Network Management Challenges in Software-Defined Networks

Slawomir Kuklinski, Prosper Chemouil

▶ **To cite this version:**

Slawomir Kuklinski, Prosper Chemouil. Network Management Challenges in Software-Defined Networks. IEICE Transactions on Communications, 2014, E97.B (1), pp.2-9. 10.1587/transcom.E97.B.2 . hal-02840942

HAL Id: hal-02840942
https://hal.science/hal-02840942v1

Submitted on 13 Jul 2020

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

IEICE TRANS. COMMUN., VOL.E97–B, NO.1 JANUARY 2014

INVITED PAPER *Special Section on Management for Flexible ICT Systems and Services*

Network Management Challenges in Software-Defined Networks

Sławomir KUKLIŃSKI[†,††] ***and*** **Prosper CHEMOUIL**[†††a)]**,** ***Nonmembers***

SUMMARY Software-Defined Networking currently appears to be a major evolution towards network programmability. In this paper, we first analyze the network management capabilities of OpenFlow in order to identify the main challenges that are raised for SDN management. We address current deficiencies of SDN management and suggest solutions that incur research directions to be carried out for the management of enhanced SDN.

key words: *Software-Defined Networking, network management, autonomic network management, Future Networks*

1. Introduction

The Software-Defined Networking (SDN) concept has recently gained enormous popularity and it is perceived as one of the largest innovations in networking since the introduction of MPLS, which was developed about 15 years ago [1]. In fact, SDN has emerged as an approach to foster network innovation through enhanced flexibility, programmability, manageability and cost-effectiveness. It relies on several networking and IT ingredients which allow network providers to tailor service provisioning and chaining, and network resource allocation in the most efficient way. Such ingredients include flow-level manipulation, clear separation of control and data planes, open interfaces and data models as well as virtualization techniques. As such, SDN represents a significant step towards programmable packet networking. Still, there is an ongoing debate about the SDN architecture and its scope, and while the SDN acronym can be used for many programmable networks, the main SDN direction is still linked with the OpenFlow protocol that has been standardized by the Open Networking Foundation [2], [3]. The main feature of OpenFlow is programmability of selected network operations. These operations are mostly related to packet forwarding at flow level; however their potential range of applications is much broader.

As of today, the concept has limited management capabilities and completely ignores the latest advances in network management such as autonomic and cognitive networking concepts, though it tries to resolve some management-related problems (i.e. self-healing, network performance optimization, etc.). Basically, it is often admitted that OpenFlow operations could be complementary to classical management systems; however there are also other possibilities for incorporating management features in SDN networks.

In this paper, we raise the importance of management issues in SDN networks. We address the classical network management functions as well as SDN intrinsic management. In Sect. 2 we briefly present the main features of OpenFlow in the light of the traditional networking framework. In Sect. 3, we highlight some limitations of the OpenFlow model and we emphasize the need of extending the existing SDN concepts by management-related functions. In Sect. 4, we present main research directions in network management and some specific issues related to management of SDN are raised in Sect. 5. Finally, several variants of incorporating network management features in future SDN are presented in Sect. 6.

2. OpenFlow Features

OpenFlow has been developed by proposing an organic separation between control and data forwarding through an open interface to populate the forwarding information bases of switches. Network programmability is not a new concept and more than 10 years ago, significant research related to programmable [4] and active networking [5]–[7] has been carried out. Unfortunately, at that time, despite relatively high maturity, these concepts had gained marginal popularity and were not recognized by the networking industry as promising approaches. From 2004, the IETF Working Group ForCES — Forwarding and Control Element Separation — had initiated some work on such an issue and has produced many RFCs [8]. ForCES has hardly emerged and has not been adopted so far, though some aspects had been deeply analyzed.

In contrast, OpenFlow has been widely adopted by industry as a major enabler for network programmability. This approach is quite different from the older, above-mentioned concepts. In opposite to them, it can be seen as a simple bottom-up approach that can be easy deployed now, at least in a small scale. In fact, OpenFlow can be perceived as a kind of extension to the well-known Linux operating system firewall mechanisms, like *iptables* and *ipchains* [9]. The important feature of this approach is the identification of packet streams (i.e. flows) that have to be handled in the same way. The packet filtering rules are programmed in a

Manuscript received June 11, 2013.
Manuscript revised July 31, 2013.

