Leveraging SDN To Improve Security in Industrial Networks

Leveraging SDN To Improve Security in Industrial Networks

Manuel Cheminod, Luca Durante, Lucia Seno, Fulvio Valenza, Adriano Valenzano, Claudio Zunino
National Research Council of Italy (CNR–IEIIT), Corso Duca degli Abruzzi 24, I-10129 Torino, Italy
Emails: {manuel.cheminod, luca.durante, lucia.seno, fulvio.valenza, adriano.valenzano, claudio.zunino}@ieiit.cnr.it

***Abstract*—In recent years, several important initiatives have appeared worldwide, aimed at bringing significant innovation in industrial networked systems (INSs). As an example, the Industry 4.0 and Factory of the Future frameworks are paving the way to modern intelligent factories, where issues such as the communication complexity between smart devices and system on-the-fly reconfiguration are dealt with in efficient and cost-effective manner. However, global connectivity also implies constant increase of cyber threats targeting industrial systems, so security must be considered since the very beginning when new appealing solutions need to be conceived.**

In this paper, we exploit the innovative Software Defined Networking (SDN) paradigm to introduce improvements in managing the network infrastructure of INSs, as this can help in reducing the management costs and complexity. In particular, enhanced SDN functionalities are adopted, which are able to provide security support in additions to their native switching/routing functionalities. The paper also shows how this approach can overcome some limitations of many current INS security architectures. The feasibility of the proposed solution is confirmed by the development of a simple laboratory prototype based on commodity hardware, and used to obtain some preliminary evaluation of the achievable functionality and performance benefits.

I. Introduction

Since some years academy and industry have been focusing researches and investments on setting the stage for a new generation of advanced industrial networked systems (INSs). Factory of the Future [1] and Industry 4.0 [2] are two clear examples of initiatives that share ambitious technical goals such as the ability, in future factory automation systems, to manage more and more complex communications between smart devices or the capability to orchestrate modular systems connected in heterogeneous and distributed architectures. Other significant investigations are oriented at satisfying the demand for exceptional (real-time) flexibility of both production and plants.

As a matter of facts, however, all evolutionary scenarios consider the INS communication infrastructure and its security as critical issues, since the exposure to cyber-threats, which already affects many INSs today, is expected to increase dramatically in the near future.

As the size and complexity of INSs increase, the correct configuration of traffic control and security devices becomes a harder and harder task. Moreover, too frequently this burden relies on special proprietary software applications, which are difficult or even impossible to integrate in a general management framework. A mandatory requirement is then introducing flexibility directly in the network infrastructure and devices, so that new communication needs, that can appear dynamically in the system, can be satisfied in a timely and efficient manner.

One viable solution is moving the management of both the network traffic and its security to a more abstract and global level not based on the device-by-device configuration approach, thus decoupling the control and (low-level) data planes. High-level strategies and policies can then be enforced in the devices if they support a way for software-programming. These concepts are the core of the Software Defined Networking (SDN) and Network Function Virtualization (NFV) paradigms. SDN, in particular, separated the control and data planes and provides an open and standard interface to enable the programmability of the network, while NFV offers a virtual execution environment to run network and security functions independently of the underlying physical equipment. As a consequence, any security application, such as a firewall, can be run on commodity hardware as an instance of plain software.

To exploit the power of SDN and NFV, this paper proposes the adoption of *active* SDN switches (called A-switches in the following), which are able to host security applications besides their native switching/routing functionalities. A-switches include native support for global management because of their SDN nature, and also allow the dynamic deployment and reconfiguration of security mechanisms in an efficient and distributed fashion.

The paper is structured as follows. Section II summarizes some limitations of current approaches to INS security that are relevant to this paper. Section III deals with the introduction of SDN and NFV in industrial networked systems. Section IV presents the proposed approach, while Section V describes a small laboratory prototype implementation and discusses the obtained preliminary results. Section VI recalls some relevant works appeared in the literature and, finally, Section VII concludes the paper.

II. INS Security Issues

INS security is a manyfold and very complex process [3], [4] involving aspects such as identification of assets, risk analysis and management, protection strategy definition, countermeasures design, deployment and maintenance just to mention a few.

