Formally Verified Latency-aware VNF Placement in Industrial Internet of Things

Guido Marchetto*, Riccardo Sisto*, Jalolliddin Yusupov*, Adlen Ksentini†
* Politecnico di Torino
Email: {name.surname}@polito.it
† Eurecom
Email: adlen.ksentini@eurecom.fr

***Abstract*—The innovative applications of 5G core technologies, namely Software Defined Networking (SDN) and Network Function Virtualization (NFV), are the key enabling technologies of industrial Internet of things (IIoT) to improve data network robustness. In the industrial scenario, with strict demands on end-to-end latency and reliability during critical events, these technologies can be leveraged to construct chains of network functions (service graphs) characterized by guarantees about latency, jitter, packet loss or redundancy. Moreover, real-time monitoring techniques provided by network virtualization help in mitigating critical events (e.g. failures or network attacks), which can be faced by updating the service graph and imposing new policies in the network. In practice, the distributed and safety-critical nature of IIoT applications requires both an intelligent placement of services across physically separated locations, which has a direct impact on latency, and a proper policy enforcement system, which guarantees service reliability, safety, and security. This paper considers both aspects by proposing a novel Virtual Network Function (VNF) placement solution for IIoT that minimizes the overall latency and, at the same time, also verifies that network-wide policies such as connectivity or isolation hold between the endpoints. In particular, this work relies on recent advances in SMT (Satisfiability Modulo Theories) solvers, which are being enhanced to solve the Maximum Satisfiability (MaxSAT) problem.**

I. Introduction

A. NFV and SDN in Industry

Industrial Internet of Things (IIoT) is gaining broader interest because it can improve the effectiveness and efficiency of modern industrial production and applications. IIoT brings together the worlds of industry and network connectivity, by making everything in the industry connected, monitored and controllable.

The key to manage millions of distributed, intelligent, and autonomous machines in a flexible way is the use of NFV (Network Function Virtualization) and SDN (Software Defined Networking). Both of them use network abstractions: SDN separates network control functions from network forwarding functions, while NFV virtualizes network services and abstracts them from dedicated hardware. Together they enable programmable connectivity, rapid service provisioning, chaining, and cyber-security.

B. Problem description, Motivation, and Contribution

The combination of new services, devices, protocols, and standards makes IIoT systems very complex and vulnerable to a wide range of attacks or cyber threats - like any other complex distributed system is. In addition, IIoT systems are safety-critical, which makes the need for security even more compelling. There are two main causes of threats [1], which are the major research points addressed in this paper: (1) *the absence or weakness of resiliency controls* which can lead to a stream of failures in cyber-attacks, (2) *the misconfiguration of Virtual Network Functions (VNFs)* which can cause, for instance, unreachability, inconsistency, and broken security tunnels.

By leveraging the SDN/NFV-enabled networks, IIoT can benefit greatly from effective countermeasures dynamically placed so as to mitigate security threats caused by point (1). Given that the virtual functions can be orchestrated and combined as service function chains between the industrial edge, endpoint devices (Master Terminal Units (MTUs), Remote Terminal Units (RTUs), Programmable Logic Controllers (PLCs), Intelligent Electronic Devices (IED), Human Machine Interfaces (HMI), smart meters, etc.), the resilience delivery problem comes down to select a proper set of security functions and to place them across different substrate nodes while meeting ultra-reliable low-latency communications' (URLLC) requirements. On the other hand, misconfiguration of VNFs discussed in point (2) can be detected either by straightforward simulation/testing or by formal verification techniques. Since soundness and completeness are not provided by simulation, formal methods are the most suitable solution, also considering that IIoT systems are safety-critical.

In this paper, we focus our attention on the so-called Virtual Network Embedding (VNE) problem, or simply VNF placement problem, while we leave the selection of the security functions to be placed in the chain as a separate component to be studied in the future. In addition, we consider the requirement of providing formal assurance that the selected function chain correctly implements the required security policies and that, at the same time, the latency requirements are met. In particular, we formulate the involved optimization and verification problems and we solve them through the use of the MaxSAT[2] approach. The tool needs some inputs: (*i*)

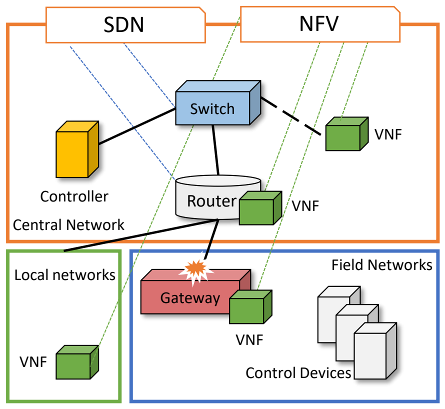


Fig. 1. NFV/SDN enabled attack mitigation scheme

proper models of VNFs representing both their forwarding behavior and their configuration parameters, (*ii*) a model of the substrate network, and (*iii*) the resiliency policies that must hold in the industrial network. Given these inputs, it generates a formally verified placement plan. Even though the two problems of placement and of formal verification are widely covered separately in the existing literature, to the best of our knowledge our approach is the only existing one for IIoT systems that solves both problems in ”one shot” by merging these two concepts together.

The IIoT paradigm finds application in many different domains, such as Industry 4.0, smart grids, smart production, and smart logistics. The Smart Grid, a concept for modernizing power systems by integrating information, telecommunications, and automation, is one of the most appealing applications in the list. Thus, a Smart Grid system has been identified as a representative example of industrial networks and adopted as a use-case in this paper to evaluate our approach.

The remainder of this paper is organized as follows. We briefly discuss the related work in Section II. We present our methodology in Section III, which is further developed in Section III-A for the part that pertains to VNF placement and in Section III-B for the part that pertains to formal verification. In Section IV we evaluate our methodology by means of a Smart Grid use case. Section V concludes the paper.