[†]The author is with Orange Labs Poland, Poland.
[††]The author is also with Warsaw University of Technology, Warsaw, Poland.
[†††]The author is with Orange Labs Networks, Issy-Les-Moulineaux, France.
a) E-mail: prosper.chemouil@orange.com

DOI: 10.1587/transcom.E97.B.2

Copyright © 2014 The Institute of Electronics, Information and Communication Engineers

way that is already used by many operators for so-called Deep Packet Inspection (DPI), which main applications are network security and traffic engineering [10]. In OpenFlow the mentioned mechanisms have been reused and some new ones, taken directly from Linux networking mechanisms, have been added.

The main innovation introduced by ONF is the definition of the OpenFlow protocol that allows a computer (aka the OpenFlow controller) to control most of the networking operations, especially the forwarding rules, of other nodes named OpenFlow switches. The controller plays the master role in the master-slave model whereas the OpenFlow flow-level switches are the slaves. Their relatively simple functionalities include the identification the data flows and switching of flows according to the decisions taken by the controller. These decisions are disseminated using a relatively simple and light protocol. The controller sets individual forwarding rules for each identified flow or group of flows for all the switches in the network. Additionally, it can remove or clone the flows and manage flow tables and queues of the switches. It also has the ability to monitor the state of the switches. For example, it can read the status and the load of switch interfaces (ports). Due to such properties, adaptive operations and detection of some faults are possible. Resulting from the centralized architecture which makes it quite different from all other well-known data networking approaches, the switches do not take locally any decision regarding flows. It is worth to emphasize that flow-based approaches have been introduced from about a decade and include theoretical works [11] as well as implementations of flow-based networking [12], [13]. Also, widely popular monitoring tools for IP networks, like Netflow or IPFIX [14], are also flow-based. Despite its simplicity, the described approach brings a significant change into the data networking in many dimensions.

First, in opposite to packet-based routing, it introduces the notion of flow switching. The data flows can be considered as sessions that are not announced a priori by the user. The packet header inspection mechanism eliminates the need of signaling for such sessions, therefore the approach requires no change in existing applications and no end-to-end signaling protocol is required for proper operation of the OpenFlow-based network. Flow session durations can vary significantly, but in most cases, flows stay long enough to provide them individual way of switching.

Second, the introduction of a controller brings back the decoupling paradigm of the network control plane from the network forwarding (data) plane. Such separation has been well-known in telecommunication networks since more than two decades. The control plane is realized through the OpenFlow controller while the data plane is composed of connections between the switches. At present, the control plane is centralized and only one controller in the network may exist, though the introduction of multiple controllers is now under consideration as a way to ensure reliability and scalability. The supporters of OpenFlow often claim the benefits of a centralized approach. By providing network abstraction, it gives new abilities that were unavailable in classical, distributed routing-based networks such as centralized traffic engineering. In addition, the controller can use a generic computer platform and due to the relative simplicity of switches, their cost should then be lower than of current routers. Therefore, a new network based on OpenFlow should be cheaper and more flexible than the classical IP network.

Third, OpenFlow can be used for relatively rapid and simple creation of virtual networks [15], which importance is currently growing fast.

Fourth, OpenFlow offers the ability to program functions at a level that has not been achievable in IP networks. Thus the approach is open and the network owner or the operator can define the way in which each flow should be handled. It allows for high level of personalization of services offered to the end-users.

Finally, the approach has gained significant interest in the research community. Using OpenFlow, it is easy to create experimental networks in order to develop and test advanced networking mechanisms. Such experimental solutions can be easily converted into a commercial network. From a research perspective, this is a completely new situation in a comparison to IP networks. In the latter case, the scientists had only possibility to simulate new networking mechanisms and their deployment depends on vendors' strategy and standardization. It is worth to mention that the dominant trend now is to use GNU Public License (GPL) controller platforms [16]–[18].

While initially created for campus networks, OpenFlow is currently targeting data centre use cases, as SDN benefits could be more easily assessed in a "contained" environment under the control of network providers than in carrier networks. As of today, there are many experimental networks based on OpenFlow, and there is at least one commercial network [19]. Most of the networking equipment vendors offer OpenFlow-enabled devices (switches or controllers). Some of these solutions are solely OpenFlow-based whereas other can be seen as a double function solutions, i.e. they work as classical IP routers, but also support OpenFlow operations [20], [21].

