Some Controversial Opinions on Software-Defined Data Plane Services

Fulvio Risso
Dept. of Control and Computer Engineering
Politecnico di Torino
Torino, Italy
fulvio.risso@polito.it

Antonio Manzalini
Telecom Italia Strategy
Future Centre
Torino, Italy
antonio.manzalini@telecomitalia.it

Mario Nemirovsky
ICREA Research Professor
Barcelona Supercomputing Center
Barcelona, Spain
mario.nemirovsky@bsc.es

***Abstract*—Several recent proposals, namely Software Defined Networks (SDN), Network Functions Virtualization (NFV) and Network Service Chaining (NSC), aim to transform the network into a programmable platform, focusing respectively on the control plane (SDN) and on the data plane (NFV/NSC). This paper sits on the same line of the NFV/NSC proposals but with a more long-term horizon, and it presents its considerations on some controversial aspects that arise when considering the programmability of the data plane. Particularly, this paper discusses the relevance of data plane vs control plane services, the importance of the hardware platform, and the necessity to standardize northbound and southbound interfaces in future software-defined data plane services.**

I. Introduction

The idea of transforming the network into a programmable platform is probably one of the hottest topics in the current research domain, which originates from the impossibility to deeply change the behavior of current network devices. In fact, in most cases only the manufacturer of the network equipment has the privilege to create the software that controls the device itself, while the possibilities for any other actor (e.g., a network operator) are more limited. In fact, a network operator can only *configure* the software already provided by the network manufacturer (changing parameters, choosing a routing protocol instead of another, etc.) but it cannot directly install *its own* software on the network device, such as a routing protocol customized for its particular environment.

Among the approaches that have been proposed so far toward a greater flexibility, we can cite Software-Defined Networks, Network Functions Virtualization [1] and Network Service Chaining [2]. Software-Defined Networks (SDN) represent a new architectural model in which the control plane is transformed into a programmable entity and is decoupled from the data plane. Instead, Network Functions Virtualization (NFV) and the IETF Network Service Chaining (NSC) proposals can be considered fairly orthogonal to SDN and aimed at simplifying the complex data plane processing path present in network operator's networks. Although a more in-depth discussion of SDN and NFV/NSC is left for Section II, we can summarize how SDN is more oriented to *control plane* programmability, while NFV/NSC focuses on *data plane* functions. Briefly, control plane refers to the set of functions that influence how packets are forwarded to the destination; for instance, the control plane faces the problem of *managing network paths* between source and destination hosts. Vice versa, data plane refers to the set of functions that can inspect, and potentially modify, the content of the packets in transit, i.e. it faces the problem of *processing each single packet*.

This paper discusses the problem of customizing data plane services with a view limited to a *single network device*, and assumes that future NFV/NSC solutions will allow users to deeply change the behavior of the data path of the network, e.g., by installing and running custom applications that operate on an arbitrary portion of the network traffic. In case of such of this event, we speculate that some assumptions that may be valid for the SDN world may no longer be appropriate when deep data plane programmability comes into play.

This paper presents the personal (and potentially controversial) opinion of the Authors on some issues related to the future software-defined data plane services, namely the importance of a programmable data vs control plane (Section III), the necessity of the network hardware to evolve (Section IV) and the necessity to define standard interfaces for future data plane services (Section V). The paper includes also an introduction to the existing SDN and NVF/NSC concepts (Section II), and a final section (Section VI) that summarizes current findings and presents some conclusive remarks.

II. Background

A. Software-Defined Networks

Software-Defined Networks are based on the separation between the *control* and the *data* plane of the network. The former, which is supposedly where most of the intelligence is, is transformed into an open programmable platform that can potentially host any network control application, usually provided by the network operator. This may allow different actors (e.g., network operators) to finely control the forwarding decisions taken in any portion of their network and implement the best traffic forwarding strategy according to their necessities. As a consequence, SDN can enable the implementation of smart algorithms e.g., to balance traffic across different links based on several criteria such as the sender/receiver of the traffic, the application, or anything else that is considered useful for the network operator.

Risso F., Manzalini A., Nemirovsky M., "Some Controversial Opinions on Software-Defined Data Plane Services," in Proceedings of the 2013 Software Defined Networks for Future Networks and Services (SDN4FNS), Trento, IT, November 2013, DOI 10.1109/SDN4FNS.2013.6702558.

