# Performance analysis of Open vSwitch

Adel Belkhiri Michel Dagenais

December 09, 2019

Polytechnique Montréal

Laboratoire **DORSAL**

# Agenda

## Introduction

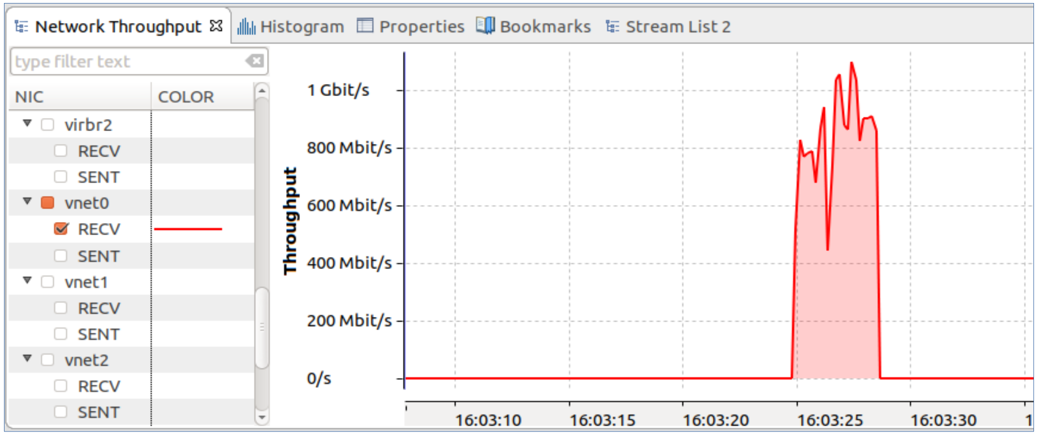
- Previous work
- Software-Defined Networking
- Open vSwitch

## Investigations

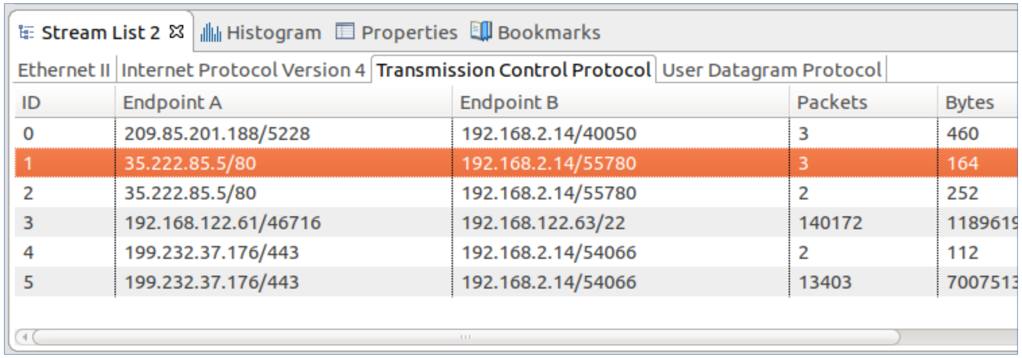
## Use cases

## Conclusion and future work

# Previously on VN analysis



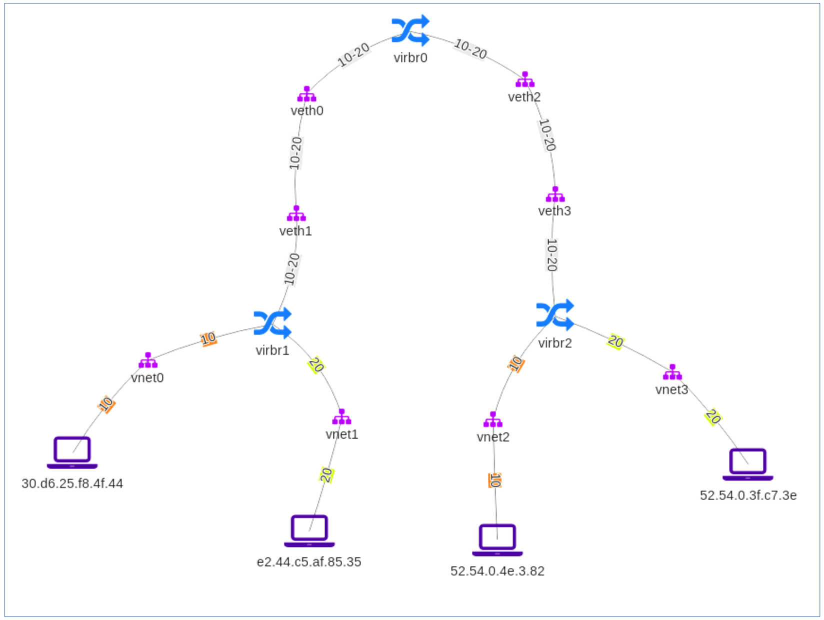

Throughput and packet rate per NIC


Stream List 2 | Histogram | Properties | Bookmarks

Ethernet II | Internet Protocol Version 4 | Transmission Control Protocol | User Datagram Protocol

| ID | Endpoint A | Endpoint B | Packets | Bytes |
|---|---|---|---|---|
| 0 | 209.85.201.188/5228 | 192.168.2.14/40050 | 3 | 460 |
| 1 | 35.222.85.5/80 | 192.168.2.14/55780 | 3 | 164 |
| 2 | 35.222.85.5/80 | 192.168.2.14/55780 | 2 | 252 |
| 3 | 192.168.122.61/46716 | 192.168.122.63/22 | 140172 | 1189619 |
| 4 | 199.232.37.176/443 | 192.168.2.14/54066 | 2 | 112 |
| 5 | 199.232.37.176/443 | 192.168.2.14/54066 | 13403 | 7007513 |

New Stream List view



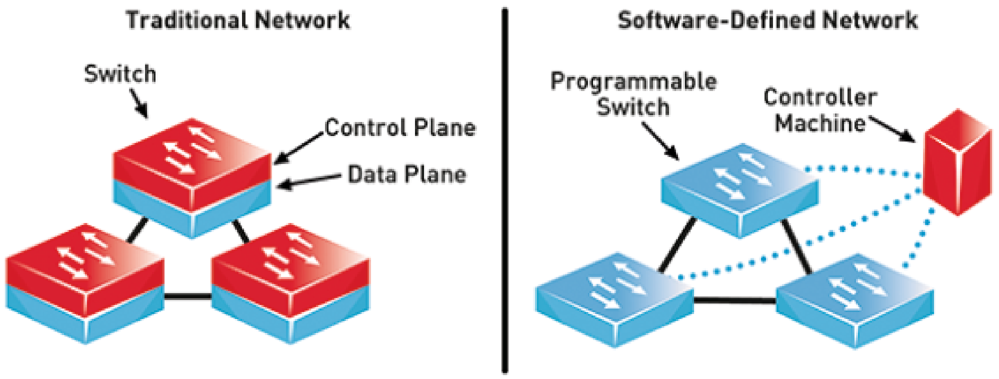

2 techniques to automatically discover a VN topology

# Network Virtualization

- **Network Virtualization** (**NV**) refers to abstracting network resources traditionally delivered in hardware to software.

  - NFV *(Network Functions Virtualization) :* Decoupling the network functions from proprietary hardware so that it can run on software or standardized hardware.

  - SDN (*Software-Defined Networking*) : Separation, at the hardware level, of the network control plane from the forwarding plane.