OpenFlow is neither seen nor can be seen as the ultimate solution for SDN. As already emphasized, this is a bottom-up approach and a more systematic work on SDN has already been initiated. The ongoing works deal first with the analysis of the capabilities of SDN, overcoming some OpenFlow drawbacks, taking into account requirements related to large, commercial deployments, and, second, with the standardization of a more generic solution. Whereas the original OpenFlow specification has been developed by ONF, other standardization bodies like ITU-T [22], IETF [23] and IRTF [24] have also started working on SDN recently. It seems that there is a consensus that OpenFlow is a first and important step towards Future Networks, but there are still many issues that have to be solved in order to provide network operator-grade solutions. There is an important question whether these issues can be solved by

the OpenFlow evolution or if a completely new protocol or even set of protocols is required to cope with all required functionalities.

3. OpenFlow in the Context of SDN Evolution

The benefits of OpenFlow, mentioned in the previous section, do not mean that the concept is neither complete nor free of problematic issues. In this section we point out some problems, mostly those which have a potential impact on the network management.

One of the main functional issues is the lack of interfaces to the applications (so-called the Northbound interfaces). Such standardized interfaces would enable higher personalization of network services and a better way of interaction with applications and service providers. Additionally, they would provide a demarcation line between basic and advanced network services.

Another drawback of OpenFlow is related to the centralization of network control. There are obvious benefits of such an approach, which include a global view of the network state in one network controller and lower cost of the switches. Drawbacks are merely related to network reliability as the controller appears to be a single point of failure, and scalability which raises potential performance issues of the control plane in terms of reaction time or volume of control traffic. The centralization issue could be at least partially solved by splitting the network into smaller domains, but so far there is neither multi-controller architecture nor inter-controller interfaces (so-called Eastbound-Westbound interfaces) defined.

In OpenFlow-based networks, the controller is programmable and there is thus a need to manage the controller programmability. Such functionality requires on-the-fly software update capability and the ability to update or stop some controller functions from a remote console. From a network and service management perspective, a standardized controller platform providing an execution environment used to install OpenFlow applications would help in the controller management.

In general, it may be unrealistic and against open networking to a priori define the full controller functionalities. It is widely accepted that such functionalities should be incrementally enriched. Unfortunately, after adding new functions to the existing ones, the controller may exhibit unstable or chaotic behavior. This problem is well-known in 3GPP SON [25] and it is solved with so-called coordination of SON functions. In order to alleviate it, the behavior of the network should be observed and appropriate actions can be taken. It appears that such observation and coordination belongs to the SDN management.

In current SDN approaches, the usage of classical, legacy network management systems combined with OpenFlow is often assumed. The analysis of the OpenFlow protocol leads to the conclusion that it lacks primitives that are able to cope with network management operations.

Last, no controller or switches intrinsic management functions are currently defined and no controller management interface is yet proposed. The openness of OpenFlow enables the implementation of some network management functions, but the lack of standardization of a management interface makes impossible to use third-part management solutions in a way as they are used in classical management systems.

The above-mentioned OpenFlow drawbacks show that there are many open issues related to large scale deployments of OpenFlow or SDN-based networks and that a significant evolution of the concept is needed. Such evolution should not ignore the main advantages of OpenFlow presented in Sect. 1, but it should at least partly reuse some ideas from other research domains, especially the latest network management concepts as well as the possibility of integration with the existing network management systems.

4. Latest Trends in Network Management

Network management functions were defined long time ago by the TeleManagement Forum and ITU-T [26], [27]. Functionalities of classical management systems typically include so-called FCAPS functions, i.e. fault management, configuration management, accounting management, performance management and security management. Classical management solutions are typically centralized ones and their high complexity is well known. For management purposes, specific management interfaces have then been developed. In current network operations, the role of a human operator is still significant. It is commonly agreed that in order to cope with future demands, especially with the ever-growing number of managed devices and services, the network management paradigm has to be changed. A widely accepted direction lies on the automation of network management that will finally lead to a plug-and-play management solution. Due to such evolution, network availability is expected to be very much improved and OPEX related to the network management should be significantly reduced.