SDN predicates that the control is a logically centralized function (although the implementations may be distributed) and it should be programmed through a set of well-defined interfaces, possibly standard, which could allow independent developers to execute their applications on different controllers.

While SDN represents definitely an important innovation in the networking area, it limits its scope to the control plane. In other words, it offers interesting possibilities to control the path traversed by a generic network flow, but it is not appropriate for data plane-oriented tasks that are very common in nowadays networks, such as many functions implemented by dedicated middleboxes (e.g., firewalls, network address translators, web caches, etc.) often placed at the edge of the network.

B. Network Functions Virtualization and Network Service Chaining

Network Functions Virtualization (NFV) focuses on the problem of consolidating and optimizing the processing of the network traffic that needs to traverse many middleboxes mentioned before, which is an increasingly important problem particularly in network operators' networks. In fact, new network services often require the traffic to traverse a large set of those boxes, each one implementing a specific function, with a huge impact in terms of costs (CAPEX, power, management, physical space), reliability and complexity of the network.

NFV is based on flow processing [3] and proposes to implement in software the network functions that today run on proprietary hardware, leveraging high-volume standard servers (e.g., Intel-based blades) and IT virtualization. This potentially enables greater flexibility and reduces costs, e.g., by consolidating several functions on a few physical servers. NFV can exploit control plane technologies such as OpenFlow [4] to dynamically reprogram the paths of network flows and allow them to traverse exactly the set of components (called *functions* or *applications*) that are needed for the selected service.

Network Service Chaining (NSC) is currently an unofficial IETF working group still in the embryonic Birds-of-a-Feather (BoF) discussion stage, whose vision looks similar to NFV. In fact, NSC goal is the standardization of the general architecture and the building blocks that are required to create network service chains, such as protocols (for setting up, configuring and managing the service chain), metadata (for passing additional information among different services) and more.

Although NFV/NSC are oriented to the data plane services, currently they do not enter into the detail of the implementation of each single function. For instance, a possible implementation consists in having applications running as virtual machine images, executed on hypervisors installed on standard servers. This allows to exploit the best of both worlds, i.e., servers execute applications, and network devices forward traffic, but that in this case data plane processing blocks are not *integrated* in the network device, but are located at its *border*, which may be a possible source of inefficiency. However, this possible implementation can be seen as a pragmatic move toward a (near) future with customized applications running on the data plane, enabled by an architectural model that leverages the highest possible number of existing components instead of starting the design from a blank new sheet. For instance, the reuse of existing components is one of the explicit goals of NSC. In essence, the architectural model proposed to implement the NFV/NSC in the near future is shown in the block in the middle of Figure 1: flexible data plane processing is not achieved by changing the architecture of the network device and integrating the new customized data plane applications in it; instead, service chains are created by establishing a closer collaboration between network equipment and mainstream computing platforms. Finally it is worth mentioning that the NFV/NSC architecture supports distributed processing chains, in which services are installed in different locations, even in a remote datacenter, and then chained in the proper order.

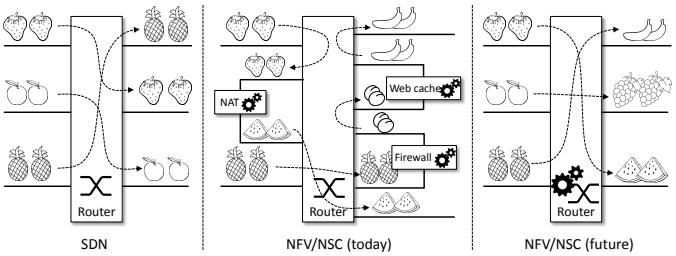

Fig. 1. An overview of the SDN and NFV/NSC (present and future) paradigms with respect to data plane functions, focusing on a single node.

C. A long term view on NFV/NSC

This section focuses on the implementation of the single functions defined in NFV/NSC on a long term horizon. Although predicting the future is always a difficult activity, we can foresee the presence of at least two fundamental differences from today. First, we foresee that future data plane architectures will spot a deeper integration between networking and general purpose processing components, possibly deeply integrated within the same network box[1]. Second, we foresee that new network devices will allow third parties (e.g., network operators) to install their own data plane applications directly on network device. This new software could either complement the standard data plane processing code provided by the device manufacturer with new functions or could replace it totally, and it will potentially operate on the all the packets flowing through the network device. In fact, several proposals in this direction have been made so far; among the others we can cite Click [5] and some recent papers such as [6], [7], [8], [9] and [10].