***** *Taken from :* *https://www.commsbusiness.co.uk/*

# Software-Defined Networking

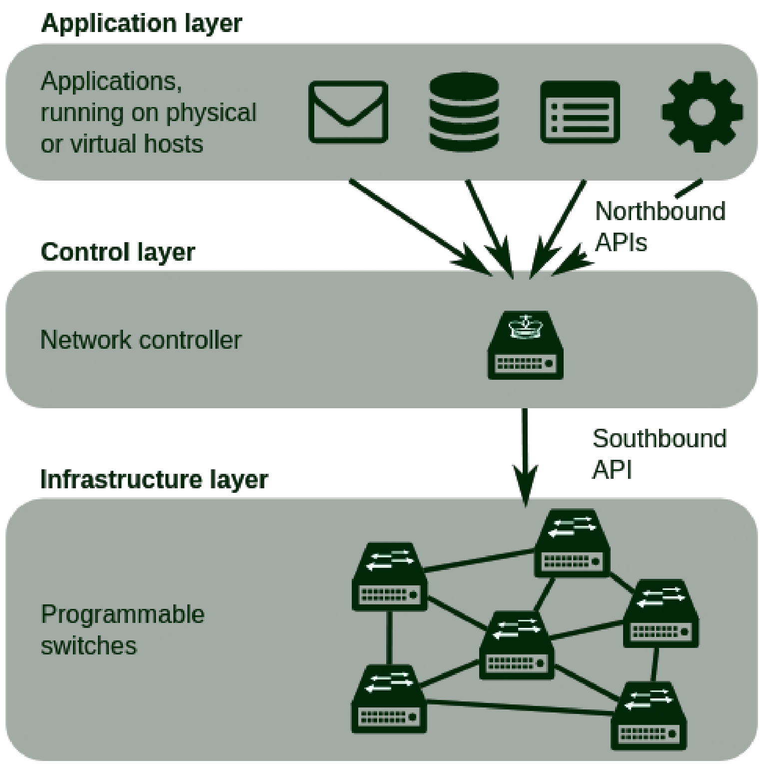
**Key benefits :**

Centralized provisioning

Reduced Hardware Footprint

Scalability

Security




**SDN Controllers :**

OpenDaylight

Floodlight

ONOS

**SDN switches :**

Indigo Virtual Switch (IVS)

Open vswitch (OVS)

OVS/DPDK

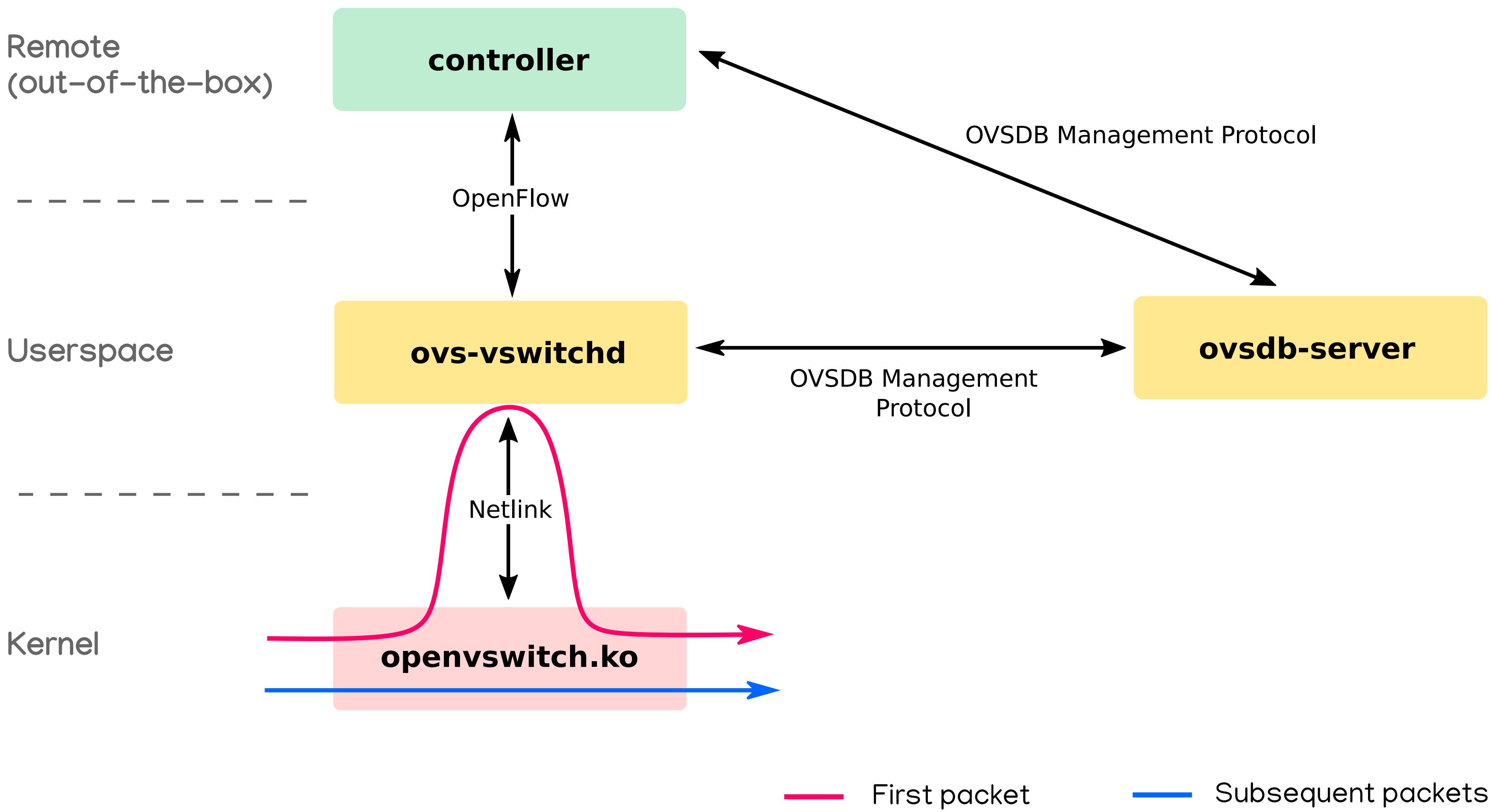
**Picture taken from :** https://dl.acm.org/citation.cfm?id=3164563

# Open vSwitch (1)

- A software implementation of a network switch/router
- First release : May, 2010
- An SDN switch : centralized control via OpenFlow protocol
- Many OSs are supported : Linux, BSD, Windows, ....
- Deployed on many cloud/virtualization platforms : OpenStack, OpenNebula, ...
- Support for several network protocols : Ipv4/Ipv6, TCP, UDP, VLAN, MPLS, sFlow, Netflow, ..