There is also a consensus in the community that in order to achieve such goals, management decisions should be distributed, and the management system should react quickly; hence the human role in management has to be reduced. The distribution of management decisions helps in providing of quick and local management operations while reducing the volume of the management traffic, and it further increases the reliability of the management plane. In-depth analysis has shown that management cannot be fully decentralized and there are still some operations such as network topological properties, operator's preferences handling that can be efficiently implemented in a centralized manner. So, it appears that an adequate management system should be a hybrid one, i.e., it has to combine both centralized and decentralized management operations. This is a conclusion of many research projects related to the management of Future Networks that were recently carried out worldwide. In this context, it is worth to mention several European FP7 projects [28]–[31], ETSI activities on

Autonomic Future Internet [32] and 3GPP works on Self-Organizing Networks (SON) [25], which all deal with new network management concepts. Most of them are labeled as autonomic network management. This type of management lies on performing in "real-time" or "near real-time" selected management operations in a fully automated manner in order to provide so-called self-x properties, namely: self-optimization, self-healing and self-configuration of network nodes. Such autonomic concepts can be also reused for the management of services or for joint management operations. All present autonomic management concepts are based on the MAPE paradigm [33]. In this approach, management decisions are taken according to some monitoring data extracted from the network. As a result of these decisions, the network state changes, and the network state is of course sensed again. This behavior describes a classical control system with a feedback, often referred as loop-based. The problem with such an approach lies in the interaction of multiple loops that can lead to potential instabilities. This issue can be somehow resolved by coordination functionalities as proposed in [28].

The autonomic approach to network management has made significant progress in terms of algorithmic solutions and standardization with the definition of reference architecture [32], but there is still lack of commercial deployments. As already mentioned, the most advanced approach is 3GPP SON. Unfortunately, it is applicable only to the radio access network part of the mobile systems that are defined by 3GPP. Moreover the solutions provided so far are not programmable, they remain vendor-specific.

There is then an open issue on how the experience taken from the autonomic management paradigms can be reused in SDN and how present SDN concepts can be integrated with autonomic management solutions.

5. Management Issues in SDN

The SDN approach comes with the promise of programmable functionality and the low cost of devices, mostly due to the expected lower cost of SDN switches. Such an approach may certainly reduce CAPEX, however in case of network management, OPEX is the main driver. As it was previously emphasized, the latest trends in network management leads to network automation which reduces role of human operator and which provides fast reaction to events which finally results in plug-and-play solutions.

Meanwhile, the management issues in SDN-based solutions have been relatively ignored so far. This has been justified in the beginning by the experimental status of OpenFlow-based networks, which aimed at assessing the approach. Setting management capabilities to add new functionalities, to provide highly-reliable operations and to scale with network extension is however of paramount importance when considering large-scale commercial deployments of SDN. The well-known complexity of network management calls for some standardization of the management components in order to decompose the management system and to give room for third-party management sub-systems, as it is currently available in commercial networks. Typically, in such networks, the operators interact with the network via the management system. After network deployment, the most typical network operator's actions are usually related to network reconfiguration, as a result of network enlargement or topology changes. Network performance optimization is typically carried out via policy mechanisms, which can be seen as indirect mechanisms that influences the network behavior. In SDN-based networks, the situation appears to be significantly different as the operator has the ability to change network operations in a direct way, due to the programmability of the control plane. He can thus design the algorithms that will be used for management or control operations. There are many operations in which strong interactions between the management system and OpenFlow are then required. Examples are traffic rerouting, dynamic resource provisioning or even energy-efficiency solutions, in which unused or lightly loaded nodes or ports of switches can be switched off.

The management functionalities of SDN can be generally divided into two parts. The first part is related to classical management operations, i.e. the already-mentioned FCAPS functionalities. The second part is related to SDN-specific management that copes with the problem of controller and switches management. The first part of the management operations (FCAPS) is now subject of intensive research, but it seems that the second part is almost neglected. One notable exception is the development by ONF of the OF-Config protocol [34]. This protocol is used for configuration of links between the OpenFlow switches and the controller. In the short term, the SDN management operations could be supported by the existing network management protocols (SNMP, NETCONF). There are however many limitations of such an approach: lack of automation of the management procedures, performance and functional limits of the existing protocols to support real-time management based on the loop paradigm.

The management framework for SDN can reuse some well-known network management concepts and it should incorporate as well the latest trends in network and service management.

The functionalities of the SDN management have to incorporate FCAPS functions, but the notion of these functions in SDN must be extended in comparison to classical networks. For example, the SDN approach should support dynamic switching off some unused resources (e.g. ports, switches) in order to implement energy efficient operations. Unfortunately, the OpenFlow protocol has no such direct support, however they can be provided in combination with the classical network management systems.

