Software-Defined Networking: Perspectives, Requirements, and Challenges

Antonio Marcos Alberti
Instituto Nacional de Telecomunicações - INATEL.
Av. João de Camargo 510. CEP 37540-000. Santa Rita do Sapucaí, Minas Gerais, Brazil.
alberti@inatel.br

Abstract—Software-defined networking is emerging as a new paradigm for network design and implementation. However, many issues regarding this paradigm have been explored only on specific scenarios, e.g. network operating systems, networking customization, open commodity equipment, software-defined radio, data centers, etc. More general and deep studies are necessary to explore better this paradigm as it emerges as a key ingredient for future network architectures. This paper provides a first glance discussion on the perspectives, requirements, and open challenges behind software-defined networking.

Index Terms—Software-defined networking, virtualization, network operating systems, architecture, future Internet.

I. INTRODUCTION

What does this mean software-defined networking? Networking can be defined as the act of establishing a network among equipment in order to exchange data and information. Software-defined means that the functionality of "something" is defined by software, i.e. it works accordingly to some controlling software. Therefore, in the context of this paper, software-defined networking (SDN) [1] means to establish networks where equipment functionalities are controlled by software.

The equipment could be user devices, routers, base stations, radios, switches, computers, i.e. everything that could be used to do networking. Functionalities could be routing, forwarding, queueing, scheduling, or any network functionality. Therefore, SDN establishes networks where hardware implemented network functionality is controlled by software. In this context, what are the perspectives behind this paradigm in a broad sense? How it could be used on new designs targeting future networks? What are the technological requirements to apply such paradigm? What are the open challenges? This paper provides a preliminary outlook on these questions.

II. PERSPECTIVES

SDN exposes substrate resources to software. If generically applied, allows to virtualize equipment, decoupling hardware from software evolution. The implications are:

- Developers can generalize/homogenize functionalities on their equipment. They can use state-of-the-art technologies to develop high capacity, general usage hardware. The tuning on how these general functionalities will work is given by the software. An example of device aligned with this idea is the software-defined radio (SDR) [2][3].
- Control software can evolve to new versions without requiring general hardware replacing. In other words, owners do not need to throw away their hardware just because some control protocol needs to change. OpenFlow is an example of a network operating system that centralizes control of open commodity hardware [4].
- Reconfigurable hardware can implement functionalities according to software controls. This is already possible with field-programmable gate arrays (FPGAs) [5]. An example is the NetFPGA platform [6]. In this case, the software can even control how the hardware is going to evolve.
- Sharing of SDN equipment via software. This enables the reduction on energy consumption; the concurrent implementation of different software controls for slices of the shared hardware; the real performance comparison among candidate software controllers; the independent migration of software on different slices, and the mobility of software to compatible hardware when necessary. The former possibility allows the customization of hardware controls as someone moves in the network.
- Raising of virtual networks. SDN allows the creation of virtual networks that take advantage of customized slices of the substrate resources.

III. REQUIREMENTS AND CHALLENGES

SDN requirements are close related to the prerequisites for network [7], [8] and computer [9], [10] virtualization.

A first hardware requirement is genericity. Functionalities on software-defined equipment need to be generic enough to be customized by different software instances. Also, they need to be fully programable. Therefore, a challenge is how to generalize networking functionalities to be potentially used by several technologies. Take for example the multiple access control (MAC) functionality on link layer. How to implement a software-defined MAC that can be used simultaneously not only by Wi-Fi, but also by long term evolution (LTE) [11]?

Another requirement is related to the performance. One can not expect that a low-performance hardware can be simultaneously shared by multiple virtual instances with good performance. In addition, network equipment developers are adopting multicore platforms to implement their equipment. Examples are Huawei LTE base station [12] and Intel multicore packet processing [13][14]. Real-time software is also
fundamental to accomplish protocol time sensible tasks at software layer. Therefore, developing platforms are evolving towards multi-core hardware with multi-target, real-time operating systems (RTOS) [15].

Isolation is another requirement, since the shared hardware resources can not intervene in other concurrent resources. In this context, a challenge is how to isolate slices of virtualized hardware. For example, how to isolate slices of a software-defined MAC used concurrently by Wi-Fi and LTE? In addition, software control needs to be securely and privately exchanged. One software technology can not interfere on the controls aimed at other technology. Even the privacy regarding control exchanging is important. In addition, inconsistent software controls can create instability on hardware. This is a significant risk because unstable controls may create instability and affect other technologies on the same device. Also, software stability needs to be granted.

Another issue is reconfigurability. It means that the hardware can modify its building blocks and functionalities depending on software controls. It means to change functionalities by software. A challenge here is how to “hot swap” from the previous configuration to the new one. However, the combination of virtualization and reconfigurability enable hardware designs to remain competitive as long as possible, preserving owners investment.

Expose ability of hardware resources is also a pre-requisite. It means to be capable to expose hardware attributes and states to software, in order to provide adequate software control and orchestration. A challenge here is that hardware developers are not familiar with the technologies required for exposing hardware resources. One can not expect a computer hardware developer will know how to implement contextualized exposition of relevant information for software.

Finally, there is the manageability requirement. Since hardware functionality is going to be exposed and controlled by software, some hardware management functionalities are required. In the software layer, life-cycle management of virtual entities is a big challenge. It comprises search, selection, negotiation, admission, installation, configuration, monitoring, logging, and releasing of substract resources used by virtual entities. One can not expect the manual management of such army of entities. Therefore, technologies to reduce human intervention are a pre-requisite, e.g. autonomic or self-management approaches.

IV. CONCLUSIONS

The SDN paradigm deeply impacts on how communication networks are designed and implemented. At the same time it creates a new world of possibilities for network engineering, it breaks several well established paradigms related to: (i) networking design, implementation, control, and management; (ii) hardware development and testing; (iii) how components are structured in networking equipment; (iv) technology migration; and (v) networks performance, efficiency, and availability. More deep studies are required to precisely understand how SDN paradigm affects each of these aspects.

REFERENCES