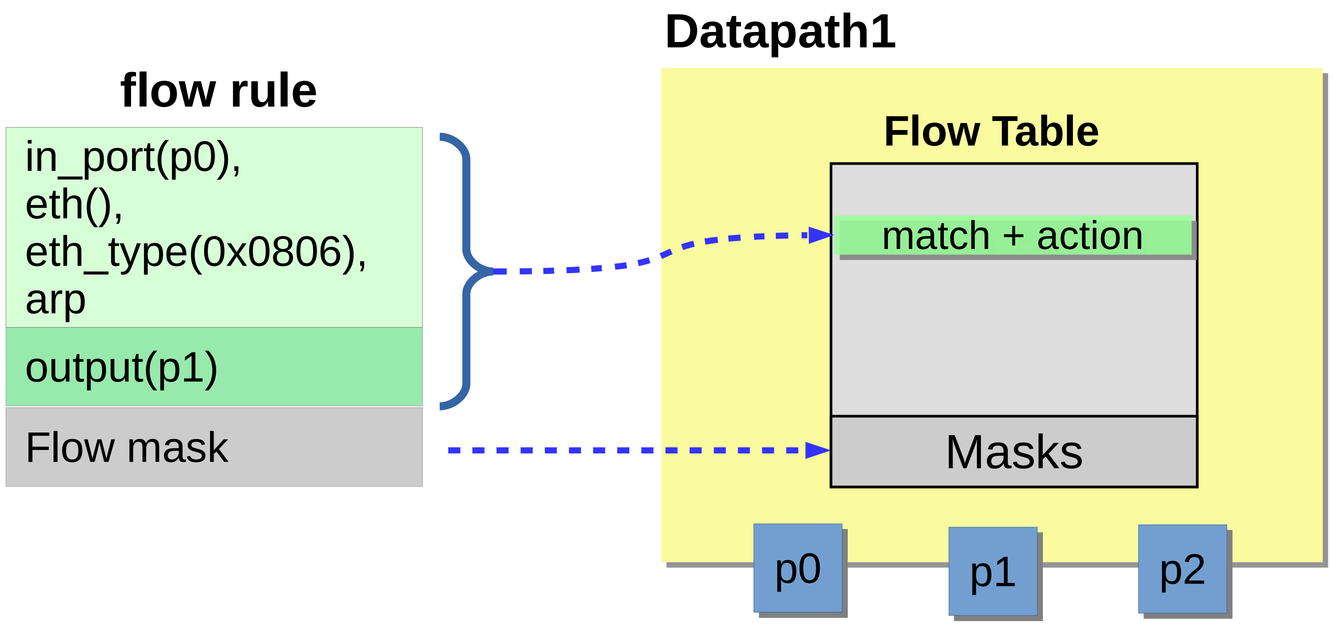
OvS Open vSwitch

# Open vSwitch (2)



# Open vSwitch (3)

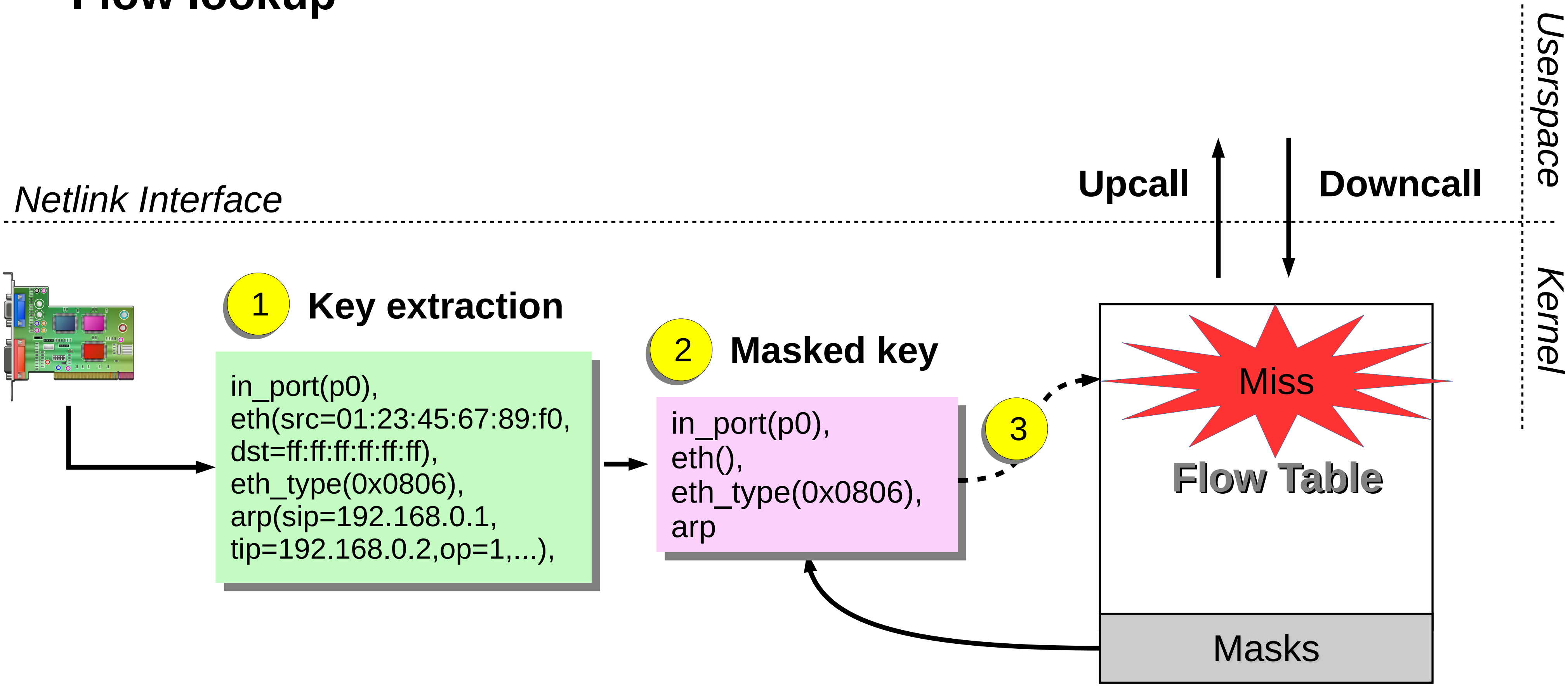
- **Flow-based policy :** Same actions on packets belonging to the same flow.
- **Flow :** Set of packets that share some common criteria (Packet header fields + metadata).




# ovs-dpctl add-flow datapath1 “in_port(0),eth(),eth_type(0x0806),arp()”, 1

# Open vSwitch (3)

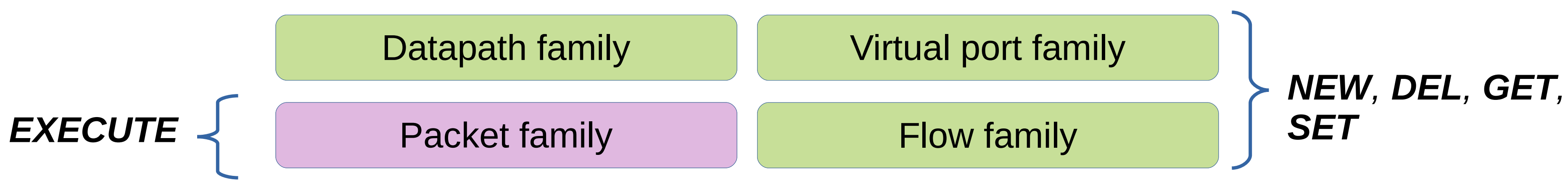
- **Flow lookup**



# Open vSwitch (4)

Userspace

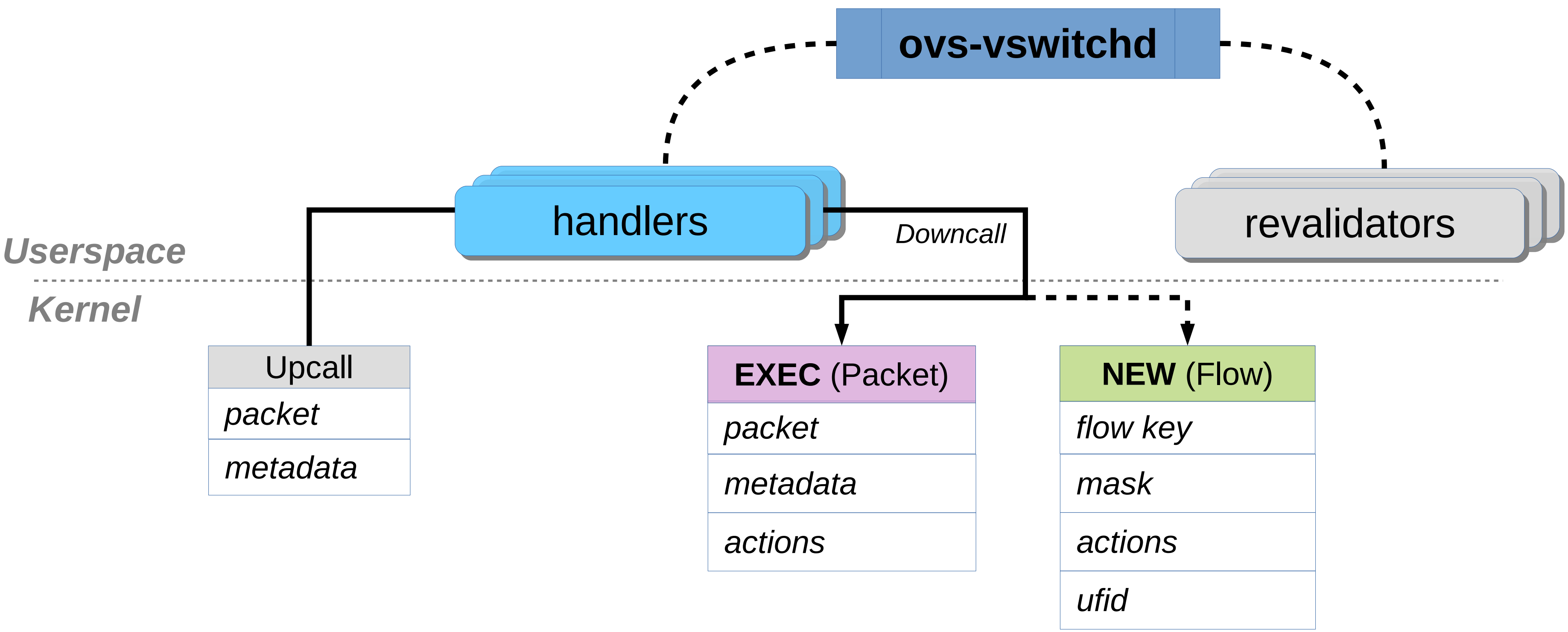
- Generic Netlink protocol :



