Merging NFV with SDN
break the boundaries between the two domains!

Daniele Moro, Antonio Capone, Michele Mangili, Davide Sanvito
SDN/P4 Workshop - Politecnico di Milano
02/02/2018
The two domains

SDN and NFV are two separate domains, one useful to the other but with a clear separation between them.

- SDN: traffic steering, packet switching, function chaining...
- NFV: agile Network Function allocation, easy to allocate new functions....

Can we try to merge them? Find ways to jointly orchestrate the two domains?
Limitations in NFV

- Scalability to hundreds of Gbps (difficult with general purpose HW)
- NFV is increasing its computation requirements
- Network functions have common patterns not currently exploited
Limitations in SDN

- It is used mainly for traffic steering
- Lots of capabilities
  - OpenFlow
  - Data plane programmability (P4 - Tofino)
  - Stateful data plane (OpenState, OPP, “Domino-Banzai”)
  - SmartNIC
  - “Networked” FPGA (NetFPGA)
- Controller is used only as an interface between Applications and Network
IDEA: Break the boundaries!

We want to propose a framework to merge NFV and SDN

It will allow to:

- **offload computation** from NFV to SDN in an easy way
- program **easily** and **jointly network and functions**
- **optimize how** and **where** the functions are placed
Network functions offloading

- Already a lot of examples (e.g. DPI offloading, iptables offloading, checksum offloading...)
- No commonalities that can be exploited for other works
- Offloading usually built as a *specific implementation*

→ We can identify different “offloading layer”
→ We can divide network function to exploit commonalities
Different offloading layers of Network Function

Network functions can be allocated on different offloading layers depending on:

- network capabilities
- function requirements

1. Dedicated Middlebox
2. VM/Container
3. SDN Controller
4. Data Plane
NFs can be divided in smaller functions that can be exploited by other NFs. We want to introduce the concept of **reusability** of network functions.
Network Micro-Functions Library

- Matching (header/payload)
- Actions (drop, output....)
- Filtering
- Metering
- Replication
- Aggregation
- Small computation (header/payload) (e.g. checksum verification, sum....)
- Load balancing (splitting)

- Classification
- Shaping/Throttling
- Encryption
- Feature extraction
- Traffic generation
- Inter-flow/intra-flow conditions
- End system protocols (e.g. TCP or upper layers)
Network as a set of multipurpose devices

Each network device has some **computation capabilities** other than simple network related functions (e.g. packet switching, traffic steering).

We can map the previously identified $\mu$-Functions in the network devices.

We can create a **unified view** of capabilities of network devices.
A simple firewall can be divided into:

- **Header matching**
- **Action (Drop/Output)**
- **Traffic analysis (further separation)**
  - Filtering
  - Classification

- Header matching, actions and filtering can be implemented in the data plane
- Classification can be implemented as DPI in a VM
Example

Network Function

INSTANTIATION

ORCHESTRATOR

SDN domain

NFV domain

Control plane

Data plane

μNF1
μNF2
μNF3
μNF4
Starting points (Framework)

- Network Programming Language: *Merlin (extension)*
- SDN Controller: *ONOS*
- Programmable (Stateful) Data Plane: *OPP/OpenState/P4*
What we need (for the Framework)

- **Offloading** use cases!
- Identify **commonalities** between network functions (identify more micro-functions)
- **Divide** network functions into micro functions
- **Language** to define how network and functions can cooperate
- **Compiler**, multiple optimization models with different objectives:
  - network capabilities
  - network requirements (limitations in terms of bandwidth, latency, servers...)
- **Instantiation process** that translates high level language to low level network implementation (merge language with optimization models)
Defined as multiple optimization models with different objectives

- **Network Capabilities:**
  - available offloading layers
  - layers capabilities
  - devices capabilities
  - network topology

- **Network Requirements:**
  - bandwidth
  - latency
  - max number of instantiable VMs
  - ...

Daniele Moro
Offloading example: DPI offloading to stateful data plane

Monolithic DPI NF divided into:

- filtering
- statistics collection
- classification (less traffic to analyze)
- analysis

Davide Sanvito, Daniele Moro and Antonio Capone

"Towards traffic classification offloading to stateful SDN data planes", 2017 IEEE Conference on Network Softwarization (NetSoft)
Some results

Davide Sanvito, Daniele Moro and Antonio Capone
“Towards traffic classification offloading to stateful SDN data planes”, 2017 IEEE Conference on Network Softwarization (NetSoft)
Conclusion

Offloading to SDN is an hot topic.

No one has developed a common framework to work with.

Short term objectives:

● More offloading use cases
● Identify commonalities between network functions
● Define optimization models with different trade-off between constraints and network capabilities to offload NFs

Long term objectives:

● Define a language to merge both high and low level functionalities
● Compile the language with different network objectives and capabilities
● General framework for the compilation process
Micro function for edge computing

The framework and an extended library of micro-functions can be provided by network operators to developers for building application at the edge of the network.

Micro-functions as base functions for **Edge Computing**

The framework can be an *enabler* for Edge Computing
Help us!

Early stage research work!

**Feedbacks, suggestions** and also **critiques** will be really helpful.

If someone is interested in a **collaboration**, let us know!!!

Daniele Moro

daniele.moro@polimi.it