There are five main new management issues that appear to be specific to SDN only.

- A first issue is imposed by the critical role of the controller, which has to perform most of its operations in a real-time regime. It means that the performance of

the controller has to be monitored, and the controller processes should be categorized and handled according to their real-time requirements [35]. Moreover, the controller resources including the storage and the execution environment have to be monitored and managed. A potential overload of the controller needs to be anticipated and handled. Appropriate reports should be then periodically generated and stored. In case of controller failure, a hot swap operation is necessary.

- A second SDN management issue is related to the programmability of the controller. This important SDN feature requires special care. In order to fully use such capability, the network operator should be able to remotely add new controller functions and to update the existing ones. Such functionality means on-the-fly reprogrammability of the controller. In order to perform such operations, the operator should have a repository of controller's functions that could be installed in the controller, a mechanism for secure transfer of the code that will be used by the controller and the ability to manage the controller software modules. The management operations should be able to start, stop and monitor the module status. An example of such functionality is provided by OSGi [38]. There are many important reasons that call for the standardization of the controller platform. In a standardized case, the software modules and the way which they are managed could be reusable. Last but not least, the security of the controller platforms can be also provided that way.
- The third management issue is related to the detection of abnormal behavior of the controller due to unexpected interactions of some of the controller modules which indirectly disturb each other, resulting in system performance degradation or instability. This problem can be solved by additional modules or software components to the controller. It is beneficial, however, to define additional monitoring and conflict resolution functionalities for management. All conflicting or problematic situations have then to be reported to the management system for diagnosis purposes.
- The fourth issue is related to the security of SDN control plane and the authentication of newly added switches. It has to be combined with the security of the management platform.
- Finally, that the last issue is related to the determination of the demarcation line between the control plane and the management plane in SDN. Such a problem has been already identified in some latest network management research. In these approaches, some management operations are real-time or near real-time. Therefore they can be also classified as control plane operations. There is not yet consensus on how the problem should be solved since, at present, the OpenFlow control plane supports also such management functions, for instance self-healing procedures.

To summarize, the management issues in SDN lie first in the way the separation between the control and the management functions is performed. Then, there is a need to add a new set of intrinsic procedures to SDN management functions in order to achieve classical or autonomic management functionalities. Finally, a border line between the SDN network operations and SDN-based applications has to be set. Such a separation can help in the definition of the management system role.

There is also an issue whether virtual networks can be seen as applications and how to define Network as a Service (NaaS) and Network Functions Virtualization (NFV) functions in the SDN case [39]. Indeed, the current activity on NFV carried out at ETSI may raise similar concerns regarding management of applications, and proper interaction between SDN and NFV should be analyzed. In this context, the collaboration between the OpenFlow/SDN controllers and cloud computing orchestrators must hence be thoroughly analyzed, so as to leverage existing solutions.

SDN architecture may certainly have an impact on the management functions. For example, in a solution with multiple controllers, mechanisms of hot swappable controllers with load balancing should play an important role. Typically, since fast management reactions call for distributed and embedded intelligence, all SDN switches should not only detect some events, but also take appropriate actions. Local actuation is faster, more scalable and such a system is more reliable as it eliminates single point of failure. The mentioned benefits are balanced by the price of additional complexity and increased cost of the SDN switches. Consequently, the most efficient management approach seems to be a hybrid, hierarchical one in which the network, for the management purposes, can be split into smaller geographical domains that are locally managed and coordinated by a network management center.

The above-presented discussion leads to the conclusion that there are several variants in which SDN network management can be implemented. These variants are described in more details in the next section.

6. Variants of SDN Management

The first variant of SDN network management lies in enriching the controller with some network management functions (see Fig. 1.). This approach, which is a low cost solution, provides a high level of integration of control and management functions within the SDN controller and it can be used right now. Unfortunately, it comes with some drawbacks.

