Leveraging SDN to Monitor Critical Infrastructure Networks in a Smarter Way

Roberto di Lallo*, Federico Griscioli*, Gabriele Lospoto*, Habib Mostafaei*,
Maurizio Pizzonia* and Massimo Rimondini*
*Roma Tre University, Department of Engineering
Via della Vasca Navale 79, 00146 Rome, Italy
{dilallo,griscioli,lospoto,mostafae,pizzonia,rimondini}@ing.uniroma3.it

Abstract—**In critical infrastructures, communication networks are used to exchange vital data among elements of Industrial Control Systems (ICSes). Due to the criticality of such systems and the increase of the cybersecurity risks in these contexts, best practices recommend the adoption of Intrusion Detection Systems (IDSes) as monitoring facilities. The choice of the positions of IDSes is crucial to monitor as many streams of data traffic as possible. This is especially true for the traffic patterns of ICS networks, mostly confined in many subnetworks, which are geographically distributed and largely autonomous. We introduce a methodology and a software architecture that allow an ICS operator to use the spare bandwidth that might be available in over-provisioned networks to forward replicas of traffic streams towards a single IDS placed at an arbitrary location. We leverage certain characteristics of ICS networks, like stability of topology and bandwidth needs predictability, and make use of the Software-Defined Networking (SDN) paradigm. We fulfill strict requirements about packet loss, for both functional and security aspects. Finally, we evaluate our approach on network topologies derived from real networks.**

I. Introduction

ICSes are the core of critical infrastructures. They are composed by many elements that interact by means of a communication network, which we call *ICS network*. Main elements of an ICS are embedded devices that control actuators or gather data from sensors. Special servers are in charge to collect data from these embedded devices, show them to the control room operators, record them in a database, change settings according to operators requests, etc. While the data that flow in an ICS network are very specific, standard networking technologies can be adopted for its implementation.

In the past decade, a growth of cyber-attacks directed toward ICSes has been observed [1]. For the security of the ICS networks, best practices suggest to deploy network-based IDSes [2]. In regular networks it is acceptable to observe traffic in a small number of relevant points. However, for reliability reasons, in ICSes, Supervisory Control And Data Acquisition (SCADA) servers are close to sensors and actuators, hence, traffic is mostly local. Further, attacks to ICSes are potentially carried out by organizations (e.g., governments, intelligence agencies, terrorist groups) that can have insiders and that can carefully design attacks so that they pass unobserved by sparsely deployed IDSes. Tapping traffic close to all embedded devices and servers can easily lead to prohibitive costs.

Work partially supported by EU FP7 project "Preemptive - Preventive Methodologies and Tools to Protect Utilities", grant no. 607093.

In this paper, we present a methodological approach and an architecture to (i) allow an operator to choose which traffic has to be observed within an ICS network without installing new hardware, (ii) enable the use of the spare bandwidth in the network to forward the traffic to be observed toward an IDS, while avoiding packet loss for regular traffic, and (iii) guarantee that the IDS receives all the traffic that the operator configured to be observed in order not to introduce false negatives due to packet loss. Our solution takes advantage of the fact that topology and bandwidth usage are quite stable in ICS networks (e.g. [3]), allowing us to assume in advance knowledge of ICS network's traffic, since it derives from ICS design, and to perform a global off-line optimization of switching paths. Furthermore, we support the usage of the ICS network for additional and occasional traffic, which are always considered potentially dangerous. We assume that this traffic can be served with a best-effort approach while maximizing the endeavor in observing it. We propose an architecture that exploits the Software-Defined Network (SDN) approach as prescribed by the OpenFlow specifications [4]. We evaluated our methodology against four network topologies, derived from real topologies and augmented with realistic networks in the domain of electrical distribution. Our experiments show that our optimization problem can be easily solved for those scenarios in reasonable time and our approach makes efficient use of the bandwidth when the topology allows it.

The rest of the paper is organized as follows. In Sec. II, we describe the state of the art. In Sec. III, we introduce terminologies and requirments. In Sec. IV, we describe our methodology and our proposed architecture. In Sec. V, we evaluate our approach against realistic scenarios. Conclusions are drawn in Sec. VI. Further details are available in [5].