While everybody agrees that future network devices should become hybrid platforms that can execute both traditional networking tasks and new complex data plane processing applications, the hardware architecture needed for those new network

[1]Although many network manufacturers already offer network equipment that supports also general-purpose processing blades, currently those components are not properly integrated with the other data plane modules.

devices is still under debate. Options range from replacing the currently dedicated network hardware with standard high-volume components (e.g., clusters of servers equipped with mainstream CPUs) to the creation of hybrid devices that integrate both dedicated networking components (e.g., special purpose ASICs) and general purpose linecards.

Independently from the resulting hardware implementation of future network devices, the different approaches (control vs. data plane) taken by SDN/NFV/NSC may suggest the necessity of different hardware architectures, which are summarized in Figure 1. For instance, an SDN router may appear as a simple (dumb) switch (indeed, the intelligence is in the control plane), a NFV/NSC router may look like a (dumb) switch connecting several (smart) modules that operate on the data plane, while a future data plane router should resemble to a more integrated device, including switching and customized data plane processing functions in the same box. As a consequence, we foresee that future routers for data plane services will become smart devices with the capability to deeply modify the network packets, such as changing the value of some fields (e.g., in NAT applications) or even implementing complex data plane applications such as transparent web proxies, WAN accelerators, and more.

III. On the relevance of the data vs. control plane

SDN speculates that the network control plane is the place where the intelligence should be, while the data plane should look similar to a fast (but dumb) switch. As a consequence, SDN assumes that the control plane is much more relevant than the data plane because of the presence of more high-level functions.

While not questioning that the control plane is where most of the network intelligence would be, we would like to analyze the relevance of the data vs. control plane by looking at the point of view of two different entities, namely *network operators* and *end users*. With *network operators* we intend all the entities that are in charge of operating a network, such as companies whose business consists in selling network services (e.g., connectivity) to end users, companies that own a datacenter and sell added value services to third parties, and more. Vice versa, with *end users* we intend all the entities that buy network connectivity and, possibly, services from network operators, such as domestic ADSL users or corporate networks connected to the Internet.

This section makes use of three steps to present the personal view of the Authors on the relevance of the control vs. data plane. First, we analyze the relevance of the data vs. control plane from the point of view of the network operator. Second, we move our focus to end users and we discuss the possibility of giving them the privileges to to customize the behavior of the network (being it the control or the data plane). Finally, we discuss the relevance of the control vs. data plane focusing on the point of view of the end user.

A. The view from the network operator

The capability to customize the control path of the network represents a big value for a network operator, which can use this technology to optimize its infrastructure, create overlays or partitions in the network (e.g., Virtual Private Networks), avoid bottlenecks, and more. In essence, a clever control plane allows the network operator to improve its costs and to provide new services, although, from private conversations with some network operators, Authors infer that the first objective (i.e., reducing costs) looks more important than the second (i.e., providing new services).

Similar advantages can be achieved by a programmable data plane as well. The network provider can improve its operating cost by consolidating and optimizing many functions currently in use (firewalls, intrusion detection systems, network monitors, transparent web caches, etc.). Furthermore, it can introduce new services with shorter setup time and higher customization capabilities, such as parental controls, personal security software, network mobility solutions (e.g., LISP [11]), etc. For instance, a recent study [12] on the Italian market, which accounts 41.37M Internet users and 60.9M inhabitants at December 2012, evaluates the potential customers and the corresponding revenues coming from the five possible data plane services listed in Table I, all belonging to the security domain. The result is that those services alone have the potential to bring to the telecom operators more than 500M€, on a market that is estimated at about 60 billions euro/year. Although this preliminary number does not account the cost of delivering the service, nor the fact that a potential customer may decide not to buy the service, it shows that the introduction of a programmable data plane can offer to the telecom provider interesting opportunities for new added value services.

Although previous numbers should be considered as ball-park estimations[2], they seem to suggest that data plane customization may become an important source of revenues for the network operator, possibly originating a bigger economic impact than control plane customization.

B. User customization of the control/data plane

Historically, network manufactures looked suspiciously at the idea of allowing other actors (e.g., network operators) to create and install their own software on their network devices. In addition to the several reasons that come from the business side, we can cite the problem of guaranteeing the integrity of the "core" functions of the devices, which should run unaffected by the behavior (including bugs) of the external applications.