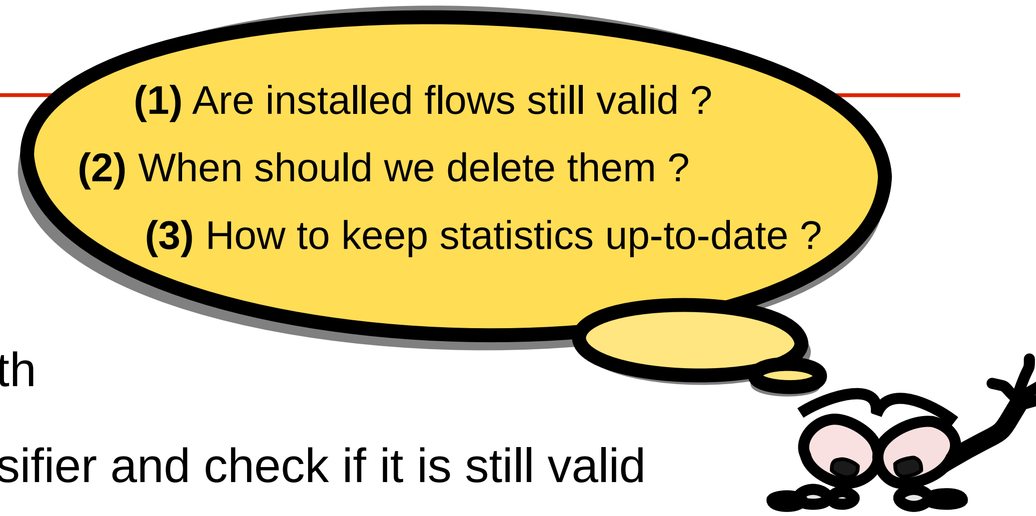

Kernel

# Open vSwitch (4)

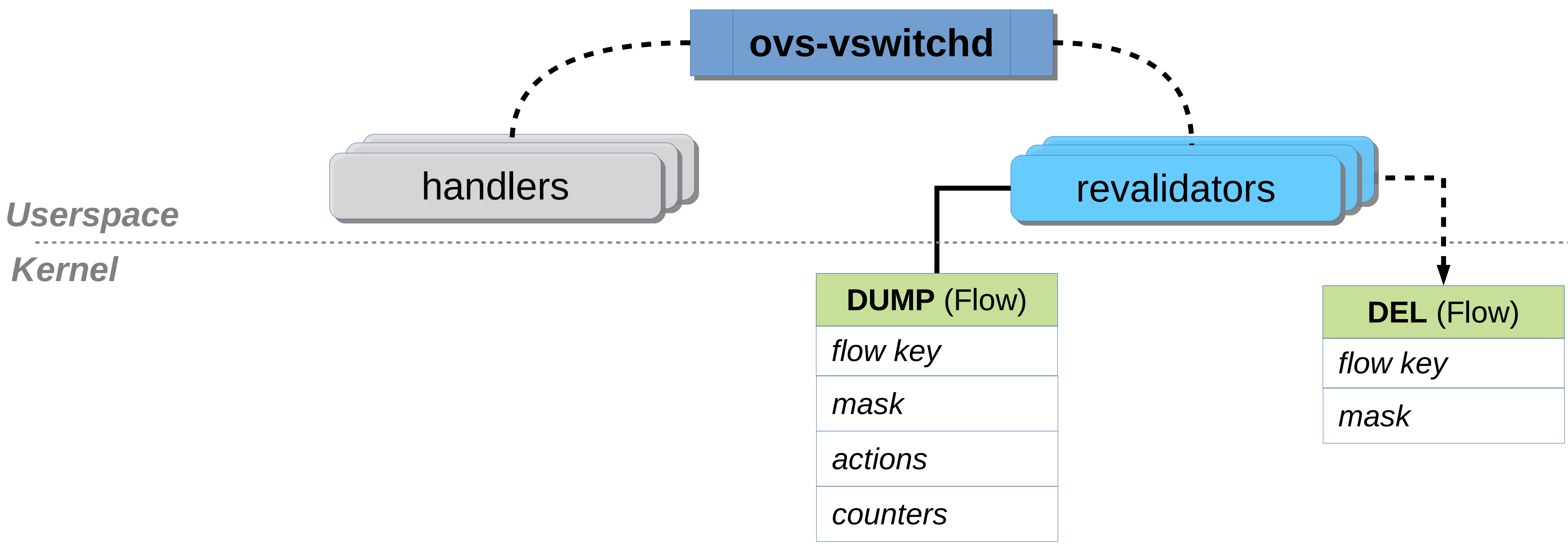
- **Handler threads :**
  - Process upcalls and reactively install flows in datapath flow table.



# Open vSwitch (5)




- **Revalidator threads :**
  - Periodically fetch flows from the datapath
  - Run each flow through ofproto-dpif classifier and check if it is still valid
  - Delete idle/incorrect flows
  - Update statistics



# Motivations

- OVS offers some debugging tools to help users design their flow rules.
  - **Example :** a tool to track a packet through the pipeline of OpenFlow tables without sending actual packets.

**ovs-appctl ofproto/trace**

- OVS offers several commands to allow users get a snapshot of a given internal state … but this latter could quickly change.

**Ovs-dpctl dump-flows**

- Many network monitoring protocol are supported (sFlow, Netflow, SPAN/RSPAN, …).

# Goal

- **Propose an efficient performance analysis tool to help (administrators) ...**

1) Analyze the performance of Open vSwitch based on adapted performance metrics.

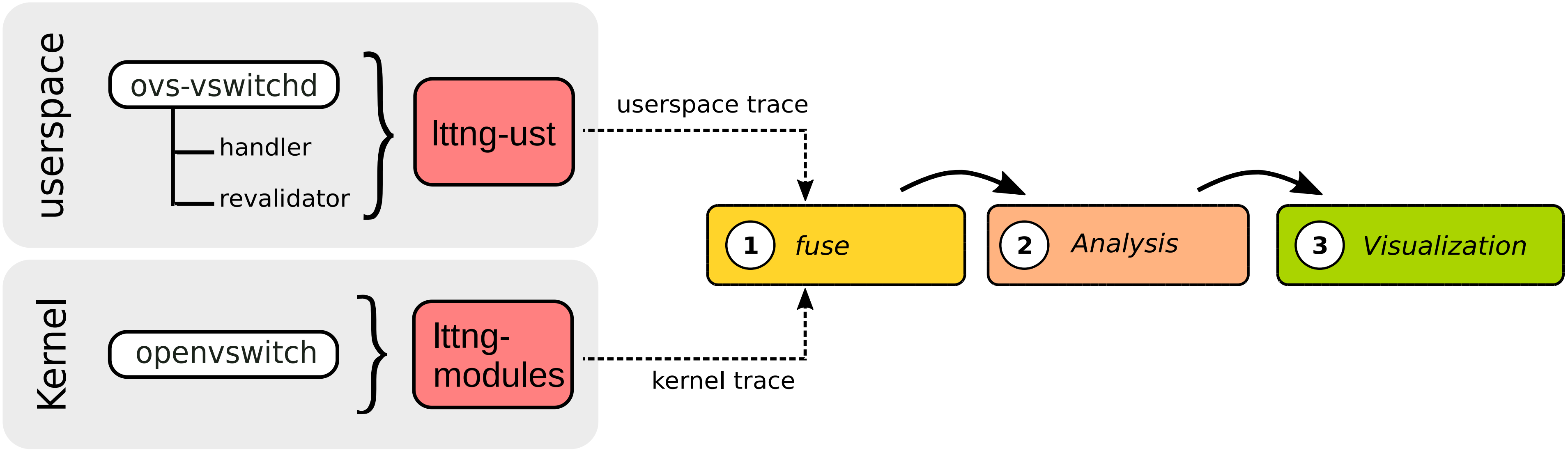
2) Troubleshoot its performance issues.