Actually, the security of most INS infrastructures is addressed directly by means of dedicated devices. In particular,

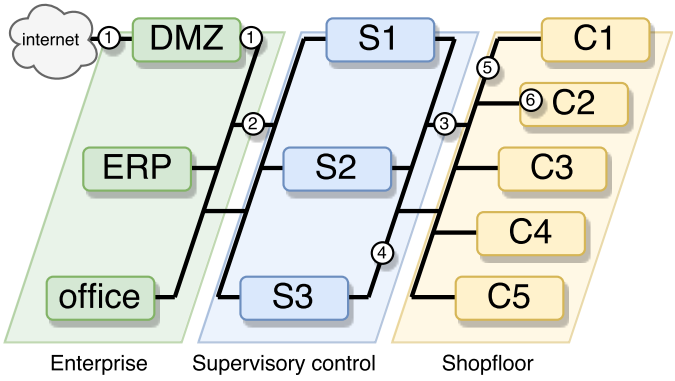

Fig. 1. Typical INS architecture and (sub)network hierarchy.

defense against external threats is obtained by deploying countermeasures (often referred to as security controls) at the perimetral border of the whole system. Of course, malicious entities attacking the system from its inside cannot be neglected and must dealt with carefully.

For the purpose of this paper we take into account INSs whose logical architecture resembles that shown in Fig. 1. In practice, the system consists of a hierarchy of (sub)networks interconnecting devices belonging to different subsystems. Typical areas in this architecture are the following:

- shopfloor (e.g., automation cells) including intelligent sensors/actuators and control/automation devices, such as PLCs (Programmable Logic Controllers), HMIs (Human Machine Interfaces) and robots;
- supervisory control including engineering workstations, data servers, HMIs, SCADAs (Supervisory Control And Data Acquisitions) and so on;
- enterprise including management and office-related servers, workstations (e.g., Enterprise Resource Planning servers, web servers, database servers) and the demilitarized zone (DMZ).

Fig. 1 also shows conventional placements for protection devices, where the meaning of labels in the picture is the following:

- (1) protection from unwanted accesses from/to the Internet and separation of some (critical) services from the plant;
- (2), (3) protection between different plant areas;
- (4), (5) protection between different subsystems in the same area;
- (6) protection inside a (production/automation) cell at the shopfloor level.

Indeed, for technical and financial reasons security devices cannot be deployed everywhere and, consequently, the evaluation of priorities plays a main role in finding suitable trade-offs between protection coverage on the one hand and costs and complexity of management on the other hand.

The most popular scheme adopted for securing INSs is based on different levels of countermeasures that an attacker has to face when moving from left to right in Fig. 1 (defense in depth) [5]. Although this solution has several advantages, it is not very flexible and efficient if implemented with conventional technologies, in particular when communication and security requirements change dynamically to adapt to new fast-evolving functional needs, as it is likely to occur in future automated production plants/machines. A typical example of stiffness, from this point of view, is represented by some widespread industrial protocols (e.g. Modbus/TCP, Ethernet/IP) used to move commands and data at the shopfloor level. Ad-hoc security mechanisms (i.e. deep packet inspection) are often needed in this case, that are embedded in special devices to make protection effective. Unfortunately, this also significantly decreases the ability to (re)configure the network seamlessly when needs to do so arise during the system operation.

In this paper we focus on firewalls as they are devices widely used for INS protection. However, our proposal has a broader scope and can be easily adapted also to other kinds of security functions and countermeasures. In particular, we are interested in those firewalls that are able to recognize and deal with typical industrial protocols such as Modbus, OPC (Open Platform Communications) or Ethernet/IP (e.g. [6], [7]). In such a context, our approach aims at dealing with INS security efficiently and flexibly, by equipping traffic control devices (switches) with modular virtual functions able to implement the required mechanisms in an optimized way. Classic aspects, which are taken into account in this solution, are the following:

- needed number of firewalls, assuming an operating environment where resources are limited (e.g., some current solutions foresee the deployment of an industrial firewall for every device to be protected, so leading to difficult trade-offs in balancing costs and level of protection);
- impact on communication performance (e.g., delays introduced by firewalls can become unacceptable in certain operating conditions).

For example, in the remaining part of this paper we show how the problem of delays introduced by firewalls (for both general purpose and industrial protocols) can be tackled and lessened at least, by distributing the set of filtering rules among the available resources.

III. SDN and Industrial Networks

Software Defined Networking is perhaps the most promising paradigm for network management that has rapidly gained consensus in many information technology communities [8]. SDN splits the management of network devices into two disjoint layers, concerning, respectively, the control functions (configuration and behavior commands) and data forwarding (implementation of commands through low-level configuration instructions). A central logical controller connecting all devices is defined and deployed in the system, which is responsible for sending commands to the devices themselves through well defined protocols, so as to supervise the behavior of the whole network. Evident advantages of this are augmented flexibility and management transparency, moreover

high level applications can be better integrated in the overall system workflow and leverage physical resources in optimized ways [9].