The first drawback is related to the limited management capabilities of the OpenFlow protocol. This issue can be resolved by the reuse of existing management protocols, for example by the use of SNMP. Though such protocols have to be handled by the SDN controller, this approach has now become pretty popular [17]. In some cases, equipment vendors simply add OpenFlow capabilities to the existing IP routers [20], [21] which makes such an approach natural. However this approach does not solve the problem of the management of the controller platform itself. As explained

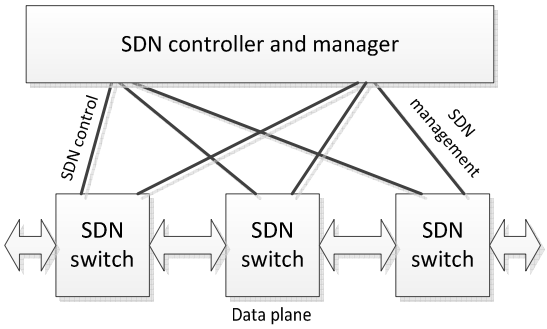

Fig. 1 Integrated SDN control and management.

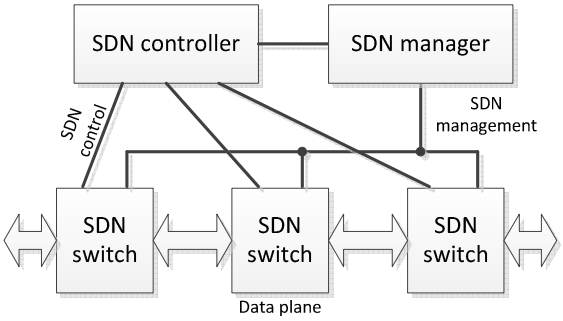

Fig. 3 Separated SDN control and management platforms, distributed management.

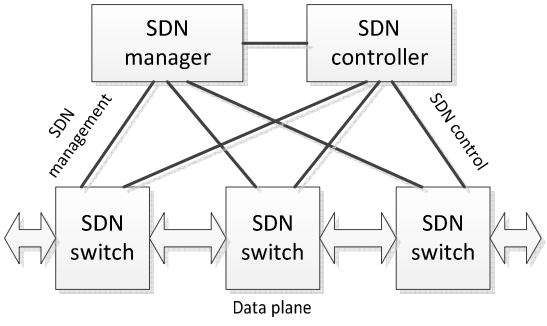

Fig. 2 Separated SDN control and management.

in the previous section, the controller platform has also to be monitored and the best way is to do that externally. In the described approach, in case of a failure of the controller, all the management capabilities would be then lost. Moreover, adding the management functions to the controller would significantly increase the controller software complexity.

A second variant of the SDN management can be seen as the classical management approach applied to SDN (see Fig. 2). In this approach, all network devices (switches, controller or controllers) have a management interface that is used for the connection to a centralized management platform. In this approach all the well-known management protocols are re-used as well as software management platforms. The reusability of the existing platforms is the strongest value of this approach. Moreover, in comparison to the previous approach, this one enables external monitoring and management of the controller(s). The main issue here lies in a lack of well-defined communication procedures between the SDN management plane and the control plane and such functionalities have yet to be defined. There are neither management interfaces nor protocols of the switches or the controllers defined, except of the OF-Config protocol, which is used for the configuration of the OpenFlow control plane links. As in the previous variant, another issue is the limited capability of the management protocols for the implementation of advanced real-time management functions.

The cost of this approach is slightly higher than the previous one, but for network operators the value lies in the reuse of some already deployed management platforms, which anyway have to be properly adapted.

The third variant lies in an implementation of the latest autonomic/cognitive network management paradigms (see Fig. 3). The main difference here lies in the distribution of the management functions — all network devices, including switches, have the ability to take some management decisions and they may directly cooperate in order to achieve some management goals.

In this approach, there is a need to exchange management messages not only between the controller and the switches, but also between the switches and an abstract Autonomic Network Management protocol (ANM protocol) must be used for the purpose. Despite the distribution of management intelligence, this approach cannot be fully distributed. In this management architecture, there should be still some nodes that could play the role of a domain or network managers. So, probably we could have a hierarchy of the management nodes similar to that proposed in [36]. As in the previous variants, such an approach can be implemented in the controller platform or in a separate management one. The “central” nodes in this approach have lower management information processing load, in comparison to the two previous variants, because some management functions are handled locally. The benefits of this approach include high reliability of the management plane, fast reaction time to events, and low complexity of the central management platform. In contrast, this also implies higher price of switches and issues related to the deployment of distributed algorithms. An important question still is related to the proper definition of the management functionalities of the switches. Such functionalities can be predefined or have the ability to be programmed and uploaded. The first case can lead to a problematic and long standardization process. The second one means some standardized solution on the execution environment that could be used to implement programmable management on board of SDN switches. In fact, a distributed execution environment would be needed in such a case. The management functions should be then enhanced with functionalities related to the maintenance of the management code distribution among the switches and its execution control in a similar way to that of the management code of the controller. There are some software technologies that can be reused for the implementation of such

an approach, for example distributed agent-based solution FIPA [37] that can be combined with the OSGi [38].