II. Related work

The classical literature on VNE, mapping each VNF to specific nodes and links in the substrate network, does not take into consideration the verification of network-wide properties during the optimization phase. These two notions do not cooperate with each other and constitute two completely separate worlds. In this section, we provide a precise discussion of the most relevant literature on these separate subjects.

A. Formal verification

Many approaches and methods for static network analysis have been proposed ([3], [4], [5], [6], [7]). Verigraph [3] requires to model complex network scenarios as sets of First Order Logic (FOL) formulas and uses Z3 [8], a Satisfiability Modulo Theories (SMT) solver, to verify satisfiability of these formulas. Concerning the specific industrial context, [7] verifies firewall configurations against security policies in SCADA systems. Instead, [6] discusses the idea of formal analysis as a technical approach that can effectively analyze Smart Grid security and resiliency. In particular, it uses Z3 to verify the system with respect to the given resiliency specifications. In contrast to our work in this paper, where the Service Graph (SG) comprises a general set of industrial devices and network middleboxes, in [6] communications only involve the SCADA physical devices in the Smart Grid. Whereas, SmartAnalyzer [9] proposes an automated security analysis tool for industrial networks, which models a number of middleboxes (e.g., Router, Firewall) and endpoint devices. This tool uses the Yices [10] SMT solver as core analysis engine.

B. VNF placement

The other wave of literature on the related subject covers VNE. These approaches can be categorized into different groups based on the featured settings. Taking into account the NP-hardness of the VNE problem, exact ([11], [12], [13]) and heuristic-based ([14], [15], [16], [17]) solutions have been used. VNE consists of finding the optimal solution relative to a particular objective. With respect to the importance of end-to-end delay in IIoT applications, as it reflects the real-time capability of the system, we further classify research efforts depending on whether the minimization of the overall latency is the main objective ([13], [18], [16], [15], [17]) or not ([11], [12], [14]).

In fact, IIoT applications have strict requirements of reliability and latency where the dynamic control allows 1-100 ms ([19], [20]) of end-to-end latency and where the communication range is up to a few kilometers. In [17] the authors formulate the placement problem of VNFs across a set of distributed data centers (DC) to realize a minimum end-to-end latency along the delivery path. The considered use-case is based on a realistic network connecting 11 DCs. SECaaS [13] discusses cloud-based industrial management solutions that consider geographically distributed private clouds. Similarly, the authors formulate the service placement problem to minimize the latency between servers and client nodes. The Abilene topology ([21]) is used as data center backbone network, where security functions for an advanced metering infrastructure (AMI) concentrators are deployed. The edges that connect different nodes in the DC backbone are weighted with propagation delays, which are assumed to be proportional to the Euclidean distances between these nodes.

The existing literature formulates the VNE problem using Integer Programming (IP) formulation, which is limited to a set of linear constraints over binary, integer, or real variables. This restriction does not apply for the MaxSAT formulation adopted in this paper, which allows us to model the problem more directly and using very expressive constraints.

C. NFV and SDN

The representative use cases of industrial networks presented in this section highlight the benefits of SDN/NFV-enabled networks to deploy enhanced, reactive security mechanisms. A DDoS Attack Mitigation Framework is proposed to defend critical industrial systems against attacks [22], [23]. In presence of anomalies in a network, the NFV orchestrator decides appropriate countermeasures by introducing new security functions into the original service graph. Then the necessary VNFs are allocated and instantiated to handle the attack traffic. Consequently, the SDN module sets the rerouting paths to connect those VNFs (Figure 1).

To demonstrate the SDN and NFV ecosystem, VirtuWind [24] has chosen a wind park control network as a key industrial application. Firewalls, IDS, DPI, Honeypots, and Honeynets are the examples of security functions that are combined interchangeably as an ordered list of services depending on the needs. These services help to analyze potential attacks and isolate attackers to protect the wind park industrial network.

The literature review reveals that the approaches addressed by the research community lack appropriate mechanisms that glue together the optimal placement and verification of a number of safety and security-related properties of the orchestrated virtual networks.

III. Methodology

Our target is to develop a solution to provide formal verification and optimal placement of VNFs, with specific reference to the IIoT environment.

By making use of an intensive modeling approach, we formulate the placement and verification problem with propositional logic formulas in Conjunctive Normal Form. The goal of verification is then to find a truth assignment for all the logical variables of the model that makes true all the logic formulas (clauses) in the network model, if one exists. This is the traditional form of the well-known Satisfiability (SAT) problem, where all the formulas are "hard" clauses and must be satisfied. This set of hard clauses represents in our case the VNF forwarding behavior models, the reachability/isolation properties that we want to ensure, and the hard constraints we have on placement (e.g. we cannot exceed the resources available in each infrastructure node).

This methodology can be further extended by introducing optimization goals, thanks to the MaxSAT approach, which gives us the possibility to go further and introduce a set of "soft" clauses in addition to hard clauses, which can be falsified if necessary. The use of soft clauses is important in modeling the VNF placement problem. For instance, the placement of a VNF on a particular substrate node can be modeled as a soft clause, meaning that this clause can be falsified in favor of placing the VNF on a different node. Finally, the tool looks for a truth assignment that satisfies all hard clauses and as many soft clauses as possible. Further, we exploit the possibility to have *weights* assigned to soft clauses, in order to find a truth assignment to the propositional variables that maximizes the total weight of satisfied clauses.