SDN also contributes to virtualize the network infrastructure, providing foundation for dynamic abstraction and sharing of resources [10]. Indeed, management of virtual devices is easier because of their software nature which exposes a uniform interface through standard abstractions.

For this reasons SDN is receiving increasing attention for industrial applications too, even though main instances of this technology are at present found in general-purpose hardware, supporting, in particular, the OpenFlow standard [11]. It is worth remembering that Openflow forwarding devices (OpenFlow switches) include flow tables and an abstraction layer for secure communications with the centralized controller via the OpenFlow protocol. In turn, flow tables consist of entries, each one specifying how packets belonging to a specific flow have to be processed and forwarded.

The introduction of SDN in industrial scenarios is quite recent, but some promising results have started to appear [12], [13]. Actually, these works confirm that SDN can offer clear advantages also in industrial applications, though security aspects of our interest, such as protection-in-depth and deep inspection capabilities of industrial protocols, are not considered at all. A limitation of most solutions proposed so far is the adoption of hybrid architectures including both SDN-enabled devices and legacy industrial middleboxes. Actually, any device, which is not managed by the central controller, decreases the system flexibility and, mostly important, may overrule the coherency of deployed configurations. By contrast, approaches that host network and security applications directly in the central controller are not viable in many practical situations, since the controller itself may either became a sort of performance bottleneck or even an appealing target for DoS attacks (Denial of Service attacks).

The proposal presented in the following takes full advantage of SDN by adopting active SDN switches (A-switches) equipped with a custom application layer to implement the required network and security functionalities locally (that is by hosting them on the switch which is well integrated with the SDN controller). Main benefits obtained in this way are:

1) distribution of security applications over the network, lessening problems such as bottlenecks and DoS attacks;
2) flexible, dynamic and optimal configuration/update of the network, as A-switch functioalities can be moved through the network according to different strategies and at a different time;
3) reduction of both costs and complexity in deploying, operating and managing the overall system, as A-switches can be run on conventional hardware supervised by the SDN controller.

IV. Active SDN switches

The basic structure of A-switches is shown in Fig. 2. From a logical point of view, each device includes two functional

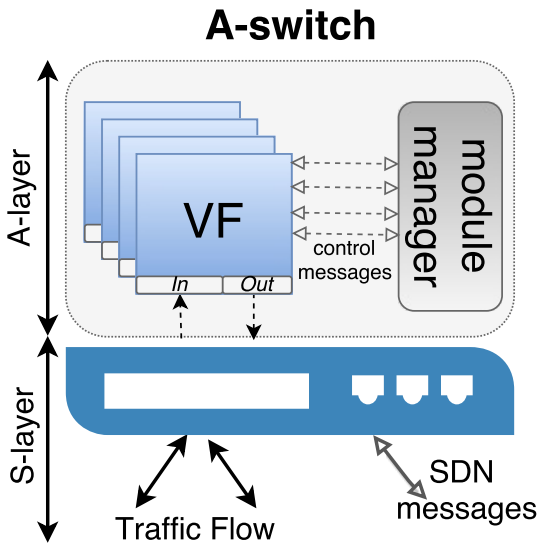

Fig. 2. Active SDN switch architecture.

layers hosted on the same hardware and referred to as S-layer and A-layer respectively. Basic S-layer functionalities are provided by conventional SDN (Openflow-based) switches and usually implemented by software applications. S-layer manages the flow of network packets reaching and exiting the device through its physical ports. The configuration of the underlying flow tables establishes the actual behavior for the switch.

The A-layer consists of a general-purpose software environment where virtual functions (VFs) can be installed and activated in a modular way. Any VF is implemented as a software module connected to two logical network interfaces labeled *In* and *Out* in Fig. 2. *In* provides the module with packets to be analysed and processed, then resulting packets are sent back through *Out*.

Connections between S-layer and VFs adopt couples of virtual interfaces that enable packet transfers between the SDN switch and the *In*/*Out* pair of each VF module. Flow-tables are used by S-layer to route packets to/from the relevant VF module.

