scikit-learn, etc.
- For the SR-Proxy
  - Software running in common linux-based computing machines, developed in Python 3.8, with scapy, Hashlib and JSON
- For the controller
  - Software running in common linux-based computing machines, developed in Python 3.8

Implementation & Demonstrator



MPLS Label	P4 ID IP@ / Port
130	P4_ID1
131	192.168.1.10 / 3333
132	192.168.1.11 / 4444
133	192.168.2.10 / 5555
134	P4_ID2

Implementation & Demonstratorr

- Example of P4 traces for the SR-MPLS processing

- Example of sessions decision for 2 sessions

```
:08-11 17:15:33.969096:    :0xc:-:<0,0,->:------------ Stage 0 ------------
:08-11 17:15:33.969264:    :0xc:-:<0,0,0>:Ingress : Table SwitchIngress.mpls_label0_check is hit
:08-11 17:15:33.969277:        :0xc:-:<0,0,0>:Key:
:08-11 17:15:33.969292:        :0xc:-:<0,0,0>:  hdr.mpls$0.label[3:0] = 0x1
:08-11 17:15:33.969301:        :0xc:-:<0,0,0>:  hdr.mpls$0.label[19:4] = 0x0
:08-11 17:15:33.969312:        :0xc:-:<0,0,0>:Execute Action: SwitchIngress.packet_for_me
:08-11 17:15:33.969343:        :0xc:-:<0,0,0>:Action Results:
:08-11 17:15:33.969352:        :0xc:-:<0,0,0>:  ----- ModifyFieldPrimitive -----
:08-11 17:15:33.969360:        :0xc:-:<0,0,0>:  Operation:
:08-11 17:15:33.969368:        :0xc:-:<0,0,0>:  set
:08-11 17:15:33.969376:        :0xc:-:<0,0,0>:  Destination:
:08-11 17:15:33.969384:        :0xc:-:<0,0,0>:  hdr.udp.dst_port[15:0] = 0x63
:08-11 17:15:33.969392:        :0xc:-:<0,0,0>:  mask=0xFFFF
:08-11 17:15:33.969400:        :0xc:-:<0,0,0>:  Source 1:
:08-11 17:15:33.969408:        :0xc:-:<0,0,0>:  Val=0x10
:08-11 17:15:33.969415:        :0xc:-:<0,0,0>:  ----- RemoveHeaderPrimitive -----
:08-11 17:15:33.969423:        :0xc:-:<0,0,0>:  Destination:
:08-11 17:15:33.969431:        :0xc:-:<0,0,0>:  hdr.mpls$0.$valid=header
:08-11 17:15:33.969439:        :0xc:-:<0,0,0>:  ----- ModifyFieldPrimitive -----
:08-11 17:15:33.969447:        :0xc:-:<0,0,0>:  Operation:
:08-11 17:15:33.969455:        :0xc:-:<0,0,0>:  set
:08-11 17:15:33.969462:        :0xc:-:<0,0,0>:  Destination:
:08-11 17:15:33.969470:        :0xc:-:<0,0,0>:  ig_md.forme[3:0] = 0x2
:08-11 17:15:33.969478:        :0xc:-:<0,0,0>:  ig_md.forme[19:4] = 0x8
:08-11 17:15:33.969486:        :0xc:-:<0,0,0>:  mask=0xFFFFF
:08-11 17:15:33.969493:        :0xc:-:<0,0,0>:  Source 1:
:08-11 17:15:33.969501:        :0xc:-:<0,0,0>:  hdr.mpls$1.label[3:0] = 0x2
:08-11 17:15:33.969509:        :0xc:-:<0,0,0>:  hdr.mpls$1.label[19:4] = 0x8
:08-11 17:15:33.969523:        :0xc:-:<0,0,0>:Next Table = SwitchIngress.forward_packet
:08-11 17:15:33.969535:    :0xc:-:<0,0,->:------------ Stage 1 ------------
:08-11 17:15:33.969696:    :0xc:-:<0,0,1>:Ingress : Table SwitchIngress.forward_packet is hit
:08-11 17:15:33.969709:        :0xc:-:<0,0,1>:Key:
9722:        :0xc:-:<0,0,1>:  ig_md.forme[3:0] = 0x2
9730:        :0xc:-:<0,0,1>:  ig_md.forme[19:4] = 0x8
9757:        :0xc:-:<0,0,1>:Execute Action: SwitchIngress.set_outport
9774:        :0xc:-:<0,0,1>:Action Results:
9783:        :0xc:-:<0,0,1>:  ----- ModifyFieldPrimitive -----
9792:        :0xc:-:<0,0,1>:  Operation:
9800:        :0xc:-:<0,0,1>:  set
9808:        :0xc:-:<0,0,1>:  Destination:
9816:        :0xc:-:<0,0,1>:  ig_intr_md_for_tm.ucast_egress_port[8:0] = 0x1
9824:        :0xc:-:<0,0,1>:  mask=0x1FF
9846:        :0xc:-:<0,0,1>:  Source 1:
9871:        :0xc:-:<0,0,1>:  port=action_param
9879:        :0xc:-:<0,0,1>:Next Table = --END_OF_PIPELINE--
```

```
[!] AGG : Lasts ~ 34.385243 seconds
      > 1684961410336926231 : 886 CG vs 114 NCG -> 0.886000 CG
      > 100.110.120.130 <-> 20.21.22.23

      > 15106734211528108567 : 0 CG vs 1000 NCG -> 0.000000 CG
      > 90.11.12.13 <-> 91.21.22.23


[!] AGG: Mean Time by classifier : 0.001770
[!] AGG: Nb conversations : 2, ~ 1000.000000 reports per conversation
[!] AGG: received 2000 classifs
[!] AGG: ~58 classifs per second
```

Conclusion & Evolution

- Proposal to connect the two levels of programmability (NFV & P4) , based on SR-MPLS, to allow the execution of a global service chaining micro-services operated at both levels, to benefit of the best of each
- Definition of a MPLS proxy, intermediate between P4 nodes and VNFs, to keep VNF softwares unchanged
- Demonstrator implemented on a hardware P4 switch and VNFs running in software, for the use-case of a cloud gaming traffic detection

- The controller is currently a Python program, but it might be integrated with an orchestrator such as ONOS or MANO.
- Another option is to benefit from SR automatic announcement to populate the MPLS routing tables

Thanks

&

Questions