TABLE I
Summary of key notations

Symbols	Notations
$G^s = (N^s, L^s, A^s_V, A^s_L)$	Substrate network
$G^s = (N^v, L^v, A^v_V, A^v_L)$	Virtual network
N^s, E^s, L^s	Set of substrate nodes/endpoints/links
N^v, E^v, L^v	Set of VNFs to be allocated/endpoints/links
A^s_N, A^s_L	Attributes of substrate nodes/links
A^v_N	Attributes of virtual functions
$l^s_{j,k}$	Link between substrate nodes indexed by j and k
$n^v_i \uparrow n^s_j$	VNF n^v_i is hosted on substrate node n^s_j
$x_{i,j}$	Boolean variable, true if a virtual function x_i is mapped onto substrate node j
y_i	Boolean variable, true if substrate node is in use
$Soft(c, w)$	Clause c is a soft clause with weight w
$route(v^v_i, v^v_{adj}, l^s)$	True if the adjacent neighbor of v^v_i is v^v_{adj} and it is reached via link l^s

This enables the introduction of an optimized placement in conjunction with its formal verification, our target in this paper. Clauses are given as inputs to the z3Opt [25] engine, which allows us to solve these optimization objectives in addition to check the satisfiability of the formulas. Giving the considered scenario, we use weights corresponding to the propagation delay (latency) of the substrate links and the cost of selecting the substrate node, which are of interest for an IIoT system. However, it is worth noticing how the methodology is more general and might be applied in other contexts with different interesting parameters (i.e., weights) to optimize.

In the rest of this section we firstly describe how the VNF placement problem is formalized, starting from an introduction of the mathematical models associated to a substrate and a virtual network. Then, we present the adopted formal verification methodology together with the formalization of network policies and VNF models used for this purpose.

A. VNF placement

Substrate network. Similar to previous works in [26], [27], the backbone network of cloud data centers - substrate network, is modeled as a weighted undirected graph and denoted by $G^s = (V^s, L^s, A^s_V, A^s_L)$, where $V^s = N^s \cup E^s$ is the set of vertexes, made up of the two disjoint subsets N^s (substrate nodes) and E^s (substrate endpoints), L^s is the set of edges (representing the links that connect substrate nodes and endpoints with one another), while A^s_V and A^s_L are the sets of values that can be taken by the attributes of the vertexes and edges, respectively. In this paper, v^s ranges over V^s, n^s ranges over N^s, e^s ranges over E^s, and l^s ranges over L^s. The attributes of a vertex v^s may include CPU capacity, memory, storage capacity, buffer size, geographical location, etc. And the attributes of a link may include bandwidth, latency, bit error rate, etc. In this paper, we consider for simplicity only one substrate node attribute (storage capacity) and only one link attribute (latency), but the extension to multiple attributes is straightforward. Therefore, each substrate node n^s is associated with the available storage capacity $storage(n^s) \in A_V^s$, while each substrate link l^s is associated with the introduced latency $latency(l^s) \in A_L^s$. We associate vertexes with integer indexes that uniquely identify them, and we use the notation v_j^s for the substrate vertex with index j. Depending on the type of the vertex this may correspond to a substrate node n_j^s or substrate endpoint e_j^s. Also, we use the notation $l_{j,k}^s$ to denote the link between the vertexes indexed by j and k.

Service Request. We model a virtual network service request as another directed graph similar to the one that describes the substrate network, denoted $G^v = (V^v, L^v, A_V^v, A_L^v)$. The set of vertexes V^v is partitioned into the two disjoint subsets N^v (the VNFs n^v to be allocated) and E^v (the endpoint VNFs e^v, whose allocation on the substrate network is assumed to be fixed). The direction of the edges denote the outgoing and the incoming packet flows respectively. Similarly to what is done for the substrate network model, for each VNF n^v to be allocated we consider the required storage capacity, denoted $storage(n^v) \in A_V^v$, the processing delay introduced, denoted $latency(n^v) \in A_V^v$, and its functional type, denoted $func(n^v) \in A_V^v$. Instead, edges are assumed to have no attributes (i.e., A_L^v is empty). We introduce integer indexing for vertexes of the virtual network too, with the same notation used for the substrate network.

Upon the arrival of a service request G^v, an orchestrator component of NFV (Figure 1) must decide how to optimally allocate the VNFs of G^v onto the substrate network nodes. In our case, this problem is combined with the problem of verifying that a number of reachability/isolation properties are satisfied by the virtual network, with a given configuration of the VNFs. The mapping of the endpoint VNFs is assumed to be already specified in the service request.

Problem Formalization. To formalize the VNE problem, we introduce boolean variables y_i and $x_{i,j}$ that take true value when substrate node n_i^s is in use and when VNF n_i^v is hosted on substrate node n_j^s, respectively. This last predicate is also denoted $n_i^v \uparrow n_j^s$. The mapping of a service request is then represented by two mapping functions: M_n, which maps VNFs of the service request onto substrate nodes that meet their resource requirements, and M_e, which maps endpoints. M_n can be formally defined as follows. For all $n^v \in N^v$

$$M_n(n^v) = n^s, \tag{1}$$

subject to $n^s \in N^s$, and $n^v \uparrow n^s$, and, for each j such that $n_j^s \in N^s$,

$$\left(\sum_{\forall i | n_i^v \uparrow n_j^s} storage(n_i^v) * x_{i,j} \right) \leq storage(n_j^s) * y_j \tag{2}$$

where we are assuming the $true$ value of x_{ij} and y_j corresponds to 1, while their $false$ value corresponds to 0.

Equation 2 specifies that the sum of all storage required by VNFs allocated on a substrate node should be less than or equal to the storage available on that substrate node. Here we are assuming that VNFs from the same service request can share the same substrate node, which is common in NFV systems, e.g., in order to reduce latency.

Given this formalization, we build the set of clauses to represent M_n as follows. First of all we include, as hard clauses, the inequalities in (2).

In addition to these inequalities, we need to represent explicitly that M_n is a function, i.e. it maps each VNF onto exactly one node. For each i such that $n_i^v \in N^v$, this constraint is expressed by the following equation

$$\sum_{\forall j | n_j^s \in N^s} x_{i,j} = 1 \tag{3}$$

Finally, for each j such that $n_j^s \in N^s$, in order to correctly bind variable y_j to variables $x_{i,j}$, we add the implication

$$y_j \implies \bigvee_i x_{i,j} \tag{4}$$

i.e., when substrate node n_j is in use, there is at least one VNF deployed on this node.