This kind of architecture is quite flexible, as functions can be developed and added to the base SDN switch as needed, that is without requiring custom modifications of the existing switch implementation. It is also worth noting that several VFs can be concurrently active on the same hardware, and packets can be forced to "traverse" ordered sequences of VFs by suitably crafting flow rules to connect the *In* and *Out* pairs of different VF modules. In this way complex per-packet analysis and processing can be obtained in a modular way.

Configuration of A-switches occurs through the conventional SDN support (i.e. the standard OpenFlow protocol in our laboratory prototype) which enables the reception and interpretation of commands sent by the SDN controller. Configuration of VF modules, instead, is performed through a VF "module

manager" application which is hosted on each A-switch to receive and process special messages from the controller.

A-switches can help with the implementation of many kinds of network functions. For instance, services designed to change the contents of specific packet fields such as the NAT (Network Address Translation) can be easily obtained by configuring the S-layer flow rules so as to forward packets to the NAT module and return results back to the switch layer. Similarly, support for anomaly detection can be added to A-switches as well: S-layer flow rules can be configured to forward packets as usual, while sending copies to a VF module performing anomaly detection at the same time. In general, flexibility of flow rules allows to treat different packet streams flowing through the same A-switch with different VFs or even chains of VFs.

In our prototype implementation we leveraged rule selectors of Openflow to implement basic and fast packet filtering and then added more complex filtering operations (e.g., Deep Packet Inspection) through another VF module (DPI) developed ad-hoc. Indeed, when Modbus/TCP messages are considered, S-layer can be configured to recognize that a deeper analysis is needed for packets addressing TCP port 502 so that they are routed to the DPI module. DPI, according to configuration set by the VF module manager, processes the incoming packets and, if specific criteria are met, forwards them to the S-layer again through its *Out* port. More details about the advantages achievable with this architecture are also given in the following section together with a brief comparison to traditional solutions in the light of drawbacks of conventional firewall deployment discussed in Sec. II.

The ability to distribute filtering rules all over the system, optimizing the use of resources, must be stressed once again here. In fact, one main problem of firewalls implemented with conventional technologies is the need to include the whole set of filtering rules in their configuration. This might turn the device into a real bottleneck, depending on the number of rules to be checked, and also introduce excessive delays for packets subject to real-time requirements. A-switches, instead, allow to scatter the filtering activities in different network nodes and also contribute to fully support the defence in depth strategy with any kind of system topology.

V. Laboratory testbed

To assess the validity of our approach we developed a simple A-switch prototype to be configured with basic filtering functions for both the S- and A- layers.

The switch has been implemented using common off-the-shelf (COTS) hardware, as our main goal was evaluating the approach feasibility and flexibility rather than optimizing its performance. For this reason, low-cost Raspberry Pi 3 (RPi3)[1] embedded devices were selected for their wide spread and openness of the software environment. S-layer was based on the Open vSwitch (OVS) [14] open source project running on the RPi3 Linux (Raspbian) distribution. Each A-switch was also equipped with two additional physical network interfaces

[1]http://www.raspberrypi.org/ (23 March 2017).

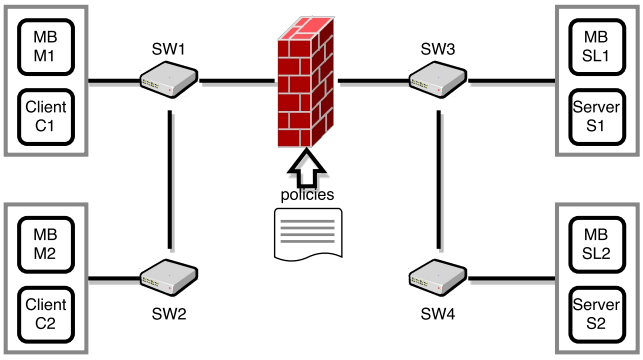

Fig. 3. Case study, with industrial firewall

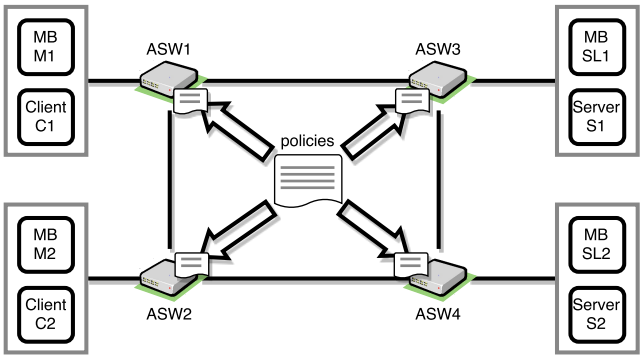

Fig. 4. Case study, with *A-switches*.

obtained by means of USB-to-Ethernet adapters, to reach the total number of 3 Ethernet ports per device running at 100Mbit/s.