3) Understand the root causes of packet processing latencies

# Work Environment

- **Software :**
  - Open vSwitch (*version 2.11.9*)
- **Data Collection :**
  - LTTng (*version 2.10*)
  - Kernel and userspace tracing / static instrumentation
- **Performance Analyses :**
  - Trace Compass framework

# Framework Architecture



# Performance metrics

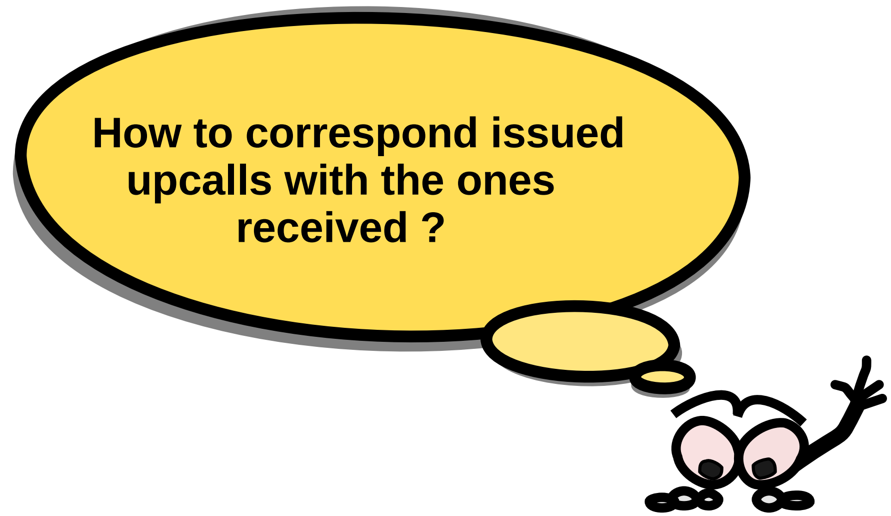
- Microflow and megaflow cache hit rate :

$$Utilization_{EMC} = nbHit(EMC) * 100 / (nbHit(EMC) + nbHit(megaflow) + (nbMissUpcalls))$$

- Number of installed flows in the datapath.
- Packet rate per flow
- Evaluate the load of revalidator-threads :
  - Required time to re-validate datapath flows
  - Computed datapath flow-limit

# Performance metrics

- Evaluate the load of handler-threads :
  - Upcall waiting queue size
  - Average upcall waiting/processing time
  - Upcall rate issuing per port/Type



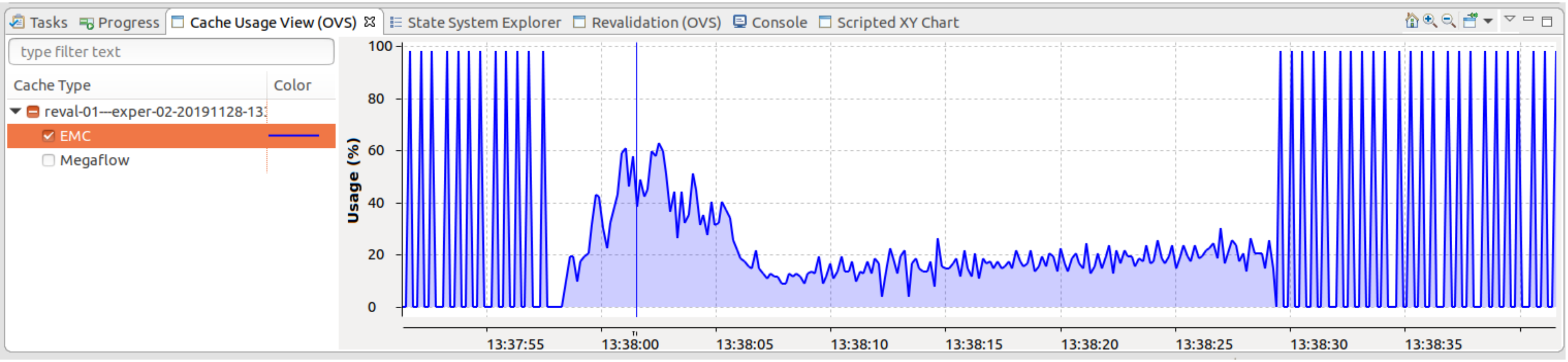

A Kernel module to tag packets encapsulated in the upcall (skb-mark field) just before sending them.

# Use case (1)

## Impact of an inappropriate configuration

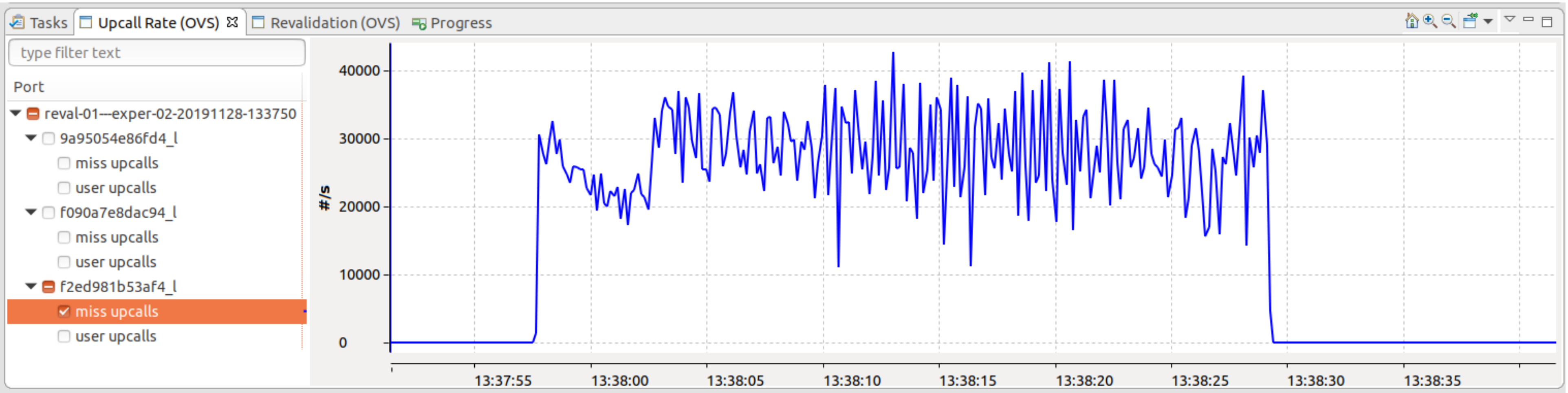
# Use Cases (1)

- **Experiment setup :**
  - OVS : 2 handler threads and 1 revalidator thread
  - Maximum number of cached flows (flow-limit) = 200k
  - Flow timeout = 10 seconds
  - CLI to install 200k flow rules
  - Trex traffic generator : 200K continuous streams with a rate of 1Mpps