II. State of the Art and Background

ICS networks make use of proprietary protocols, as shown in [2]. Those protocols (e.g. ModBus [6]) are tipically application-layer, and they allow the communication among ICS devices. In many cases, proprietary protocols are used to compute routing [7], but this does not limit the adoption of different link-layer technologies [8] and new installations are based on widely adopted standards, like Ethernet. Protocols adopted in ICS networks do not consider security aspects. Well known recommendations (e.g. [2]) suggest the usage of IDSes.

In the last years, a new centralized architecture called Software-Defined Networking (SDN) is collecting the attention of the research community due to its promising benefits and, in particular, its flexibility in the selection of the paths to route packets [9]. There have been many attempts to exploit SDN in security contexts, relying on two different techniques. First, implementing the IDS as an SDN controller module (e.g. [10], [11]). Second, exploiting SDN to easily forward the traffic to dedicated IDSes (e.g. [12], [13]). However, we argue that those approaches are not applicable in ICS context.

A relevant aspect in our approach is traffic engineering. In [14], authors show that having a traffic-matrix allows traffic engineering problems to be easily solved. Usually, the traffic engineering problem is treated as a *multicommodity flow* problem whose solution is described in [15]. Proposals that are specific to traffic engineering for SDN can be found in [16], [17], [18]. At the best of our knowledge, our approach is the first attempt to apply traffic engineering to the specific context of traffic monitoring by IDS leveraging the coordinates of the topologies and traffic in ICS networks.

III. Terminologies and Requirements

In this section, we introduce some terminology and a few requirements which we aim to fulfil with our proposal. A more detailed explanation can be found in [5].

ICS networks connect several kinds of devices. For the purpose of our discussion we divide them in two categories. We call the first category *essential*: devices in this category can have a very diverse nature, but they are essential for the correct operation of the ICS, are part of the ICS design, and are always connected to the ICS network (e.g. actuators and sensors). We call the second category *non-essential*: occasionally, other devices can be attached to the ICS network, for example operators' notebooks to perform maintenance of ICS devices.

We call *stream* a communication between two devices on the ICS network. We identify it by its source and its destination, specified by IP addresses. Even though communications are usually bidirectional, throughout this paper we consider a stream to be unidirectional, which means that a full communication between two devices generally encompasses two streams. A stream can be critical or standard. In a *critical stream*, source and destination are essential devices and the properties about the stream are known in the ICS design phase. In particular their bandwidth demand, source, and destination are known. A *standard stream* is not essential for the current functioning of the ICS and it is not known in advance and it usually involves at least one non-essential device. Supporting standard streams is important to enable occasional use of the ICS network for maintenance or other non-critical activities. A best-effort delivery is enough for such streams.

From the point of view of the security concerns, both kinds of streams are equally important, since attacks may involve any of the two with equal chance of disruptive effects. An *attack* to the ICS network consists in any action that introduces unexpected traffic or unexpected changes to standard traffic. The goal of this paper is to provide a flexible way to use a centralized IDS. To achieve this, we assume that a standard stream σ is duplicated, generating a *replica stream*; this action is performed at a network node that we call *observation point*. Each replica stream $\bar{\sigma}$, associated with σ, originates at the observation point and ends at the IDS. To do that, we define the following requirements. (1) **Observation Points**. Our methodology should be able to support the observation of potentially any stream in the network, independently from topology and IDS placement. For security reasons, we prefer observation points close to the destination of streams. (2) **Reliable Replica Forwarding**. Our methodology should guarantee no packet loss for replica streams associated with critical or standard streams. This is important in order for the IDS to inspect all observed traffic and avoid false negatives due to packet loss. (3) **Reliable Critical Streams Forwarding**. Our methodology should be able to configure the ICS network so that, for the critical streams, no packets loss can occur due to congestion. (4) **Standard Streams Usability**. Our methodology should allow operators to use the ICS network for occasional tasks, by injecting new standard streams.

We also consider the well-founded technology constraint that imposes not to split streams. In fact, if packets of the same stream take different paths, uncontrolled reordering can happen, which is detrimental for TCP performance at best and can change the semantic of UDP communications at worst.

IV. Methodology and Architecture

In this section, we describe a methodology and an architecture that solve the problem described in Sec. III, with the aim of satisfying the requirements presented in the same section.

Our methodology assumes that the network is made of SDN switches that are compliant with the OpenFlow standard [4]. We exploit the OpenFlow features to have a fine-grade control over the network devices as well as over the traffic (and the possibility to flexibly forward it).