A custom VF module, implementing DPI for Modbus/TCP messages was developed in C language and activated in the A-layer of RPi3 nodes. This VF is able to analyze each received packet and to look for matches with a set of filtering rules involving Modbus/TCP specific commands parameters. As pointed out in the previous section, a pair of *In* and *Out* ports connects the VF module to the S-layer (OVS). The configuration of OVS flow rules defines which packets flowing through S-layer have to be redirected to the DPI module. Modbus/TCP packets are detected by checking the destination TCP port number (port 502 for this kind of traffic). IP-level filtering for other packets is also carried out through OVS and flow rule configurations, that were suitably set to recognize IP and TCP/UDP headers for acceptable flows.

The testbed structure shown in Fig. 3 can mimic a very simple industrial subnetwork including two kinds of devices: nodes appearing on the right side of the picture represent equipment belonging to the shopfloor (e.g.: PLCs, intelligent sensors/actuators) while devices on the left side host client applications that are placed in the supervisory area. Shopfloor devices, in particular, are Modbus slaves (SL1, SL2) and http servers (S1, S2). In this scenario, Modbus slaves accept requests and provide response data to masters M1 and M2. Communications are established through the Modbus/TCP

TABLE I
DESCRIPTION OF FLOWS IN THE CASE STUDY.

source	destination	protocol	frequency
M1	SL1	Modbus/TCP	100 ms
M1	SL2	Modbus/TCP	200 ms
M2	SL1	Modbus/TCP	200 ms
M2	SL2	Modbus/TCP	100 ms
C1	S1	ICMP,HTTP,HTTPS	1 s
C1	S2	ICMP,HTTP,HTTPS	1 s
C2	S1	ICMP,HTTP,HTTPS	1 s
C2	S2	ICMP,HTTP,HTTPS	1 s

TABLE II
SET OF FILTERING RULES CONFIGURED ON FW FIREWALL.

n.	source	dest.	protocol	Modbus FC	action
1	C1	S1	ICMP	-	allow
2	C1	S1	HTTP	-	allow
3	C1	S1	HTTPS	-	allow
4	C1	S2	ICMP	-	allow
5	C1	S2	HTTP	-	allow
6	C1	S2	HTTPS	-	allow
7	C2	S1	ICMP	-	allow
8	C2	S1	HTTP	-	allow
9	C2	S1	HTTPS	-	allow
10	C2	S2	ICMP	-	allow
11	C2	S2	HTTP	-	allow
12	C2	S2	HTTPS	-	allow
13	M1	SL1	Modbus/TCP	3, 16	allow
14	M1	SL2	Modbus/TCP	3, 16	allow
15	M2	SL1	Modbus/TCP	3, 16	allow
16	M2	SL2	Modbus/TCP	3, 16	allow
*	ANY	ANY	ANY	ANY	deny

protocol. Servers S1 and S2, instead, expose diagnostic web pages to clients C1 and C2. Communication flows involved in the testbed are summarized in Tab. I. To keep the prototype manageable, instances of the same role have been implemented as independent tasks and suitably grouped on few physical nodes. This is shown by surrounding boxes in Figs. 3 and 4, so that tasks playing the M1 and C1 roles run of the same device and so on.

When conventional technologies are considered, FW in Fig. 3 is an industrial firewall available on the market [7], which offers Deep Packet Inspection capabilities for Modbus/TCP packets. In practice, FW is a protection device such as the one labeled (3), which separates different plant areas in Fig.1. Because of the position of FW in the network, the set of filtering rules shown in Tab. II was adopted in our preliminary experiments.

The scenario was then modified by replacing switches and FW in Fig. 3 with A-switches as shown in Fig. 4. Of course, no firewall is present in this SDN-based system architecture. Because of the absence of the (quite expensive) FW device, the new architecture is more flexible, robust and also symmetrical, thanks to the addition of a direct link connecting ASW2 and ASW4 in Fig. 4. In this condition each A-switch was loaded with a subset of the filtering rules listed in Tab. II.

TABLE III
SET OF FILTERING RULES CONFIGURED ON *A-switches*.