Nevertheless, due to market demand, some network equipment manufacturers recently opened (partially) their boxes to other actors, but, interesting, the previous problem did not

[2]For instance, in addition to the sources of uncertainty related to the economic impact of data plane programmability already mentioned in the text, other potential issues are the lack of estimates about the revenues that may come from a customized (programmable) control plane, and the cost reductions achievable through programmable control/data planes.

TABLE I
POSSIBLE YEARLY REVENUES FOR A TELECOM OPERATOR FROM NEW DATA PLANE SERVICES (ITALY ONLY)

Market description	Potential users	Pricing	Revenues
Customers (families) interested in a parental control software	1383180	30€/year	41.5M€
Customers interested in personal security software (e.g., personal firewall, etc)	18281920	15€/year	274.2M€
Corporate mobile users interested in mobile protection software	787913	50€/year	39.4M€
Corporate users that adopt the BYOD paradigm, interested in mobile protection software	2412964	50€/year	120.6M€
Companies interested in operator-based protection software (e.g., corporate firewall)	52414	1000€/year	52.4M€
Total			528.1M€

disappear. In fact, the ball is now in the hands of network operators, which prefer not to allow any other entity but themselves to install additional software on the network equipment.

In any case, the Authors believe that long-term plans for control/data plane programmability should allow the software *coming from arbitrary sources* (as proposed in [6]) to change the behavior of the network, either control or data plane, for the same reasons (both technical and economic) presented before. In fact, the future of control/data plane programmability should be the capability to offer to *several parties* (network providers, service providers, end users) the possibility to install and execute *their* software on the network and hence deeply influence its behavior. According to this model, user applications can target both control plane (e.g., changing forwarding paths) and data plane functions (e.g., drop e-mails with malicious attachments).

While the idea of allowing any entity to install its own application on the network may look dangerous because it may break the network itself, we can note that a similar behavior represents the common practice in the computing world, in which the entity that operates the hardware infrastructure (such as in public datacenters) may be different from who installs and operates the applications. In our opinion, it is just a matter of which *permissions* we grant to the users and how we enforce the *control* on their actions, not *if* they are allowed to do so. For example, we foresee that users could install applications such as personal firewalls in the network, operating only on the traffic generated by all the devices of the users itself, and more.

C. The view from the final end user

While the capability to customize the network paths can represent a value for the telecom operator, we believe that in most cases this represents an insignificant *detail* for a typical end user. In fact, end users usually expect their traffic to be delivered to the destination and they do not care about the path traversed by their packets.

Vice versa, they may be interested to install their applications on a programmable data plane, which enables them to relocate some existing functionalities (firewalls, intrusion detection systems, protocol translators, VPNs, etc.) in the network, and potentially add even new applications. From the point of view of the end user, there are two advantages in this model: (i) users have the freedom to install the data plane application they want, without being limited by the ones offered by their network operator, as in it would be in the scenario of Section III-A, and (ii) users do not have to deal with the hardware/deployment details required by those applications, which will be delegated to the network operator.

As a consequence, we believe that a programmable data plane is in general much more relevant than a programmable control plane and that the introduction of user-customizable capabilities in the data plane of the network would be noticed by the end user; hence it may represents an additional selling value for the network operator.

IV. ON THE IMPORTANCE OF THE HARDWARE IN FUTURE DATA PLANE SERVICES

A widespread opinion in the SDN world states that, as most of the intelligence resides in the control plane, the data plane will soon become a commodity, hence almost irrelevant for delivering future services.

However, if we move our focus to data plane services, this conclusion may no longer be valid because the levels of performance required for future data plane services will definitely be a challenge, not achievable with the hardware in use today. For example, telecom operators currently aggregate up to 10-20K ADSL users on a single network node that, taking into account a speed of 20Mbps per user, leads to an aggregated bandwidth of about 200Gbps. It is important to notice *(i)* that this number is expected to grow (more users could be aggregated in the future, and link speed could become larger), and *(ii)* that we are focusing on deeply programmable data plane applications, which may be required to perform custom processing (not just simple forwarding) on each single packet. This seems to suggest that the hardware may become less important for the general public, which will focus on high-value functions and care less about the underlying details (as today in the computing world), but it should definitely evolve in order to support future data plane services.

In fact, if we take a look at the world of computing, which shares many similarities with future data plane services, we can observe that the hardware has evolved considerably even in the last few years, although exploiting the advantages of the scale economies. For instance, thanks to the *standard high-volume* approach, we can use the same hardware in many different application fields (*standardization*) and we increase the number of devices that rely on the a few (sophisticated) components (*high volume*), reducing the overall cost of the system.