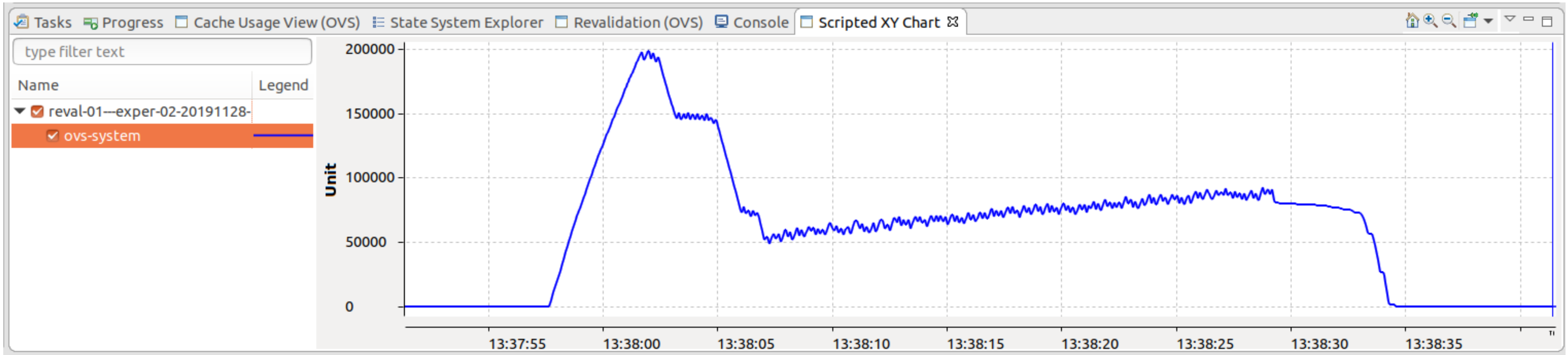



OVS cache usage

# Use Cases (1)

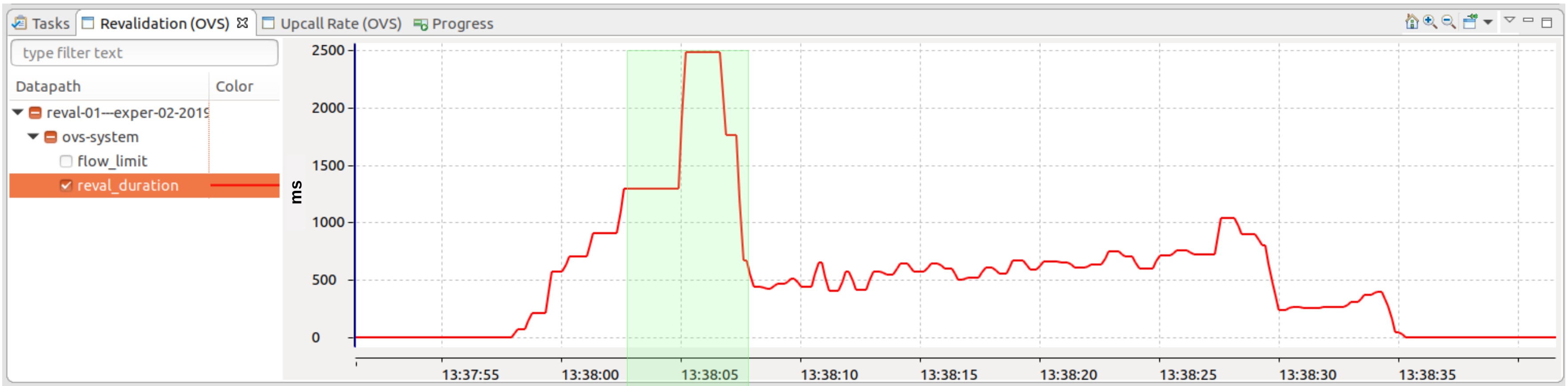



Rate of upcall issuing

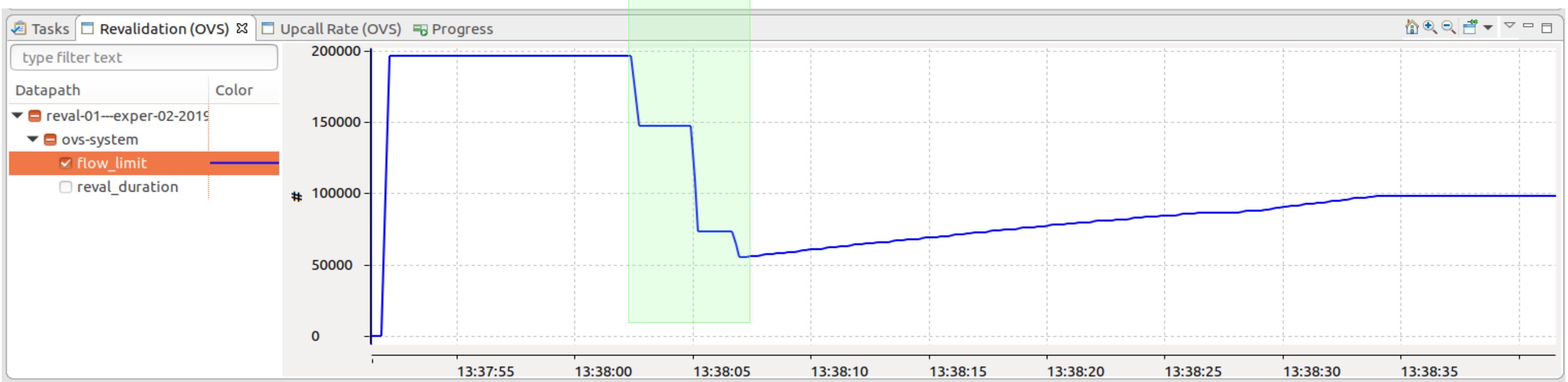



Number of active flows in datapath flow table

# Use Cases (1)




Re-validation duration

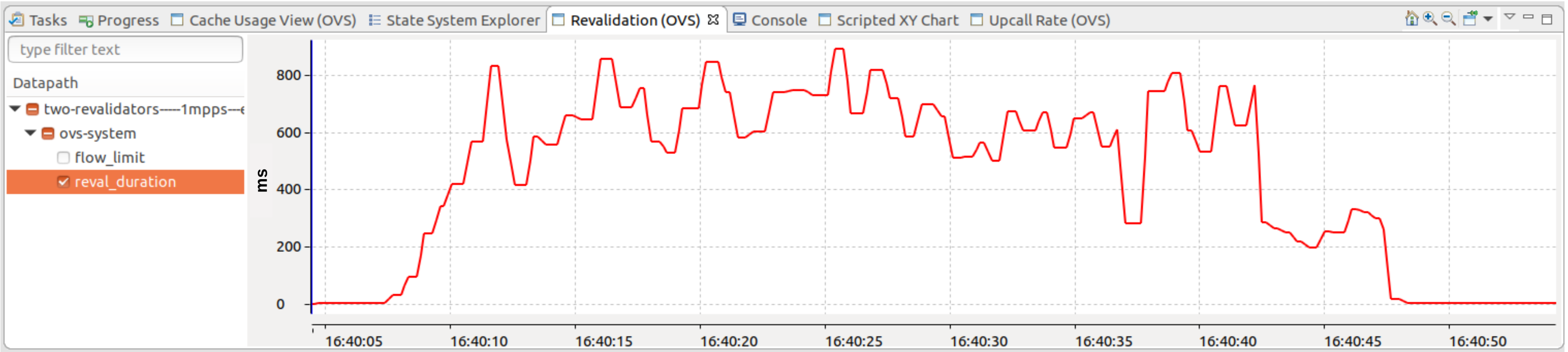



Computed flow-limit

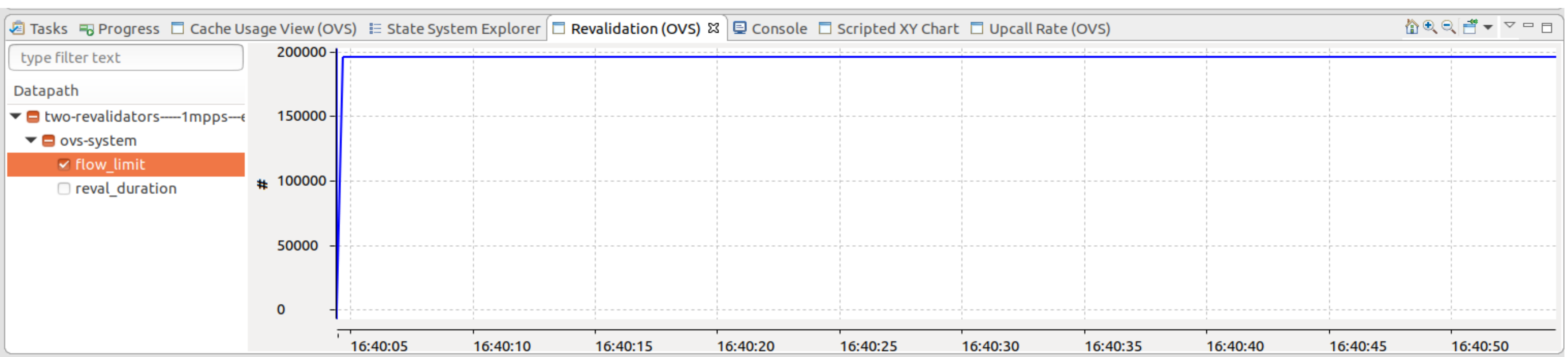
- The Revalidator-thread was not able to perform re-validation in time.

# Use Cases (1)

- Adding a second revalidator-thread solved the problem.