ASW1

n.	source	dest.	protocol	Modbus FC	action
1	C1	S1	ICMP	-	allow
2	C1	S1	HTTP	-	allow
3	C1	S1	HTTPS	-	allow
4	C1	S2	ICMP	-	allow
5	C1	S2	HTTP	-	allow
6	C1	S2	HTTPS	-	allow
13	M1	SL1	Modbus/TCP	3, 16	allow
14	M1	SL2	Modbus/TCP	3, 16	allow
*	ANY	ANY	ANY	ANY	deny

ASW2

n.	source	dest.	protocol	Modbus FC	action
7	C2	S1	ICMP	-	allow
8	C2	S1	HTTP	-	allow
9	C2	S1	HTTPS	-	allow
10	C2	S2	ICMP	-	allow
11	C2	S2	HTTP	-	allow
12	C2	S2	HTTPS	-	allow
15	M2	SL1	Modbus/TCP	3, 16	allow
16	M2	SL2	Modbus/TCP	3, 16	allow
*	ANY	ANY	ANY	ANY	deny

ASW3

n.	source	dest.	protocol	Modbus FC	action
			(responses)		
*	ANY	ANY	ANY	ANY	deny

ASW4

n.	source	dest.	protocol	Modbus FC	action
			(responses)		
*	ANY	ANY	ANY	ANY	deny

Rules were distributed by taking into account that each switch should check only packets received from directly connected hosts (packets forwarded by other ASWs have already been validated). For instance, ASW1 have to check only packets coming from M1 and C1, while it may forward other packets without any inspection.

The resulting rule distribution is shown in Tab. III. With this assignment, ASW3 and ASW4 simply accept "response" messages without the need to perform any deep packet inspection. This is coherent with the behavior of FW in Fig. 3.

A. Preliminary performance evaluation

Some preliminary performance tests were carried out with our testbed. In particular, we focused on the RTT (Round Trip Time) for request/reply pairs, involved in Modbus master-slave communications, as the index of interest. Indeed, this is a critical parameter in several industrial systems as high RTT values can violate timing constraints and affect the

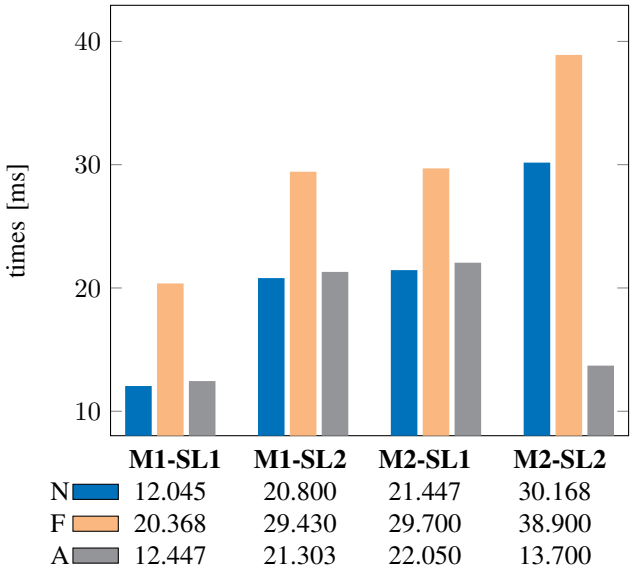

Fig. 5. Average RTTs for each configuration and master-slave pair.

control action negatively. RTTs values were then measured and compared for scenarios in Fig. 3 and Fig. 4.

For sake of clarity the reader must be warned that, in doing this, we are neither investigating absolute performance achievable with A-switches, nor comparing their filtering capabilities to FW, but rather trying to have some indication of the performance drops introduced by the two approaches with respect to a reference situation where no filtering policy is adopted. Detailed FW performance figures can be found in [15], where also the methodology we plan to follow to characterize the A-switches as a future work is described. Here we consider the three following configurations:

N : all A-switches work as pure Ethernet devices, e.g., no filtering is enabled (e.g., the network in Fig. 3 without the FW device);

F : switches are configured as in the previous case, but FW is deployed between SW1 and SW2 and configured with the filtering rules in Tab. II;

A : the network in Fig. 4, where SDN features are activated in all A-switches and filtering rules (as defined in Tab. III) are applied for both generic IP packets and Modbus/TCP specific messages.