The last variant seems to be the most open one. In such an approach, there is no need for a strict definition of the management interfaces, as they can be defined during the implementation. The drawback of this approach is a higher complexity of switches due to their embedded intelligence and possibly limited performance of the switch for the management purposes (e.g. limited memory and the processing power). As previously mentioned, a separate management platform provides some evident benefits, giving the ability to supervise the controllers and to enable a functional decomposition of SDN. The information exchange between the controllers and the management system has yet to be provided, and can be seen as a cross-layer information exchange. The issue of the separation of the management and control functions between the platforms is raised again, however we see unloading of the controller as a positive value.

7. Conclusions

In this paper, management challenges in SDN have been presented. It has been observed that the management issues in SDN are generally ignored, despite the fact that some management functions are already performed by the SDN controller. In light of the future evolution of SDN, there is a need to put strong emphasis on the management systems in order to envision large scale, commercial SDN deployments. As presented in this paper, the SDN functionalities should include, along with basic network operations, a network control and network management framework together with a programmable software platform for the administration of the controller and the switches. Such important issues like the controller platform management, especially its performance, fault handling through hot swappable solutions and functional upgradeability have to be addressed. To cope with scalability and complexity, separate platforms for the control and the management of SDN networks are recommended. Proper information exchange between these platforms is necessary, which could be provided by programmable interfaces.

We have also proposed and discussed some variants of SDN management. It appears that the variant, which is based on embedded, programmable management solutions, is the appropriate one in the long term. It is in line with the latest concepts of autonomic and cognitive techniques for network management. Such approach calls for a distributed execution environment and can be implemented using already available software technologies. The distribution of the management functions provides multiple benefits in terms of scalability and the reaction time and the programmability of such approach makes the solution flexible, relaxes the standardization efforts and reduces time-to-market. In fact, this approach would enable not only decentralized operations but also a centralized ones — the choice of the way of implementation can be selected on a per function level. The proposed approach still raised one important issue that can impact the way in which SDN could evolve. The problem is related to a proper split of functions between the controller and management platform and their mutual interactions.

It can be noted that the last solution that we advocate for the management of SDN can be reused as well for the implementation of the distributed control plane of SDN. The common denominator is the distributed execution environment that is installed in the SDN controller as well as in the SDN switches.

It is expected that slightly higher complexity of the intelligent SDN switches will result in a significant reduction of OPEX and much higher scalability and reliability of the SDN-based networks. There is however no doubt that the added complexity should not increase the cost the SDN switches significantly and that the distribution of functionalities should still give the centralized view of the network. We believe that the above-mentioned properties may have an important impact in the future SDN development.

References

[1] Y. Rekhter, B. Davie, E. Rosen, G. Swallow, D. Farinacci, and D. Katz, "Tag switching architecture overview," Proc. IEEE, vol.85, no.12, pp.1973–1983, Dec. 1997.
[2] https://www.opennetworking.org/
[3] 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, March 2008.
[4] A.T. Campbell, H.G. De Meer, M.E. Kounavis, K. Miki, J.B. Vicente, and D. Villela, "A survey of programmable networks," SIGCOMM Comput. Commun. Rev. 29, pp.7–23, April 1999.
[5] D. Tennenhouse, J. Smith, W. Sincoskie, D. Wetherall, and G. Minden, "A survey of active network research," IEEE Commun. Mag., vol.35, no.1, pp.80–86, Jan. 1997.
[6] J. Smith and S. Nettles, "Active networking: One view of the past, present, and future," IEEE Trans. Syst. Man. Cybern. C, Appl. Rev., vol.34, no.1, pp.4–18, Feb. 2004.
[7] M. Brunner and R. Stadler, "Management in telecom environments that are based on active networks," J. High Speed Netw., vol.9, no.3-4, pp.213–230, Dec. 2000.
[8] http://datatracker.ietf.org/wg/forces/
[9] http://en.wikipedia.org/wiki/Iptables
[10] T. Abu Hmed, A. Mohaisen, and D.H. Nyang, "A survey on deep packet inspection for intrusion detection systems," Magazine of Korea Telecommunication Society, vol.24, no.11, pp.25–36, Nov. 2007.
[11] J. Roberts and S. Oueslati-Boulahia, "Quality of service by flow-aware networking," Philosophical Transactions of The Royal Society of London, Series A, vol.358, no.1773, pp.2197–2207, Aug. 2000.
[12] http://www.networkworld.com/news/2007/080607-roberts-fast-flow-routing.html
[13] L.G. Roberts, "A radical new router," IEEE Spectr., vol.46, no.7, pp.34–39, July 2007.
[14] B. Claise, ed., RFC 5101, "Specification of the IP flow information export (IPFIX) protocol for the exchange of IP traffic flow information," IETF, Jan. 2008.
[15] R. Sherwood, G. Gibby, K-K. Yapy, G. Appenzeller, M. Casado, N. McKeown, and G. Parulkar, "FlowVisor: A network virtualization layer," TR-2009-1, Stanford University Technical Report, Oct. 2009.
[16] N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown,