Routing tables. The network behavior of the virtual service is modeled by a set of formulas that represent the routing tables of each network function involved in the service request. These formulas express the next hops - next gateways to which packets must be forwarded along the path to their final destination. For each VNF v_i^v and its adjacent one-hop neighbor v_{adj}^v, we define a predicate $route(v_i^v, v_{adj}^v, l_{j,k}^s)$ which is true if the adjacent neighbor of v_i^v is v_{adj}^v and it is reached via link $l_{j,k}^s$ of corresponding j and k substrate nodes.

The routing table of the endpoint VNF e_n^v in the SG is formulated as a set of soft clauses, with the negative form of the link latency as the weight. In this way, the MaxSAT solver will minimize the overall latency of the chosen path in the infrastructure. As the location of e_0^v is fixed in the substrate endpoint e_0^s, we generate the following soft constraint for each possible substrate node n_k^s onto which n_{adj}^v (adjacent neighbor VNF in the SG) can be allocated

$$Soft((route(e_0^v, n_{adj}^v, l_{0,k}^s) \implies x_{adj,k}), -latency(l_{0,k}^s)) \tag{5}$$

where the notation $Soft(c, w)$ specifies that clause c is a soft clause with weight w.

In practice, the routing table of the endpoint VNF specifies to which substrate node k a packet is forwarded depending on the allocation of the next VNF in the SG.

The soft clauses related to the other VNFs $n_i^v \in N^v$ in the SG, with $i > 0$, are formulated similarly:

$$\begin{aligned} Soft((route(n_i^v, n_{adj}^v, l_{j,k}^s) \implies x_{i,j} \wedge x_{adj,k})), \\ -latency(l_{j,k}^s)) \end{aligned}$$

i.e., if VNF i forwards packets to the adjacent VNF adj in the service graph through link $l_{j,k}$, then the corresponding boolean variables $x_{i,j}$ and $x_{adj,k}$, which indicate the locations of the VNFs, must be true. If two VNFs are allocated onto the same substrate node, i.e., $j = k$, we have $latency(l_{j,k}^s) = 0$, and a soft clause with weight equal to zero is added to the set.

Configuration parameters of VNFs allow us to model a fixed processing delay for each VNF. This is represented by the $latency(n^v)$ function, which can be used in order to include the processing delay of the given VNF when computing the overall end-to-end latency. If we have an upper bound on the overall end-to-end latency that must be guaranteed in the system, we can formulate it as an additional hard clause.

Optimization Objectives. VNE is a multi-objective optimization problem. From a network infrastructure perspective, as many service requests as possible should be mapped onto the substrate network, making efficient use of the substrate network resources. However, the industrial environment usually requires minimization of link propagation delay between the endpoints too. Accordingly, the objective function of our formulation has two goals: to minimize the number of substrate nodes in use and to minimize network latency.

The soft clauses involving the $route$ predicates cause the solver to minimize latency. In order to minimize the number of substrate nodes in use we add the following additional soft clause for each substrate node $n_i^s \in N^s$:

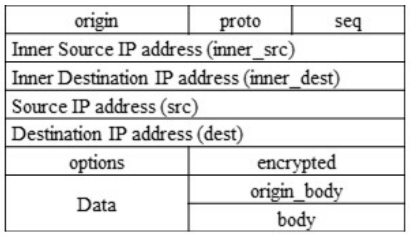
$$Soft(\neg y_i, K)$$

where K is a constant selected according to whether we want to give priority to latency minimization or to number of substrate nodes in use minimization, where the larger K means priority to minimize use, while lower K means priority toward latency minimization. The MaxSAT solver attempts to assign $false$ values to the boolean variables y_i in order to minimize the penalty for falsified clauses in the current model, thus minimizing the number of nodes in use. Then, if we feed the set of formulas defined so far along with models in Section III-B to the MaxSAT solver, it returns, if possible, a model that satisfies all hard clauses, including the ones about reachability/isolation (see Section III-B), while minimizing latency and the number of nodes in use.

B. *Formal Verification*

To satisfy the URLLC requirements of the IIoT, we focus on security as the main parameter to ensure reliability. As mentioned in Section I, misconfigurations in the network are the cause of major network failures such as reachability problems, security violations, and network vulnerabilities. Thus, the aim of the verification process in this paper is to analyze inconsistencies and misconfigurations between two or more devices in the industrial networks.

As described in Section II, many techniques and tools exist for a formal analysis of the forwarding behavior of a service graph. Among the others, the modeling approach of the Verigraph tool presented in [3] is particularly interesting for our work as it is completely compatible with the z3Opt solver. In essence, the set of logic formulas defined in Verigraph corresponds to a set of the hard clauses mentioned in Section III-A and then can be easily solved by z3Opt together with the other clauses previously defined. In particular, by means of this approach we can statically analyze network configurations of the SG to check the satisfiability of network policies such as



origin	proto	seq
Inner Source IP address (inner_src)		
Inner Destination IP address (inner_dest)		
Source IP address (src)		
Destination IP address (dest)		
options	encrypted	
Data	origin_body	
	body	

Fig. 2. A static structure of packet fields

reachability or isolation. Given a policy, Verigraph attempts to find appropriate values to uninterpreted functions and constant symbols that constitute the formal model of the network in order to satisfy the policy. If the solver cannot produce a model, a configuration error is detected, and the property is said to be unsatisfiable.

The work presented in [3] must be extended in order to be applied in our solution. First, we need to modify the presented packet forwarding model to make the approach more efficient. In particular, we can eliminate the notion of quantitative time that is currently used in Verigraph models in order to reduce the size of the problem and thus speedup the verification process in complex scenarios. This is key in a flexible and reconfigurable scenario like IIoT. Second, we need to introduce new VNF models in addition to the existing catalog in order to address specific industrial applications that are currently not covered.