For all configurations, four Modbus masters (two located on M1 and two on M2) request and get data from slaves SL1 and SL2. A sequence of 7 Modbus requests was generated by each master every 100ms. RTT was computed as the difference between the last received response and the first issued request. Average values were collected over 20 second time intervals (that is, 200 request/reply exchanges). Results obtained for the three configurations listed above are shown in Fig. 5. In the figure M1-SL1 means exchanges between master M1 and slave SL1 and so on.

Despite obtained results are very preliminary, they show that:

- delays introduced by FW in filtering a sequence of seven Modbus requests/replies exceed 8 ms on average. This confirms that currently available industrial devices can cause performance degradation.
- The SDN-based prototype implementation behaves even better than expected, as it limits delay values to less than 1 ms, on average.
- Good communication performance on the M2-SL2 path in the **A** configuration are due to the direct connection between ASW2 and ASW4 (unavailable in the reference architecture), that enables the flow of packets through a shorter network path.

VI. Related Works

SDN-based virtualization and management of middleboxes are recent issues in INSs. [16] and [17] pointed out the policy enforcement problem in presence of middleboxes able to alter packet flows. Also [18], [19] faced with middleboxes to validate the correct policy enforcement by exploiting formal methods and mathematical reasoning. [16] proposed a complex technique to force packets to flow through a proper sequence of middleboxes implementing policies correctly, where no special middlebox deployment was required. The solution described in [17], instead, is based on suitably tagging network packets. In this case changes to middleboxes are needed to manage tags correctly. Moreover, the SDN controller is responsible for tracking semantics and instructing network devices in both approaches. Our solution is not constrained by the deployment of middleboxes. In fact, applications can be properly combined and hosted on most convenient A-switches.

Deep packet inspection (DPI), which is needed by most middleboxes dealing with layer 7 protocols, was investigated in [20] and [21]. To limit computing costs, authors of [21] proposed the design of DPI as a service running once for each packet and able to forward relevant results to registered middleboxes. Our approach, instead, addresses the middlebox complexity by distributing their functionalities over the network, whose topology is no longer constrained by physical devices. In particular, network operations are performed by A-switches as close as possible to packet sources, so minimizing the number of hops and bandwidth usage.

Work in [12] tackled the deployment of DPI in SDN networks and authors concluded that the most efficient solution is the inclusion of DPI software modules in OpenFlow switches. Our work is similar to their approach from the point of view of performance, but it is not limited to DPI as more general issues are taken into account.

In [13] a suitable architecture is presented to include general middlebox functionalities in Open vSwitch and focused, in particular, on pure performance. As described in the previous sections, Open vSwitch can be a good building block to develop systems based on A-switches.

VII. Conclusions

This paper presented a novel approach to improve the IT infrastructure management of industrial networked systems,

by leveraging the Software Defined Networking and Network Function Virtualization paradigms. The adoption of active SDN switches has been described, that are able to provide network and security services in addition to their native switching functionalities. To assess the validity of the proposal, a laboratory testbed has been developed, which includes A-switches implemented by means of COTS hardware and open-source software. A preliminary evaluation confirms the feasibility of the solution and a clear improvement in flexibility and configurability. Of course, deeper investigations about performance are still needed to better assess the achievable benefits.

Future activities are then planned to extend the capabilities of the SDN-based architecture in several directions. In particular they include more detailed analyses of filtering functions and their distribution in a correct and optimized way. The prototype will also be improved with a better version of the VF manager, to deal with the dynamic scaling of resource allocation to virtual functions. At the SDN controller level an orchestrator will be developed, which is able to map high level policies and requirements to low-level configurations of traffic control devices deployed in the network.

REFERENCES

[1] A.-W. Colombo, S. Karnouskos, and J.-M. Mendes, *Factory of the Future: A Service-oriented System of Modular, Dynamic Reconfigurable and Collaborative Systems*. Springer London, 2010, pp. 459–481.

[2] H. Kagermann, J. Helbig, and W. Wahlster, *Recommendations for Implementing the strategic initiative INDUSTRIE 4.0: securing the future of German manufacturing industry; final report of the Industrie 4.0 working group*. Forschungsunion, 2013.

[3] M. Cheminod, L. Durante, and A. Valenzano, "Review of Security Issues in Industrial Networks," vol. 9, no. 1, pp. 277–293, 2013.

[4] A. Ray, J. Åkerberg, M. Björkman, and M. Gidlund, "Future research challenges of secure heterogeneous industrial communication networks," in *Proc. of the 21st IEEE Int. Conf. on Emerging Technologies and Factory Automation (ETFA)*. IEEE, sep 2016, pp. 1–6.