and S. Shenker, "NOX: Towards an operating system for networks," SIGCOMM Comput. Commun. Rev., vol.38, no.3, pp.105–110, July 2008.
[17] http://www.opendaylight.org/
[18] http://www.projectfloodlight.org/floodlight/
[19] U. Holzle, OpenFlow @ Google, Open Networking Summit 2012, Santa Clara, April 2012.
[20] http://www.enterprisenetworkingplanet.com/datacenter/cisco-opens-up-networking-fabric-for-sdn.html
[21] http://www.networkworld.com/news/2011/102611-juniper-open flow-252389.html
[22] ITU-T, Resolution 77 — Standardization work in ITU-T for software-defined networking, World Telecommunication Standardization Assembly, Dubai, Nov. 2012.
[23] I2RS — "Interface to The Internet Routing System (IRS)," https://www.ietf.org/mailman/listinfo/i2rs
[24] http://irtf.org/sdnrg
[25] 3GPP TS 32.500, Self-Organizing Networks (SON): Concepts and requirements (Release 9), Dec. 2009, Sophia-Antipolis, France.
[26] ITU-T, Recommendation M.3400: TMN management functions, 1997, Geneva, Switzerland.
[27] ITU-T, Recommendation M.3050: Enhanced Telecom Operations Map (eTOM) — The business process framework, March 2007, Geneva, Switzerland.
[28] http://www.fp7-socrates.org/
[29] http://www.4ward-project.eu/
[30] http://www.efipsans.org/
[31] http://www.univerself-project.eu/
[32] http://portal.etsi.org/portal/server.pt/community/AFI/344
[33] M.C. Huebscher and J.A. McCann, "A survey of autonomic computing — Degrees, models, and applications," ACM Comput. Surv. 40, 3, Article 7, Aug. 2008.
[34] Open Networking Foundation, OpenFlow Management and Configuration Protocol (OF-Config 1.1.1), ONF, March 2013.
[35] A. Tootoonchian, S. Gorbunov, Y. Ganjali, M. Casado, and R. Sherwood, "On controller performance in software-defined networks," Proc. Hot-ICE, p.10, 2012.
[36] S. Kuklinski, M. Skrocki, L. Rajewski, J. Meseguer Llopis, and Z. Wereszczynski, "GARSON: Management performance aware approach to autonomic and cognitive networks," IEEE Globecom/MENS 2012, Anaheim, USA, Dec. 2012.
[37] http://www.fipa.org/
[38] http://www.osgi.org/Technology/WhatIsOSGi
[39] "Network functions virtualisation," White Paper, ETSI, Oct 2012.

Sławomir Kukliński received M.Sc. and Ph.D. degrees in telecommunications from Warsaw University of Technology in 1985 and 1994, respectively. From 1985, he is with Warsaw University of Technology, Institute of Telecommunications, and since 2002 he is also working at Orange Labs Poland. His interests include autonomic network management, Future Networks and Software-Defined Networks.

Prosper Chemouil graduated from Ecole Centrale de Nantes in 1975 and obtained a PhD in control theory in 1978. He joined the Centre National d'Études des Télécommunications (CNET, now Orange Labs) end of 1980, and led research on dynamic routing and traffic engineering. He is now a Research Director on Future Networks at Orange Labs, France. His current interests are with new network design and management approaches including information-centric, programmable and autonomic networking. Prosper Chemouil is an IEEE Fellow.