In the following we present the forwarding model of the network employed in this paper, as derived by the Verigraph approach, followed by some examples of VNF models that are relevant in our context. We model the network as a set of network nodes that send and receive packets. Each packet has a static structure (Figure 2) of fields:

- *src* and *dest* are the source and destination addresses of the current packet;
- *inner_src* and *inner_dest* are the ultimate source and destination addresses of the encapsulated packet;
- *origin* represents the network node that has originally created the packet;
- *origin_body* takes a trace of the original content body of the packet, while *body* is the current body (that could be modified traversing the chain);
- *seq* is the sequence number of this packet;
- *proto* represents the protocol type;
- *options* are the options values for the current packet;
- *encrypted* is used to represent if the packet is encapsulated in another packet or not.

We also use a set of functions for retrieving information. All these functions are uninterpreted functions supported by the solver, which allow any interpretation that is consistent with the constraints over the function.

- *Bool nodeHasAddress(node, address)*, which returns true if *address* is an address associated to *node*;
- *Node addrToNode(address)*, which returns the node associated to the passed *address*;

- *Int sport(packet)* and *Int dport(packet)*, which return respectively the source and destination ports of *packet*.

The main functions that model operational behaviors in a network are:

- *Bool send(node_src, node_dest, packet)* which returns true if source node *node_src* can send packet *packet* towards destination node *node_dest*;
- *Bool recv(node_src, node_dest, packet)* which returns true if destination node *node_dest* can receive packet *packet* from source node *node_src*.

Ultimately, the uninterpreted functions are the means to impose conditions for describing how network and VNFs operate.

The general forwarding behavior of a network can be expressed by means of the following set of conditions imposed on those two functions:

$$\begin{aligned} send(n_0, n_1, p_0) \implies & (n_0 \neq n_1 \land p_0.src \neq p_0.dest \land \\ & sport(p_0) \geq 0 \land sport(p_0) < MAX\_PORT \land \\ & dport(p_0) \geq 0 \land dport(p_0) < MAX\_PORT \land), \\ & \forall n_0, p_0 \end{aligned} \tag{6a}$$

$$recv(n_0, n_1, p_0) \implies send(n_0, n_1, p_0), \quad \forall n_0, p_0 \tag{6b}$$

Formula 6a states that the source and destination nodes (n_0 and n_1) must be different, as well as the source and destination addresses in the packet ($p_0.src$ and $p_0.dest$). The source and destination ports must also be defined in a valid range of values. If a packet is received by a node (n_1), this implies that the packet was sent to this node. This is expressed by Formula 6b.

To verify the correctness of reachability properties in presence of such functions, we must also assume that the original sent packet could be different from the received one. Hence, we can verify the reachability between the src and $dest$ nodes in presence of a set of middleboxes thanks to the following formula:

$$\exists(n_0, p_0) \mid recv(n_0, dest, p_0) \land p_0.origin == src \tag{7}$$

Here we are modeling the case of a source node (src) that is sending a packet to a destination node ($dest$) of which we want to check the reachability. The destination node may receive a different packet from the one sent, because VNFs could modify the sent packet in its trip towards the destination. Thus we must impose that the destination node receives a new packet (p_0) from the last node (n_0): the received packet must have the source node as origin ($p_0.origin == src$).

On the contrary, the isolation property states that a packet sent from a source node (src) must not be received by a destination node ($dest$), as shown in Formula 8a. This can be expressed as for all the packets received by the destination node, the origin of the packets must not be equal to the source node ($p_0.origin \neq src$). Lastly, as shown in Formula 8b, we must assert that there is a packet sent by the source node and the destination address of the packet is the destination node ($dest$).

$$\forall(n_0, p_0) \mid recv(n_0, dest, p_0) \implies p_0.origin \neq src \tag{8a}$$

$$\exists(n_1, p_1) \mid send(src, n_1, p_1) \land nodeHasAddress(dest, p_1.dest) \tag{8b}$$

In addition to the above clauses, it is necessary to impose formulas describing the specific behavior of involved VNFs. In the following we present as an example some of the VNFs models used to model the middleboxes of the IIoT scenario we chose as an example.

Endpoint. An endpoint is an industrial device that sends packets towards another endpoint and receives packets from it. The sent packets must satisfy the conditions expressed in (9a): *(i)* the endpoint address is the source address; *(ii) origin* is the endpoint itself; *(iii) origin_body* and *body* must be equal. The received packet must have the endpoint address as destination, as expressed in (9b).

$$\begin{aligned} (send(endpoint, n_0, p_0)) \implies & nodeHasAddress(endpoint, p_0.src) \land \\ & p_0.origin == endpoint \land \\ & p_0.origin\_body == p_0.body \land \\ & predicatesOnPktFields, \quad \forall(n_0, p_0) \end{aligned} \tag{9a}$$

$$\begin{aligned} (recv(n_0, endpoint, p_0)) \implies & (nodeHasAddress(endpoint, p_0.dest)), \\ & \forall(n_0, p_0) \end{aligned} \tag{9b}$$

This is a basic version of an endpoint in the service graph, which can be configured to behave as an endpoint-based model (i.e., MTU, RTU, PLC, IED, HMI and smart meters). Then, endpoint configurations are the means to specify which traffic flow endpoints send, without changing their basic model (e.g., a SCADA client can generate a packet with a specific port number, destination address etc.). Initially $predicatesOnPktFields$ is set to true and, depending on the packet model configured by the user, this predicate will be appended with the assigned fields of the packet (e.g., $predicatesOnPktFields = predicatesOnPktFields \land p_0.body = 00$).