In fact, although most people think that current CPUs are just faster than some years ago, among the many innovations

we had in the general purpose processing hardware, we can cite virtualization primitives that allowed to execute virtual machines more efficiently, 64-bit instructions that enabled applications to deal with huge amount of data, efficient locking primitives for shared data; furthermore, dedicated accelerators such as graphics processing units (GPUs) and vector processors, are becoming increasingly common and are increasingly part of the capabilities of a general purpose CPU.

Similarly, we expect that the hardware needed to deliver future data plane services will be very different from today and that future network equipment need to evolve considerably. Particularly, if we assume that the end user can customize the data plane (Section III-B), we should expect a new breed of applications, possibly rather different from today network software, whose requirements could introduce additional pressure on the necessity to evolve the hardware. Probably, designing future network equipment would not be perceived as cool as creating fancy applications, but this would not mean that the hardware will become irrelevant.

V. On the standardization of Northbound and Southbound interfaces in future data plane services

The standardization of the northbound and southbound interfaces are hot topics in SDN and NSV/NSC. We define the northbound interface (NBI) as the set of APIs that allow developers to create software for a given platform. Vice versa, the southbound interface (SBI) is the set of APIs that allow that code to be instantiated on the physical hardware.

This section presents some thoughts about the standardization of NBI and SBI for future data plane service, introduced by the analysis of how a similar problem has been solved in the world of the general-purpose computing. Although the general purpose programming model may not fit perfectly the necessities of future data plane services, it represents a very strong candidate and, most likely, it could be the model that will be used first.

A. The programming model in general-purpose computing

The programming model in general-purpose computing, which is well understood and has been proved to be quite effective over the years, leverages the different layers and components shown in Figure 2. User *programs*, which implement the application-layer logic, are written in one of the many existing *languages* and make use of additional *libraries* that facilitate the implementation of specific functions. Those three components, with the help of the proper *compilers*, are used to create an executable that, leveraging the services (and, possibly, some virtualization functions) exported by the *operating system*, can run directly on the *hardware*.

If we want to introduce the NBI and SBI concepts in the general-purpose computing model, we could identify the NBI as the set of programming languages and libraries used by programmers, while the SBI is the interface represented by the hardware, which includes the CPU instruction set as one of the main components.

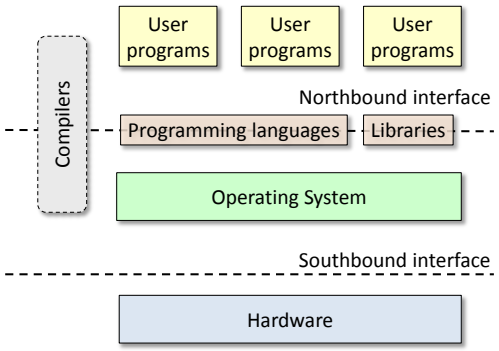

Fig. 2. The programming model used in modern computing environment.

The presence of a standard interface could guarantee many advantages, interoperability among the others, which is one of the reasons why the networking community is pushing for the standardization of NBI and SBI. However, we can note that in the computing world no standards exist for NBI and SBI and this did not prevent the creation of a very active software ecosystem.

Although the Authors believe that the standardization has many clear advantages, particularly in case of large networks where the heterogeneity of the hardware platforms is the common practice, we present in the following some motivations (both technical and economic) that suggest that NBI and SBI may not standardized in future data plane services, similarly to what happened in the general-purpose computing world[3].

B. Northbound interface

The northbound interface should be driven by the applications. Back to the computing world, we invented many programming paradigms (procedural languages, event-driven languages, functional languages, etc.) and many languages (e.g., Fortran, C, C++, Java, Python, SQL, etc.), each one probably representing the best choice in a given condition, such as resource-constrained environments, client-server programming, artificial intelligence applications, etc.. Moreover, since applications (and business necessities) evolve over time, also languages and paradigms evolved, hence new languages were defined, other were abandoned, other where enriched with new functions, and more.

Similarly to what happened in the computing world, we expect not only that the NBI should evolve over time, but that there should be many "northbound interfaces", each one being the most appropriate for a different business need. On the other way, we believe that the standardization of the NBI could limit the evolution of data plane services and prevent people to exercise their creativity and envision new applications. For instance, the lack of a standardized NBI (and the freedom to invent new "interfaces" when needed) was probably one of the key that enabled the continuous evolution that characterized the computing world.