[5] D. Dzung, M. Naedele, T. P. V. Hoff, and M. Crevatin, "Security for industrial communication systems," *Proceedings of the IEEE*, vol. 93, no. 6, pp. 1152–1177, 2005.

[6] Moxa, *EDR-810 Whitepaper*, Mar. 2017. [Online]. Available: www.moxa.com/product/EDR-810.htm

[7] Hirschmann Automation and Control GmbH, *Tofino Xenon security appliance*, Mar. 2017. [Online]. Available: https://www.e-catalog.beldensolutions.com/link/57078-24455-49853-411807/en/conf/0

[8] B. A. A. Nunes, M. Mendonca, X. N. Nguyen, K. Obraczka, and T. Turletti, "A survey of software-defined networking: Past, present, and future of programmable networks," *IEEE Communications Surveys Tutorials*, vol. 16, no. 3, pp. 1617–1634, Third 2014.

[9] H. Kim and N. Feamster, "Improving network management with software defined networking," *IEEE Communications Magazine*, vol. 51, no. 2, pp. 114–119, February 2013.

[10] N. M. K. Chowdhury and R. Boutaba, "A survey of network virtualization," *Computer Networks*, vol. 54, no. 5, pp. 862 – 876, 2010.

[11] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008.

[12] C. Cho, J. Lee, E. D. Kim, and J. d. Ryoo, "A sophisticated packet forwarding scheme with deep packet inspection in an OpenFlow switch," in *Proc. of the IEEE International Conference on Software Networking (ICSN 2016)*. IEEE, may 2016, pp. 1–5.

[13] E. J. Jackson, M. Walls, A. Panda, J. Pettit, B. Pfaff, J. Rajahalme, T. Koponen, and S. Shenker, "SoftFlow: A Middlebox Architecture for Open vSwitch," in *Proc. of the USENIX Annual Technical Conference (USENIX ATC 16)*. USENIX Association, Jul. 2016, pp. 15–28.

[14] B. Pfaff, J. Pettit, T. Koponen, E. J. Jackson, A. Zhou, J. Rajahalme, J. Gross, A. Wang, J. Stringer, P. Shelar, K. Amidon, and M. Casado, "The design and implementation of open vswitch," in *Proceedings of the 12th USENIX Conference on Networked Systems Design and Implementation*. USENIX Association, Mar. 2015, pp. 117–130.

[15] M. Cheminod, L. Durante, M. Maggiora, A. Valenzano, and C. Zunino, "Performance of firewalls for industrial applications," in *Proceedings of the 4th International Symposium for ICS & SCADA Cyber Security Research*. BCS Learning & Development, Aug. 2016, pp. 1–11.

[16] Z. A. Qazi, C.-C. Tu, L. Chiang, R. Miao, V. Sekar, and M. Yu, "Simple-fying middlebox policy enforcement using sdn," in *Proceedings of the ACM SIGCOMM conference on SIGCOMM (SIGCOMM '13)*. ACM, Aug. 2013, pp. 27–38.

[17] S. K. Fayazbakhsh, L. Chiang, V. Sekar, M. Yu, and J. C. Mogul, "Enforcing network-wide policies in the presence of dynamic middlebox actions using flowtags," in *Proc. of the 11th USENIX Symposium on Networked Systems Design and Implementation (NSDI 14)*. USENIX Association, Apr. 2014, pp. 543–546.

[18] C. Basile, D. Canavese, A. Lioy, C. Pitscheider, and F. Valenza, "Inter-function anomaly analysis for correct sdn/nfv deployment," *International Journal of Network Management*, vol. 26, no. 1, pp. 25–43, 2016.

[19] F. Valenza, T. Su, S. Spinoso, A. Lioy, R. Sisto, and M. Vallini, "A formal approach for network security policy validation," *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JoWUA)*, vol. 8, no. 1, pp. 79–100, 2017.

[20] M. Bouet, J. Leguay, and V. Conan, "Cost-based placement of virtualized deep packet inspection functions in sdn," in *Proceedings of the IEEE Military Communications Conference (MILCOM 2013)*. IEEE, Nov 2013, pp. 992–997.

[21] A. Bremler-Barr, Y. Harchol, D. Hay, and Y. Koral, "Deep packet inspection as a service," in *Proc. of the 10th ACM Int. Conf. on Emerging Networking Experiments and Technologies*. ACM, Dec. 2014, pp. 271–282.