ACL firewall model. An ACL firewall is a simple firewall that drops packets based on its internal Access Control List (ACL), configured when the service model is initialized. In particular, the ACL list is managed through the uninterpreted function $acl\_func(src, dest, sPort, dPort, Proto)$ that filters on five attributes. A possible interpretation is given by (10), when the ACL list contains two entries, like for example $ACL = [< src_1, dest_1, *, *, * >]$.

$$acl\_func(a, b, c, d, e) == (a == src_1 \land b == dest_1), \forall(a, b, c, d, e) \tag{10}$$

A negation of the function represents the "blacklisting" approach of the firewall. Hence, if an ACL firewall sends a packet, it means that it has previously received a packet of which the source and destination addresses are not contained in the ACL list:

$$\begin{aligned} send(fw, n_0, p_0) \implies (\exists (n_1) | recv(n_1, fw, p_0) \wedge \\ \neg acl\_func(p_0.src, p_0.dest, sport(p_0), dport(p_0), p_0.proto), \\ \forall (n_0, p_0) \end{aligned} \tag{11}$$

IDS. Intrusion detection system (IDS) function monitors a network for malicious activity or policy violations. We model the simplest IDS network function that also acts as an intrusion prevention system function. This function is able to analyze each received packet and to look for matches with a set of filtering rules in the blacklist, e.g., involving Modbus/TCP protocol-specific function codes. As majority of SCADA systems use plain text (HEX) communications, they are susceptible to attacks, allowing the insertion of illegitimate system commands by an attacker. Let us suppose the IDS must block the Modbus diagnostic messages with function code 00 and 43 that are used to cause denial of service attacks in [28]. In this case the blacklist can be defined as *BlackList=[00 (return query data), 43 (read device identification)]*, which is represented in (12) as a formula imposing that the *isInBlackList()* function returns *TRUE* if $(body == 00)$ or $(body == 43)$ is *TRUE*.

$$\begin{aligned} (isInBlackList(body) == (body == 00) \vee \\ (body == 43)), \forall body \end{aligned} \tag{12}$$

Formula (13) states that the IDS forwards a packet only if this packet was received and it does not contain the blacklisted packet body.

$$\begin{aligned} send(ids, n_0, p_0) \implies \exists (n_1) \mid recv(n_1, ids, p_0) \wedge \\ \neg isInBlackList(p_0.body), \forall (n_0, p_0) \end{aligned} \tag{13}$$

Conjunction of the placement constraints (III-A) with the forwarding behavior of the network (III-B) represents the overall model of the system. By feeding the solver with this input, we obtain the placement plan that ensures the network-wide properties are satisfied.

IV. Use Case and Experimental Results

This section presents some experiments we run to evaluate our methodology. In order to do that, we reuse the same topology considered for the substrate network of DCs and Smart Grid devices presented in [13], where the Abeline - inspired Internet2 network topology forms the substrate network and IEEE BUS 57 test system topology hosts the Smart Grid endpoints. The experimental topologies are depicted in Figure 3. We exploit the same network topology generator, GENSEN

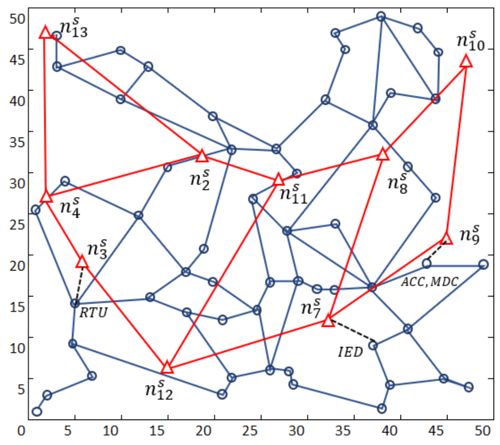


Fig. 3. Experimental topologies presented in [13] (redrawn). △ an augmented backbone network of 10 data centers and ◯ IEEE bus test system of 57 nodes

[29] to generate a realistic geographical distribution of grid edge endpoints. Similarly, the link latency is assumed to be directly proportional to the Euclidean distances between the endpoints and substrate nodes.

The IEEE Bus 57 Test Case of the American Electric Power System in the Midwestern, US, includes several sections representing information from different devices in the power grid. Buses are nodes in the network or substation locations, and branches are the connections between buses. Each power system bus works as a gateway router that is connected to the closest substrate network node either through wired or wireless links (e.g., xDSL, LTE). The gateway routers aggregate traffics from endpoint VNFs to be forwarded to other endpoints or substrate nodes throughout the substrate network. In the evaluation phase of the tool, we select random nodes from the test system topology and map the endpoint VNFs of the service on them. We illustrate the physical connections that link Smart Grid nodes to substrate nodes with dashed lines, as shown in Figure 3.

We consider an initial scenario where the service graph (represented in Figure 4(a)) consists of SCADA commodity devices to perform various grid control applications, SCADA slaves that interact with the control devices, and security functions in between. For this example, endpoints correspond to the Automation Control Center (ACC), Metering Data Center (MDC), Remote Terminal Unit (RTU) and Intelligent Electronic Device (IED) in the Smart Grid network. These endpoints are connected through an encrypted channel with the help of two VPN termination network functions n_1^v and n_2^v. Additionally, there is an IDS network function n_3^v between the endpoints ACC, MDC and RTU.

In our service model, we assume the location of the endpoints to be fixed and associated to specific substrate nodes (i.e., they are not considered in the placement procedure), whereas the network functions need to be placed in the substrate network. Concerning the verification aspects, we define two reachability policies in this SG from RTU to ACC and from MDC to IED.