[3]In fact, another possible outcome is that a standard will be defined but it will not be used in practice, which happened many times in the past.

C. Southbound interface

The southbound interface, being more close to the hardware, is probably less "cool" than the northbound but nevertheless it represents the "workhorse" that allows high-level data plane applications to work.

In the ideal world, a standardized SBI could allow the same data plane application to be executed on any network device. However, the computing world does not have a standard interface here, as each hardware platform has its own characteristics (e.g., CPU instruction set) in addition to the different API exported by the operating system. Nevertheless, we have several options to create portable software, such as source code recompilation, "cross-platform" languages (e.g., Java), interpreted languages, etc. The experience of the computing world suggests that, while a standard SBI interface would be useful for application portability, in practice the same result can be achieved with other technologies. Although the problem of code portability with respect to data plane functions seems to be more difficult to solve because of the heterogeneity of the networking platforms, we have no evidence that the same solutions that work in the computer world would not apply to the networking world as well.

On the other side, there are at least two reasons against the standardization of the SBI. First, from the business point of view, a standard SBI could be against the interest of the (major) network manufacturers, which would lose the possibility to differentiate their products from competitors. Second, on the technical side, a standard SBI could limit the degree of innovation that we can introduce in the hardware/operating system. For instance, the Authors believe not only that the hardware should evolve with the new additional primitives needed to offer a better support to future data plane applications (as presented in Section IV), but that the same applies to the operating system that supervises the network box. This could be achieved much more efficiently if several manufacturers compete to improve their products and are not tied to a fixed and the-same-for-all interface.

D. The role of Openflow in the future SBI

The OpenFlow protocol has been proposed as a possible southbound interface by the SDN community, hence we may wonder if it may be an option for data plane services as well. This section motivates why OpenFlow is not appropriate in case we would like to define a SBI for data plane services.

OpenFlow was defined by the networking community in order to increase the flexibility of current networks. The networking community focuses on the whole *network*[4] and it consider a network device as a black box that takes packets from the input ports and forwards them to the most appropriate output port. According to this network-centered view, a network switch could be modeled by a simple lookup table that, in fact, is what OpenFlow does: a network device becomes a lookup table[5] that can be reprogrammed by an external software, hence achieving very flexible forwarding policies.

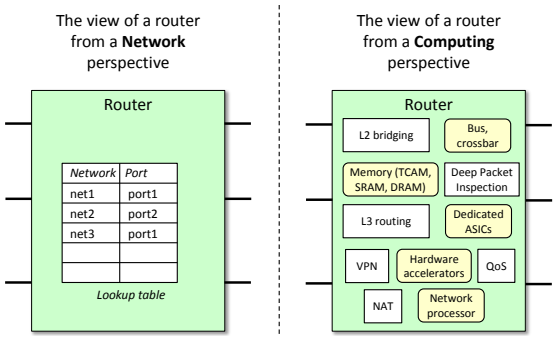

Fig. 3. Different views of a router.

However, if we change our perspective to the one of a computer engineer, the view of the network device would be tremendously different, as depicted in the right side of Figure 3. For instance, the router would be a collection of functions (L2 bridging, L3 forwarding, VPN concentrator, NAT, QoS, etc.) and of functional blocks (CPU, network processors, memories, hardware coprocessors, etc.). When looking from this angle, not only a network device requires a model that is more complex than a simple lookup table, but deriving a suitable model may not be straightforward at all. In essence, we cannot define a standard SBI until we have a suitable model of the network hardware, but the definition of a comprehensive model that satisfies the necessities of data plane applications is still a rather unexplored research topic and no clear solution is visible on the horizon.

As a consequence, the OpenFlow protocol is not a suitable SBI for data plane applications, as it would not be able to exploit the (required) hardware features present in data plane-oriented network devices, although it may be sufficient in some environments where network devices are requested mainly to forward traffic, such as the case of datacenter switches.

VI. Conclusion

This paper presents the opinion of the Authors on some (controversial) aspects that, when examined with a data plane perspective, may look different compared to the the view from the control plane world. Particularly, this paper focuses on three aspects.

