# Network virtualization and programmability

Didier Colle*, Bart Jooris*, Pasquale Gurzi**, Mario Pickavet*, Piet Demeester*
* Ghent University – IBBT, Gaston Crommenlaan 8 bus 201, 9050 Gent, Belgium
** VUB CoMo lab, Pleinlaan 2, 1050 Brussels, Belgium

***Abstract*-- we present network virtualization (building virtual or logical networks over a physical infrastructure) and network programmability (allowing the network operator to at least control the network but more fundamentally to define its behavior) concepts.**

## I. Introduction

Building networks is a very costly issue. Therefore, it is important to save as much as possible costs. Two approaches are crucial in this respect:

1. Building infrastructures delivering tremendous amount of network capacity reducing the relative per-unit cost thanks to the economy of scale effects. Of course, gaining from economy of scale effects only makes sense when the infrastructure can be shared amongst different parties in order not to waste / underutilize its massive capacity: virtualization is crucial to realize this. For example, WDM networks allow sharing the fiber / transmission infrastructure by multiplexing virtual IP network links over the same fiber pair.
2. Building as flexible as possible infrastructures without jeopardizing their performance in order to gain from economy of scope effects. The increased flexibility through configurability and programmability enables the infrastructure to serve a broader scope of applications rather than serving a single/few applications requiring building and managing several infrastructures in parallel and/or replacing them more frequently as requirements by the application changes over time.

This paper applies these key concepts to network infrastructures.

## II. Network Virtualization

Bottom line network virtualization means abstracting the physical infrastructure from the virtual networks. Thus, overlay networks are the simplest form of network virtualization: e.g., in IP-over-Optical Transport Networks (OTNs) the direct IP links are replaced by lightpaths across the OTN.

A more advanced form of network virtualization refers to the fact that part of the physical infrastructure is handled as a virtual network. In this way, the abstraction guarantees that the virtual network is hidden from what happens on the rest of the infrastructure allowing the infrastructure to be shared by several virtual networks. IP routers may instantiate multiple so-called Virtual Routing and Forwarding (VRFs) instances: VRFs are typically instantiated on a per-VPN basis in the Provider Edge (PE) routers in BGP/MPLS IP VPN provider networks [1]. An advantage of the independent VRF instances is that overlapping address ranges in the different VPNs don't lead to conflicts. In optical cross-connects, this would translate to define a virtual cross-connect by selecting the physical ports part of that virtual cross-connect.

Instantiating multiple virtual instances inside a physical network node and interconnecting these virtual network nodes (multiplexed over shared links) enables "slicing" huge physical network infrastructures that can benefit from the economy of scale effects. A slice in an OTN (see Fig. 1) may thus correspond to a wavelength range for which the virtual cross-connects can only cross-connect these wavelength between the network links part of the slice.

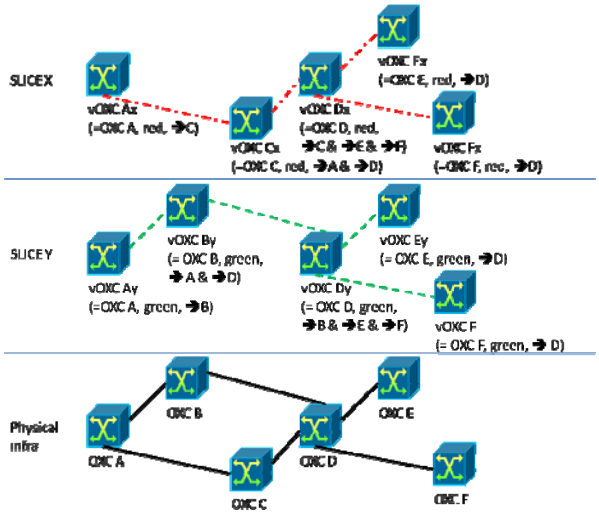


**Fig. 1: network slicing in OTNs.**

Another form of network virtualization is turning general purpose machines into network equipment: e.g., software routers like the CLICK modular router are an example of this form of virtualization / hardware abstraction [2].

## III. Network Programmability

Bottom line, network programmability means that multi-purpose bare hardware can be (re-)programmed / turned at any time into a specific product addressing specific (single or combination of) purpose(s) that may change over time. For example, today no or very few products can act as PBB-TE or (G)ELS switch [3] (or a combination of both in case of network slicing), although they are both a kind of label switch (functionally similar as an MPLS switch) where the labels are encoded in the Ethernet frames (either VLAN-ID + MAC address or

This work was supported by the European Commission through the OFELIA, SPARC and BONE projects. The work was also supported by the Flemish government (FWO-Vlaanderen) through the project G.0107.05 and G.0578.08.

only the VLAN ID): network programmability avoids these product specific limitations.