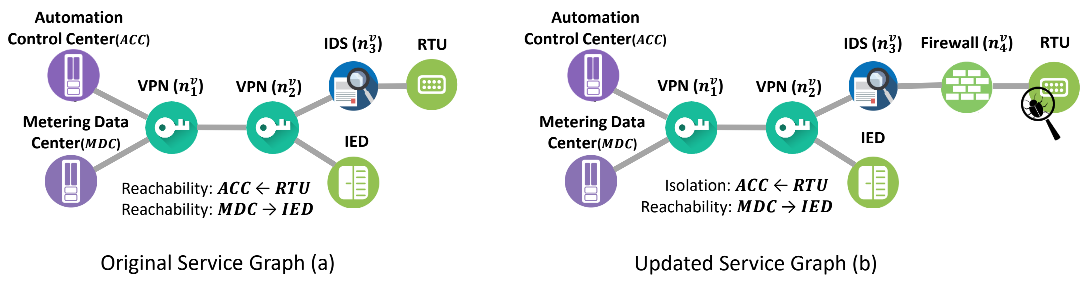

Automation Control Center(ACC)
Metering Data Center(MDC)
VPN (n_1^v)
VPN (n_2^v)
IDS (n_3^v)
RTU
IED
Reachability: $ACC \leftarrow RTU$
Reachability: $MDC \rightarrow IED$
Original Service Graph (a)
Automation Control Center(ACC)
Metering Data Center(MDC)
VPN (n_1^v)
VPN (n_2^v)
IDS (n_3^v)
Firewall (n_4^v)
RTU
IED
Isolation: $ACC \leftarrow RTU$
Reachability: $MDC \rightarrow IED$
Updated Service Graph (b)

Fig. 4. Examples of Service Graph

On the assumption of compromised RTU in the field network, NFV orchestrator proposes an updated service graph depicted in Figure 4(b) to mitigate the impact of cyber-attacks. This updated SG includes a firewall VNF n_4^v to block packets of RTU device from proceeding to the control center, while the reachability requirement between ACC and RTU is converted to an isolation policy.

In this section, we present results of the updated graph (b) that has different placement in the substrate network compared to the original graph (a). The high-level representation of the updated service that includes both aspects related to the infrastructure (i.e., which network functions implement the service, how they are interconnected among each other) is fed as an input to our tool along with the configurations of these network functions following the notation given in Section III-B. The list of these configurations is given below:

- VPN access n_1^v: IP address of the VPN exit n_2^v gateway
- VPN exit n_2^v: IP address of the VPN access n_1^v gateway
- IDS n_3^v: not allowed function code $\{43\}$
- Firewall n_4^v: {src:RTU dst:ACC sport:$*$ dport:$*$ proto:$*$}
- ACC: generates a packet with a function code different from $\{43\}$

The required storage capacities of each VNF are integers with a uniform distribution between 10 and 50, whereas the available storage capacity of each substrate node varies between 100 and 150. An overall processing delay of each VNF is randomly determined by a uniform distribution between 50-100 [30]. As the configuration parameters of the involved VNFs satisfy the new isolation property in Figure 4 (b), a latency-aware optimal placement of VNFs in substrate network generated as an output:

$$x_{1,9}, x_{2,7}, x_{3,12}, x_{4,3}, y_3, y_7, y_9, y_{12}$$

In this list, we highlighted only the variables whose value is true, which shows the allocation of the VNFs and occupancy of the substrate nodes. From this output, we can conclude that the VPN termination n_1^v is placed on the substrate node n_9^s, the VPN termination n_2^v on n_7^s, the IDS n_3^v on n_{12}^s and the firewall VNF n_4^v on the substrate node n_3^s.

We feed our tool with this updated SG including 8 VNFs on real data sets of Table II. For each substrate network topology, we use the same IEEE BUS 57 test system topology for Smart Grid endpoints. We limit our study to this data set only, for the Smart Grid, but any other data set in the "IEEE Common Data" format is applicable to our approach. The demonstrated results reflect computation time of the latency minimization problem under various conditions. Each of these scenarios consists of a substrate network and service request to be allocated. All evaluations are executed on a workstation with 32GB RAM and an Intel i7-6700 CPU.

TABLE II
COMPUTATION TIME OF DIFFERENT TOPOLOGIES

Topology	Nodes	Links	Time (O+V)	Time (O)[12]
Internet2[21]	10	13	0.6	0.029
GEANT[21]	22	36	15.4	0.1
UNIV1[31]	23	43	22.2	0.235
AS-3679[32]	79	147	35.1	3.013

It is evident from the table how the average computation time of the proposed approach for the Internet2 topology adopted from [13] is low and certainly compliant with the IIoT requirements (less than 1 sec for 10 nodes and 13 links). However, GEANT, UNIV1 and AS-3679 network topologies show significant computation overhead due to the much larger network sizes considered. It is important to note that the different combination of configurations of network functions and the number of properties to verify have a small impact on the complexity of the problem in contrast to the size of the topology.

Compared to the time of optimization (O) only given in [12], the time taken by our optimization and verification tool is higher. However, it is important to mention that the placement optimization and verification *(O+V)* of an SG is a problem fundamentally different and more complex than the simple VNE problem. APPLE solves the VNE problem in a short time, but specific aspects of NFV such as forwarding behavior, chaining, and models of network functions are not addressed by the authors of the tool. These additional details that must be considered in our work certainly introduce further time complexity. However, as part of possible future work it would be possible to develop heuristic-based algorithms to profitably consider much larger network sizes.

V. Conclusion

This paper presents joint optimization and verification model of the network services for NFV/SDN-enabled IIoT networks. This allows us to check the forwarding behavior of network service functions and provide latency-aware optimal placement in the network infrastructure. The problem is formalized as an instance of the weighted MaxSAT problem. The encoded model is fed as an input to a solver and the optimal placement of network functions in infrastructure network is obtained as an output, if the configuration of involved VNFs satisfies network properties. The output guarantees the end-to-end service delivery at minimum cost including both processing delay and network delay. To evaluate our model, we have selected the Smart Grid, as a representative use case of industrial Internet of things. We have used IEEE BUS 57 topology for the Smart Grid network and several real-world network topologies for data center backbone. As the initial results show promises in smaller instances, we plan to improve our abstract model to cope with bigger instances and use them to further scale our tool.

References

[1] M. Harvey, D. Long, and K. Reinhard, "Visualizing nistir 7628, guidelines for smart grid cyber security," in *2014 Power and Energy Conference at Illinois (PECI)*, Feb 2014, pp. 1–8.

[2] N. Bjørner, A. Phan, and L. Fleckenstein, "νz - an optimizing SMT solver," in *ETAPS 2015, London, UK, April 11-18, 2015. Proceedings*, 2015, pp. 194–199.