To meet Requirements 2 and 3, we configure the SDN network to shape each stream at its ingress node, so that packets enter the network at a specified constant rate and all packets exceeding the configured bandwidth are discarded. For critical streams, the configured maximum bandwidth is determined during the design as described below, so no packet drop should happen. For standard streams, this early limiting avoids congestion of internal nodes that could adversely impact critical streams. The shaping configuration exploits the *meter* feature of the OpenFlow specifications.

Our methodology encompasses a design phase and an operation phase, as depicted in Fig. 1. In the *design phase*, we require an *ICS designer* to determine the network topology and to list the critical streams along with their maximal required bandwidth. These data are provided as input to an *off-line routing solver*, which computes the configuration of the SDN switches for critical streams. More specifically, the input of the off-line routing solver encompasses (i) the network topology, (ii) the location of essential devices, (iii) the location

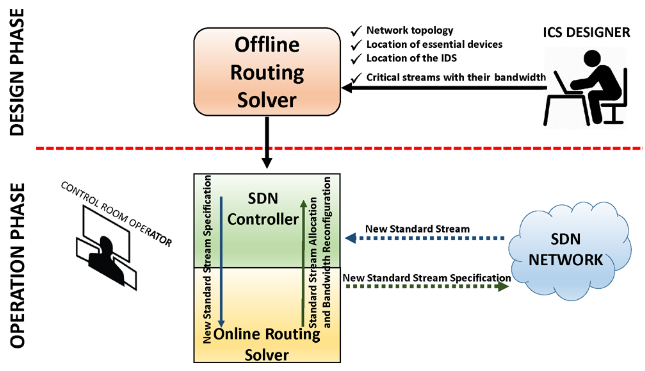
DESIGN PHASE
Offline Routing Solver
✓ Network topology
✓ Location of essential devices
✓ Location of the IDS
✓ Critical streams with their bandwidth
ICS DESIGNER
OPERATION PHASE
CONTROL ROOM OPERATOR
SDN Controller
New Standard Stream Specification
Standard Stream Allocation and Bandwidth Reconfiguration
Online Routing Solver
New Standard Stream
New Standard Stream Specification
SDN NETWORK

Fig. 1: Architecture of our system, with both offline and online routing solvers.

of the IDS, and (iv) for each critical stream its source, its destination and its bandwidth requirement. The off-line solver produces, for each critical stream, (i) a forwarding path, (ii) an observation point, and (iii) a forwarding path for the corresponding replica stream starting at the observation point and ending at the IDS. The off-line solver is based on an ILP formulation, which is described in detail in [5].

In the *operation phase*, we mandate the adoption of a special architecture (shown in Fig. 1) in which an *SDN-controller* is in charge of configuring forwarding paths and meters to implement shaping. Its configuration is divided into two parts: one for critical streams and one for standard streams. The part related to critical streams is configured on the basis of the result of the off-line solver and does not change during operation. The part related to standard streams dynamically changes during operation to adapt the configuration of the ICS network when the set of active standard streams changes. A *control room operator* can monitor the ICS network status during production time to have a clear picture of which streams are currently replicated and processed by the IDS.

During operation, any new packet reaching a network switch that does not match any of the rules configured in the switch to forward critical streams is treated as the first packet of a standard stream σ. This packet is forwarded to the SDN-controller as in the classical SDN approach. To compute the forwarding path for σ, the SDN-controller takes advantage of an *on-line routing solver*. This solver shares with the controller the network topology, and the current available bandwidth on each link derived from currently allocated paths. It takes as input the source s and destination t of σ and computes (i) a forwarding path P for σ, (ii) an observation point $op \in P$ (preferably close to t according to Requirement 1), (iii) a forwarding path Q from op to the IDS, and (iv) a new assignment of bandwidth for all standard streams comprising σ. The details of the on-line routing solver are described in [5]. These information are used by the controller to re-configure the shaping for all standard streams but σ. The new standard stream σ is configured only after a small amount of

	From Topology Zoo					Input for experiments			
	Name	$\|N\|$	$\|E\|$	min bw (bps)	max bw (bps)	q	$\|N\|+\|M\|$	$\|E\|$	num. strms
1	Cesnet	10	9	200M	600M	35	501	920	770
2	AttMpls	25	56	1G	1G	50	726	1357	1100
3	Agis	25	30	45M	155M	42	614	1123	924
4	Uninet	74	101	1G	1G	95	1405	2572	2090

TABLE I: Data about original topologies, and topologies used in the experimentation.

time τ that is dimensioned so that packets related to previous standard streams that where admitted in the network with the old bandwidth allocation are guaranteed to reach destination.