Re-validation duration



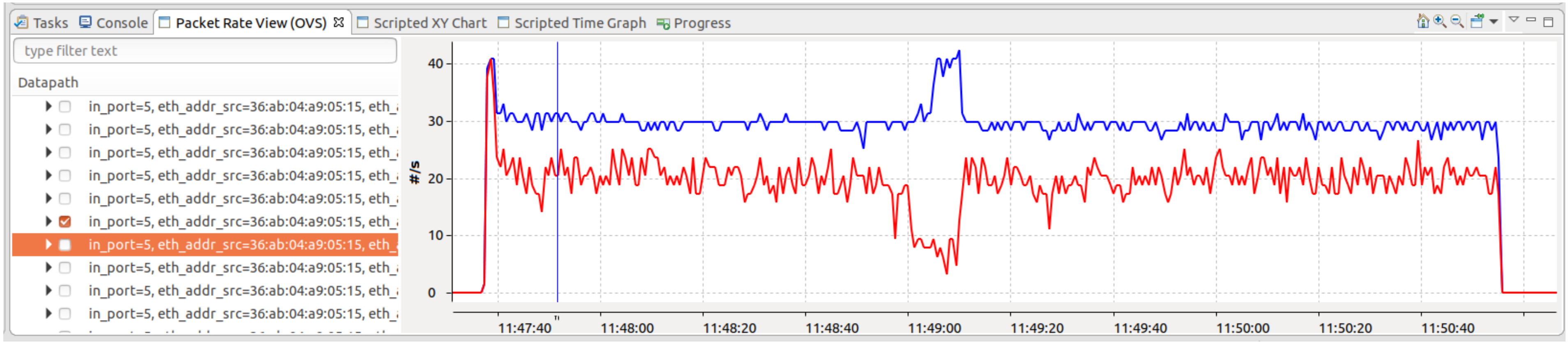

Computed flow-limit

# Use case (2)

What about OVS fairness ?

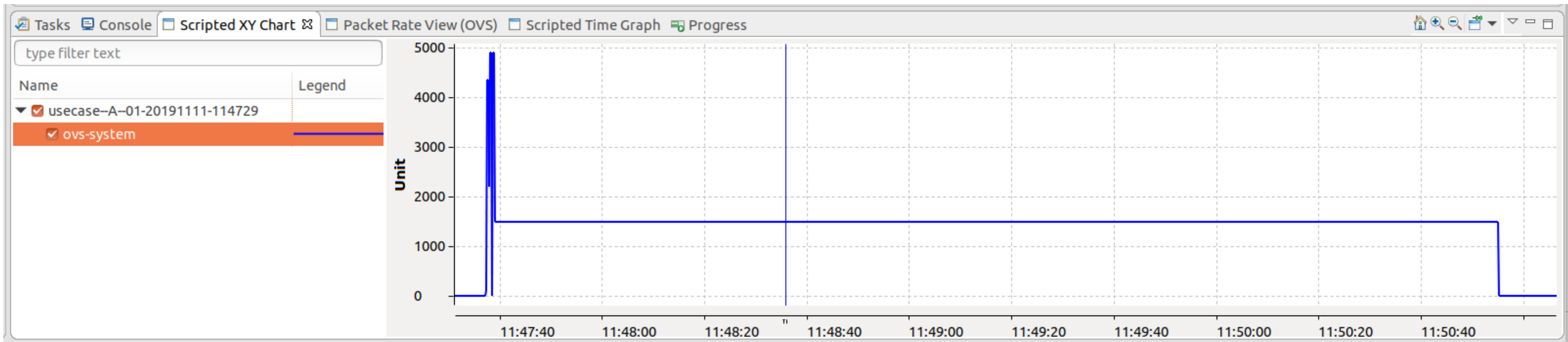
# Use Cases (2)

- **Experiment setup :**
  - OVS : 2 handler threads + 2 revalidator threads
  - Maximum number of cached flows (flow-limit) = 1000
  - Using CLI to insert 5000 flow rules
  - Trex traffic generator : 5000 continuous streams with a rate of 500Kpps

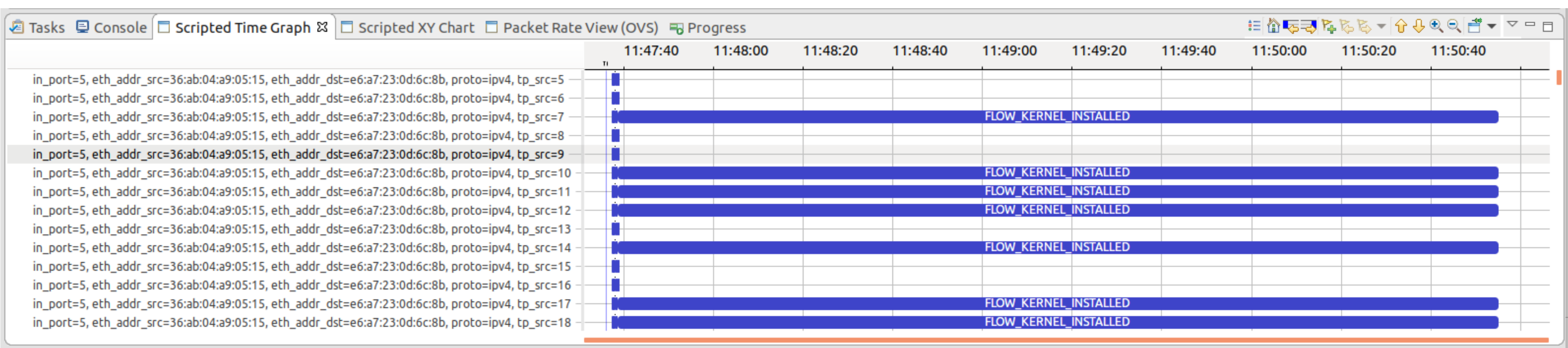



Packet rate per flow

# Use Cases (2)




Number of active flows in datapath flow table




Caching duration of flow rules

# Use case (3)

## kernel/userspace flow key mismatch

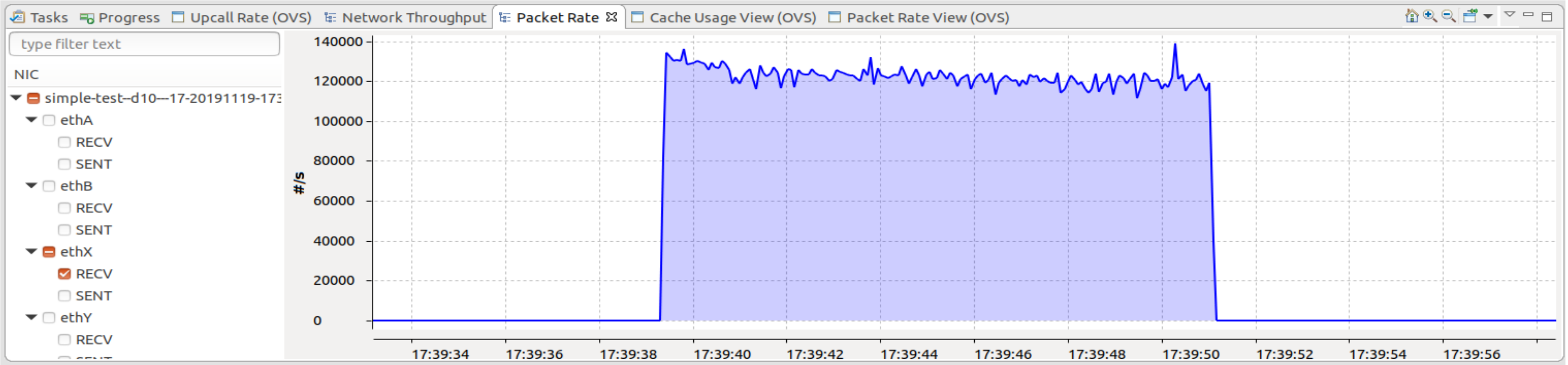
# Use Cases (3)

- Kernel and userspace modules can use different flow key representations.
- When the degree of support for a specific network protocol is different, a mismatch can happen.
  - **Example :** datapath only supports IPv4 while userspace supports both IPv4 and IPv6.
- **3 cases :**