Some initiatives going in that direction have already been initiated. The Forwarding and Control Element Separation (ForCES) [4] IETF working group defined a framework and protocol to directly control/configure the functional elements of an IP router. The OpenFlow initiative [5] provides a protocol interface that allows configuring a flow table consisting of a bit string filter identifying the flow and the corresponding actions (like forwarding on a particular port or dropping packets) belonging to that particular flow. In both cases, a split architecture is envisaged: the node controllers do not need to reside anymore in the nodes (network elements) but can run somewhere else. They differ in the sense that the ForCES initiative is currently more advanced in opening up different functional elements, but focusing on IP technology, whereas the OpenFlow protocol is more generic as it does not restrict itself to a specific network technology by more directly addressing the bare hardware (e.g., TCAMs in which the flow tables are stored). The OpenFlow protocol thus allows for example traffic engineering particular flows that can be defined on a MAC address, IP address and/or TPC/UPD port level while keeping the default routing for the other traffic.

The European FP7 research SPARC [7] will define a carrier-class split architecture blueprint, by extending the OpenFlow protocol and prototyping the proposed solution. Focusing on carrier-class split architecture means the SPARC project will research issues like scalability, reliability, etc.

A split architecture where external controllers can control/configure functional elements or bare hardware is already a huge step in the direction of network programmability. However, network programmability at its full potential should also be capable of enabling the programming of functionalities into general purpose hardware. For example, a network element based on the CLICK modular router [2] would require a protocol interface that allows external controllers to load the network element with any CLICK element and to specify how these CLICK elements should be linked together. Although CLICK modular routers are software based and thus not capable of delivering the highest performance, approaches as in the NetFPGA initiative [6] have that potential as the network ports connect to a large FPGA that can be reprogrammed with any bitware as long as it fits in the FPGA footprint.

## IV. The Programmable Network Slice

As discussed above, both the network virtualization and network programmability concepts are important tools in reducing costs. A mix of both results in an even higher cost saving potential.

Fig. 2 illustrates how this can be achieved. At the left, the programmable network element (e.g., an OpenFlow switch) is shown. In the middle, a controller proxy (in case of OpenFlow this is called a FlowVisor) guarantees that controllers belonging to one network slice cannot control/affect/access the virtual node from another slice. At the right side, a controller per slice is shown that controls the part of the hardware/network element belonging to that slice through the controller proxy.

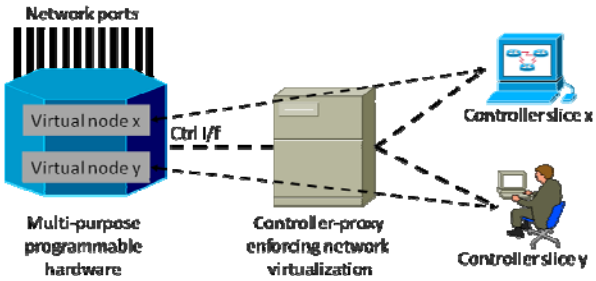


**Fig. 2: programmable network slices**

The European FP7 FIRE facility project OFELIA [8] will build an experimental facility supporting the Future Internet research activities in Europe by allowing experimenters controlling network slices consisting of several technologies through the OpenFlow protocol.

## V. Flexible Programmable Optical Network Nodes

Within the OFELIA project, a flexible optical network node will be designed and build. The intention is to develop a mother board providing an I2C or SPI interface to a General Purpose I/O (GPIO) expander chip. These GPIO pins are then grouped in smaller sets such that each set is brought in a standardized manner to a daughter card featuring an optical component like a 2x2 optical switch module. A controller card (e.g., an embedded PC like the ALIX) can then control through the I2C or SPI interface a bunch of optical switch modules plugged in onto a (few) mother board(s). From a software perspective, a virtual switch multiplexer module on top of the I2C or SPI driver module will translate the GPIO addresses within a slice to the correct global GPIO pin addresses and pass on the commands to/from those GPIO pins. Per slice, a virtual switch software module will then translate the OpenFlow commands to the proper GPIO signals/commands and vice versa, according the specific wiring of the optical components in that slice.

Assuming all daughter cards feature 2x2 optical switch modules, one experiment may use 6 optical switch modules to create a 4x4 OXC, while another one can in parallel build a ROADM, by wiring its optical switch modules to passive WDM (de)multiplexer modules.

## References


[1] E. Rosen, Y. Rekhter, "BGP/MPLS IP Virtual Private Networks (VPNs)", RFC4364, February 2006.

[2] CLICK modular router project: http://read.cs.ucla.edu/click/click.

[3] D. Colle, W. Tavernier, A. Gladisch, "Ethernet: beyond the LAN?", DRCN2007 tutorial, October 7th, 2007: http://ibcn.intec.ugent.be/downloads/Tutorial%20DRCN2007.pdf

[4] ForCES IETF WG: http://datatracker.ietf.org/wg/forces/

[5] The OpenFlow project: http://www.openflowswitch.org/

[6] The NetFPGA project, http://netfpga.org/

[7] FP7 SPARC project: http://www.fp7-sparc.eu/

[8] FP7 OFELIA project: http://www.fp7-ofelia.eu/