Concerning the path selection, our algorithm has a greedy approach keeping unchanged all paths previously allocated for both kinds of streams. There are several reasons for this choice: (i) sophisticated optimization techniques, like those used in in [5], may take a considerable amount of time, which can easily be even larger than the lifespan of the new stream and impair the usability of the network for occasional activities, (ii) modifying the path of a current stream can introduce temporary inconsistencies in the routing that can lead to packet loss, which is against Requirements 2 and 3, (iii) since standard streams have usually a short lifespan, our main goal is to support them within our requirements, keeping the optimization of their resource usage as a secondary goal.

V. Evaluation

We validated our approach from three points of view: (i) we assess the efficiency of our implementation with respect to computation time on realistic instances, inspired by the electricity distribution domain, for both on-line and off-line routing solvers, (ii) we show the efficiency of the bandwidth allocation of the on-line routing solver for standard streams, and (iii) we discuss the ability of our solution to meet requirements listed in Sec. III. We use four different realistic topologies taken from TopologyZoo [19], whose data are shown in Table I. Details about how we built the ICS networks are described in [5].

To validate our off-line routing solver, we instantiated the ILP problem for our four topologies and solved them using Gurobi optimizer ver. 6.5. The formulation set up was performed by using the Python API. The corresponding code is available on the Internet [20]. Results for the off-line solver are shown in Table II. The evaluation shows that our formulation can be practically used. Considering that the foreseen usage of the formulation is during design, running times are quite small. This makes us believing that our approach could be successfully used even in much larger scenarios. Even though solving times are small, they are not suitable for an on-line use. This justifies the introduction of the specific ad-hoc on-line solver, whose algorithm is presented in [5]. To validate the on-line routing solver, we randomly generated a sequence of events for each network. Further details are described in [5].

We initialized the status of the solver with the output of the off-line solver for critical streams. Then, we run, for each network, the on-line solver on its sequence of events generated

Results (off-line)			
	gurobi execution time	number of observed streams	max %bw on edge
1	12s	764	97.795%
2	30s	1100	62.060%
3	33s	869	98.058%
4	421s	2087	99.455%

TABLE II: Results for the off-line routing solver.

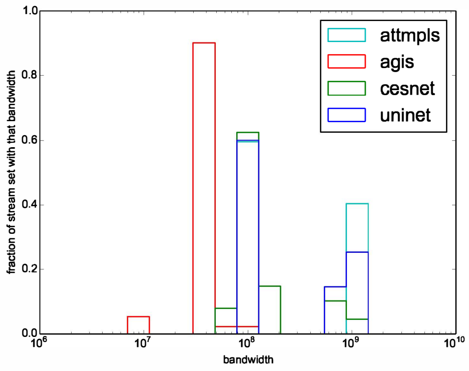

Fig. 2: Density of bandwith assigned to streams for each topology.

as described above. A density diagram is shown in Fig. 2, that has on the x-axis possible bandwidth values and on the y-axis the fraction of streams that had that bandwidth assigned in our experiments. In our experiment, assigned bandwidth is always very close to the maximum of the backbone bandwidth. Sometime, if source and destination of the stream are close each other, assigned bandwith can be larger (cf. Table I).

The off-line optimization, together with the traffic shaping approach described in Section IV, ensures compliance to Requirements 1, 2, and 3. Further, the inclusion of standard streams is performed only by using the spare bandwidth of each link, thus protecting critical stream and replica streams from packet loss due to congestion (see [5]). Requirement 4 encompasses two essential aspects: fairness of bandwidth allocation and response time. Our approach handles all streams always assigning the same bandwidth to all of them and dynamically adapting it on the basis of the current needs.

VI. Conclusions

We proposed a methodology and an architecture that enable flexible adoption of one IDS (or a few of them), while keeping the possibility to mirror any stream in the network and forward it toward the IDS independently from its deployment location. While we think that our approach can be useful in many contexts, we tailored it for the usage within ICS networks, where most of the traffic flows are critical and known in advance, and occasional usage can be handled with a best effort approach. We base our work on SDN technology, which allowed us to keep a simple centrally managed network configuration. However, the integration of a distributed approach for the SDN-controller, like the one presented in [21], in our architecture, may be the subject of additional research. Further, in our solution, we statically assigned bandwidth to all critical streams, disregarding cases in which traffic is not stable over time. Better usage of the bandwidth could be achieved by taking this into account.