Kernel flow key = / > / < userspace flow key

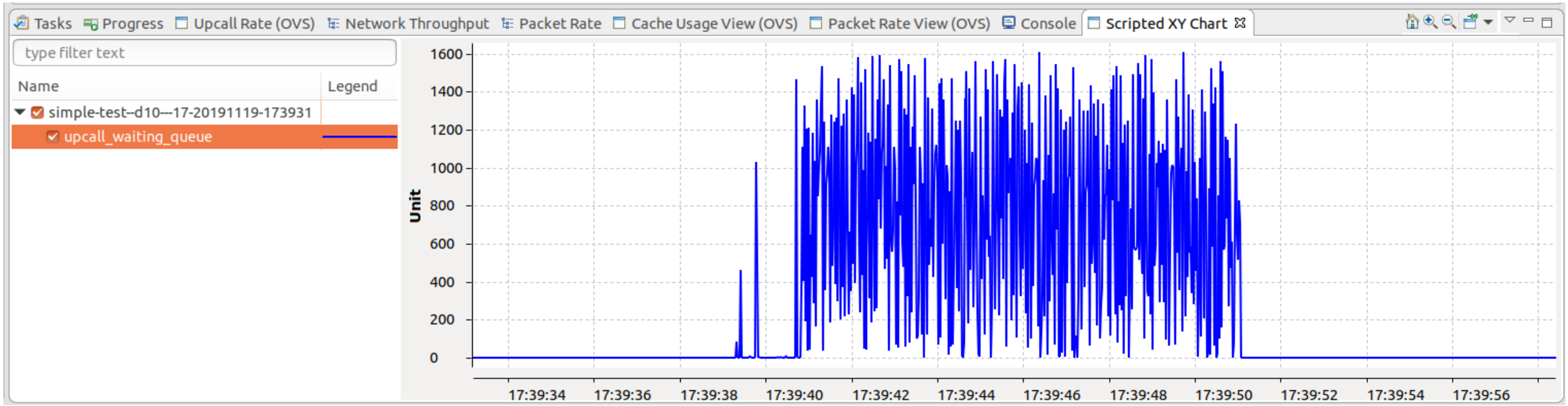
# Use Cases (3)

- **Experiment setup :**
  - OVS : 2 handler threads + 2 revalidator thread.
  - Maximum number of cached flows (flow-limit) = 200k.
  - 1 flow rule to pop an MPLS header and output it to a given port.
  - Trex traffic generator : 1 stream with a rate of 200Kpps – Packets having **double MPLS headers**.

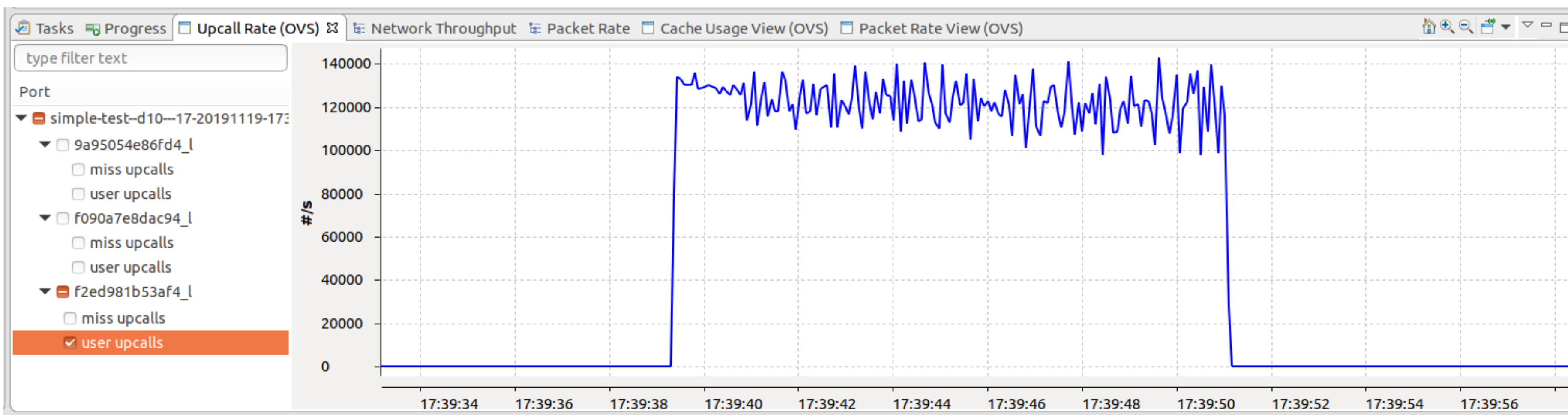



Packet rate at reception.

# Use Cases (3)

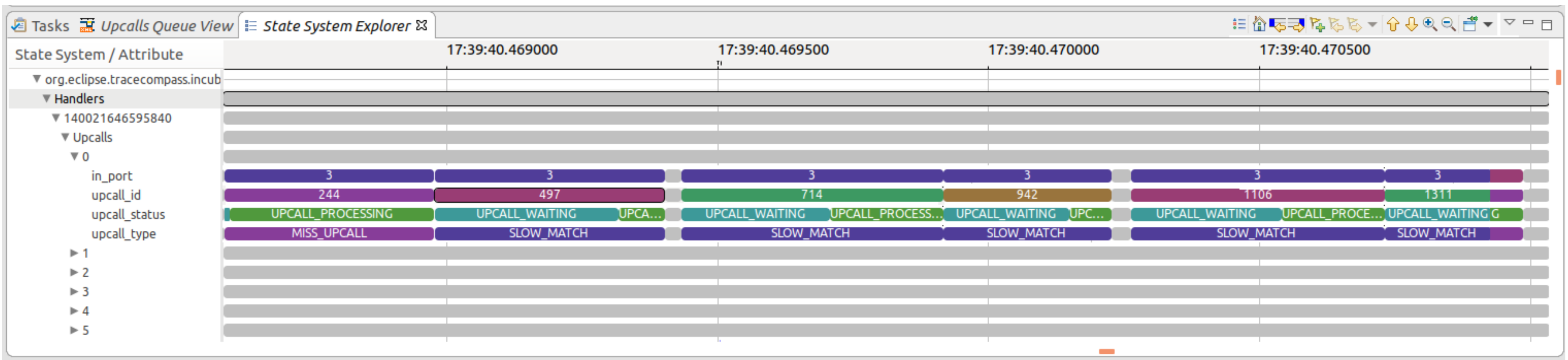



Upcall queue size



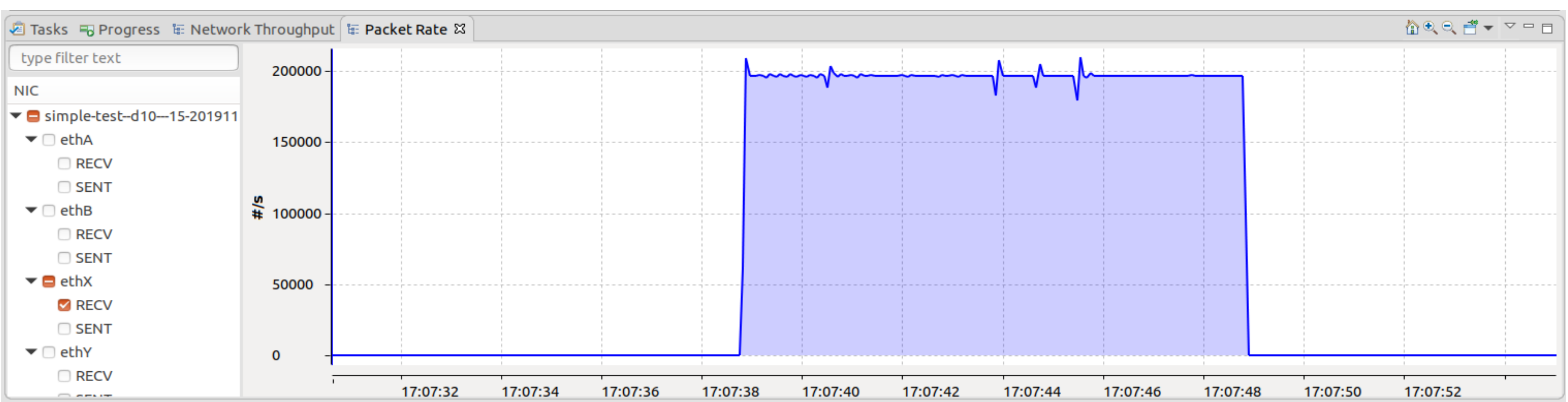

User upcall rate issuing

# Use Cases (3)




Upcall type/waiting/processing time.

- **Case when the traffic packets have only one MPLS header :**




Packet rate at reception

# Conclusion and Future Work

- Industry needs efficient tools to diagnose problems in software-defined networks and identify the root causes of traffic latencies.

- We are looking for new use cases and problems to solve in order to improve our analyses and tools

- **Future work :**

  - OVS/DPDK (Data Plane Development Kit) : better performance than standard OVS.

  - OVS entirely in userspace.

# Questions?

adel.belkhiri@polymtl.ca