First, it argues that, although the control plane received much more attention in the past, a programmable data plane may be more valuable than the control plane because it is more visible by the end users, which may be willing to pay for the possibility to customize its services. Second, it suggests that building the hardware that is needed to provide future data plane services is definitely a challenge, which looks the opposite compared to the vision suggested by the SDN paladins, i.e., that future networking gear will become

[4]Although this statement looks obvious, please do not underestimate its importance, which will be clarified in the following.

[5]In fact, more recent versions of OpenFlow (*gt* 1.0) model a router with a set of lookup tables. While this may be more appropriate for some cases, still does not capture the complexity of a modern router.

commodity. Third, it suggests that the standardization of the northbound and southbound interfaces, which is receiving a great attention in the SDN/NFV/NSC worlds, may not be successful for both business and technical reasons.

Although the Authors agree that some aspects presented in this paper may be controversial, they hope that that their personal opinions will be useful to foster the discussion on programmable data plane issues in future networks.

ACKNOWLEDGMENT

The authors would like to thank the many friends that participated to this discussion; among the other we would like to mention Marco De Benedetto (Embrane) and Pere Monclus (PLUMgrid). We would also like to thank Alessandra Colombelli and Rafael Scaglia De Paula who investigated the business side of future data plane services.

FINAL NOTE

This paper is slightly different from the one submitted to the conference and published by IEEE. This version includes some modifications that take into account the discussion during the presentation of the paper at the SDN4FNS conference. The authors would like to thank all the SDN4FNS participants for the very inspiring discussion.

REFERENCES

[1] "Network functions virtualisation," Introductory White Paper, Oct. 2012, work in progress. [Online]. Available: http://www.tid.es/es/Documents/NFV_White_PaperV2.pdf

[2] P. Quinn, J. Guichard, S. Kumar, P. Agarwal, R. Manur, A. Chauhan, N. Leymann, M. Boucadair, C. Jacquenet, M. Smith, N. Yadav, T. Nadeau, K. Gray, B. McConnell, and K. Glavin, "Network service chaining problem statement," Internet Engineering Task Force, Internet-Draft draft-quinn-nsc-problem-statement-03, Aug 2013, work in progress. [Online]. Available: http://tools.ietf.org/html/draft-quinn-nsc-problem-statement-03

[3] A. Greenhalgh, F. Huici, M. Hoerdt, P. Papadimitriou, M. Handley, and L. Mathy, "Flow processing and the rise of commodity network hardware," *SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 2, pp. 20–26, Mar. 2009.

[4] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: enabling innovation in campus networks," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, Mar. 2008.

[5] R. Morris, E. Kohler, J. Jannotti, and M. F. Kaashoek, "The click modular router," in *Proceedings of the seventeenth ACM symposium on Operating systems principles*, ser. SOSP '99. New York, NY, USA: ACM, 1999, pp. 217–231.

[6] F. Risso and I. Cerrato, "Customizing data-plane processing in edge routers," in *Proceedings of the European Workshop on Software Defined Networking (EWSDN)*, 2012, pp. 114–120.

[7] J. Martinsy, M. Ahmed, C. Raiciuz, and F. Huici, "Enabling fast, dynamic network processing with clickos," in *ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking (HotSDN)*, 2013.

[8] G. Lu, C. Guo, Y. Li, Z. Zhou, T. Yuan, H. Wu, Y. Xiong, R. Gao, and Y. Zhang, "Serverswitch: a programmable and high performance platform for data center networks," in *Proceedings of the 8th USENIX conference on Networked systems design and implementation*, ser. NSDI'11. Berkeley, CA, USA: USENIX Association, 2011, pp. 2–2.

[9] J. Whiteaker, F. Schneider, R. Teixeira, C. Diot, A. Soule, F. Picconi, and M. May, "Expanding home services with advanced gateways," *SIGCOMM Comput. Commun. Rev.*, vol. 42, no. 5, pp. 37–43, Sep. 2012.

[10] J. W. Anderson, R. Braud, R. Kapoor, G. Porter, and A. Vahdat, "xomb: extensible open middleboxes with commodity servers," in *Proceedings of the eighth ACM/IEEE symposium on Architectures for networking and communications systems*, ser. ANCS '12. New York, NY, USA: ACM, 2012, pp. 49–60.

[11] D. Farinacci, V. Fuller, D. Meyer, and D. Lewis, "Rfc 6830: The locator/id separation protocol (lisp)," Jan 2013.

[12] R. S. D. Paula, "Market analysis for programmable router," Politecnico di Torino, Torino, Italy, Tech. Rep. MSc thesis, July 2013.