References

[1] I. control systems cyber emergency response team control systems security program, "Ics-cert incident response summary report 2009-2011," ICS-CERT, Tech. Rep., 2011.

[2] K. Stouffer, S. Lightman, V. Pillitteri, M. Abrams, and A. Hahn, "Guide to industrial control systems (ics) security – nist special publication (sp) 800-82 revision 2," NIST, Tech. Rep., 2015.

[3] S. Tom, D. Christiansen, and D. Berrett, "Recommended practice for patch management of control systems," *DHS control system security program (CSSP) Recommended Practice*, 2008.

[4] O. S. Specification, "Version 1.3.3 (wire protocol 0x04)," Sept 2013.

[5] R. di Lallo, F. Griscioli, G. Lospoto, H. Mostafaei, M. Pizzonia, and M. Rimondini, "Leveraging sdn to monitor critical infrastructure networks in a smarter way," Roma Tre University, Department of Engineering, Tech. Rep., 2017, https://arxiv.org/abs/1701.04293.

[6] I. A. S. MODICON, Inc., "Modbus protocol – reference guide," Tech. Rep., 1996.

[7] I. ODVA, "The common industrial protocol (cip)," https://www.odva.org/Technology-Standards/Common-Industrial-Protocol-CIP/.

[8] M. Herrero Collantes and A. Lpez Padilla, "Protocols and network security in ics infrustructures," INCIBE, Tech. Rep., 2015.

[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] R. Jin and B. Wang, "Malware detection for mobile devices using software-defined networking," in *Proceedings of the 2013 Second GENI Research and Educational Experiment Workshop*, ser. GREE '13. Washington, DC, USA: IEEE Computer Society, 2013, pp. 81–88. [Online]. Available: http://dx.doi.org/10.1109/GREE.2013.24

[11] R. Skowyra, S. Bahargam, and A. Bestavros, "Software-defined ids for securing embedded mobile devices," in *High Performance Extreme Computing Conference (HPEC), 2013 IEEE*. IEEE, 2013, pp. 1–7.

[12] P. K. Shanmugam, N. D. Subramanyam, J. Breen, C. Roach, and J. Van der Merwe, "Deidtect: towards distributed elastic intrusion detection," in *Proceedings of the 2014 ACM SIGCOMM workshop on Distributed cloud computing*. ACM, 2014, pp. 17–24.

[13] C. Jeong, T. Ha, J. Narantuya, H. Lim, and J. Kim, "Scalable network intrusion detection on virtual sdn environment," in *Cloud Networking (CloudNet), 2014 IEEE 3rd International Conference on*. IEEE, 2014, pp. 264–265.

[14] M. Roughan, M. Thorup, and Y. Zhang, "Traffic engineering with estimated traffic matrices," in *Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement*. ACM, 2003, pp. 248–258.

[15] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network Flows: Theory, Algorithms, and Applications*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1993.

[16] I. F. Akyildiz, A. Lee, P. Wang, M. Luo, and W. Chou, "A roadmap for traffic engineering in sdn-openflow networks," *Computer Networks*, vol. 71, pp. 1 – 30, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128614002254

[17] G. Béla, "Networked Critical Infrastructures: Secure and resilient by design," in *The Proceedings of the EUROPEAN INTEGRATION BETWEEN TRADITION AND MODERNITY Congress*, vol. 6, 2015, pp. 753–760.

[18] S. Agarwal, M. Kodialam, and T. Lakshman, "Traffic engineering in software defined networks," in *INFOCOM, 2013 Proceedings IEEE*. IEEE, 2013, pp. 2211–2219.

[19] University of Adelaide, "The Internet topology zoo," Jan 2015. [Online]. Available: http://www.topology-zoo.org

[20] "**Companion Website with code**," https://bitbucket.org/sdnci/sdn-ci/.

[21] A. Tootoonchian and Y. Ganjali, "Hyperflow: a distributed control plane for openflow," in *Proceedings of the 2010 internet network management conference on Research on enterprise networking*, ser. INM/WREN'10. Berkeley, CA, USA: USENIX Association, 2010, pp. 3–3. [Online]. Available: http://dl.acm.org/citation.cfm?id=1863133.1863136