[3] S. Spinoso, M. Virgilio, W. John, A. Manzalini, G. Marchetto, and R. Sisto, "Formal Verification of Virtual Network Function Graphs in an SP-DevOps Context," in *Service Oriented and Cloud Computing - 4th European Conference, ESOCC 2015, Taormina, Italy, September 15-17, 2015. Proceedings*, 2015, pp. 253–262.

[4] S. Owre, J. M. Rushby, and N. Shankar, "Pvs: A prototype verification system," in *Proceedings of the 11th International Conference on Automated Deduction: Automated Deduction*, ser. CADE-11. London, UK, UK: Springer-Verlag, 1992, pp. 748–752.

[5] H. Mai, A. Khurshid, R. Agarwal, M. Caesar, P. B. Godfrey, and S. T. King, "Debugging the data plane with anteater," in *Proceedings of the ACM SIGCOMM 2011 Conference*, ser. SIGCOMM '11. New York, NY, USA: ACM, 2011, pp. 290–301.

[6] M. A. Rahman, A. H. M. Jakaria, and E. Al-Shaer, "Formal analysis for dependable supervisory control and data acquisition in smart grids," in *2016 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, June 2016, pp. 263–274.

[7] O. Rysavy, J. Rab, and M. Sveda, "Improving security in scada systems through firewall policy analysis," in *2013 Federated Conference on Computer Science and Information Systems*, Sept 2013, pp. 1435–1440.

[8] L. De Moura and N. Bjørner, "Z3: An Efficient SMT Solver," in *Proceedings of the Theory and Practice of Software, 14th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, ser. TACAS'08/ETAPS'08. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 337–340.

[9] M. A. Rahman, P. Bera, and E. Al-Shaer, "Smartanalyzer: A noninvasive security threat analyzer for ami smart grid," in *2012 Proceedings IEEE INFOCOM*, March 2012, pp. 2255–2263.

[10] B. Dutertre and L. D. Moura, "The yices smt solver," Tech. Rep., 2006.

[11] M. Alaluna, L. Ferrolho, J. R. Figueira, N. Neves, and F. M. V. Ramos, "Secure virtual network embedding in a multi-cloud environment," *CoRR*, vol. abs/1703.01313, 2017. [Online]. Available: http://arxiv.org/abs/1703.01313

[12] X. Li and C. Qian, "An nfv orchestration framework for interference-free policy enforcement," in *2016 IEEE 36th International Conference on Distributed Computing Systems (ICDCS)*, June 2016, pp. 649–658.

[13] M. M. Hasan and H. T. Mouftah, "Cloud-centric collaborative security service placement for advanced metering infrastructures," *IEEE Transactions on Smart Grid*, vol. PP, no. 99, pp. 1–1, 2017.

[14] B. Addis, D. Belabed, M. Bouet, and S. Secci, "Virtual network functions placement and routing optimization," in *2015 IEEE 4th International Conference on Cloud Networking (CloudNet)*, Oct 2015, pp. 171–177.

[15] J. C. Gonzalez, K. Hopkinson, G. Greve, M. Compton, J. Wilhelm, S. Kurkowski, and R. Thomas, "Optimization of trust system placement for power grid security and compartmentalization," in *2011 IEEE Power and Energy Society General Meeting*, July 2011, pp. 1–1.

[16] M. M. Hasan and H. T. Mouftah, "Latency-aware segmentation and trust system placement in smart grid scada networks," in *2016 IEEE 21st International Workshop on Computer Aided Modelling and Design of Communication Links and Networks (CAMAD)*, Oct 2016, pp. 37–42.

[17] B. Martini, F. Paganelli, P. Cappanera, S. Turchi, and P. Castoldi, "Latency-aware composition of virtual functions in 5g," in *Proceedings of the 2015 1st IEEE Conference on Network Softwarization (NetSoft)*, April 2015, pp. 1–6.

[18] M. M. Hasan and H. T. Mouftah, "Optimal trust system placement in smart grid scada networks," *IEEE Access*, vol. 4, pp. 2907–2919, 2016.

[19] I. Parvez, A. Rahmati, I. Güvenç, A. I. Sarwat, and H. Dai, "A survey on low latency towards 5g: Ran, core network and caching solutions," *CoRR*, vol. abs/1708.02562, 2017. [Online]. Available: http://arxiv.org/abs/1708.02562

[20] P. Schulz, M. Matthe, H. Klessig, M. Simsek, G. Fettweis, J. Ansari, S. A. Ashraf, B. Almeroth, J. Voigt, I. Riedel, A. Puschmann, A. Mitschele-Thiel, M. Muller, T. Elste, and M. Windisch, "Latency critical iot applications in 5g: Perspective on the design of radio interface and network architecture," *IEEE Communications Magazine*, vol. 55, no. 2, pp. 70–78, February 2017.

[21] S. Orlowski, M. Pióro, A. Tomaszewski, and R. Wessäly, "SNDlib 1.0–Survivable Network Design Library," in *Proceedings of the 3rd International Network Optimization Conference (INOC 2007), Spa, Belgium*, April 2007, http://sndlib.zib.de, extended version accepted in Networks, 2009.

[22] L. Zhou and H. Guo, "Applying NFV/SDN in mitigating DDoS attacks," *TENCON 2017 - 2017 IEEE Region 10 Conference*, pp. 2061–2066, 2017.

[23] A. Califano, E. Dincelli, and S. Goel, "Using Features of Cloud Computing to Defend Smart Grid against DDoS Attacks," *10th Annual Symposium on Information Assurance (ASIA15)*, no. July, p. 44, 2015.

[24] T. Mahmoodi, V. Kulkarni, W. Kellerer, P. Mangan, F. Zeiger, S. Spirou, I. Askoxylakis, X. Vilajosana, H. J. Einsiedler, and J. Quittek, "Virtuwind: virtual and programmable industrial network prototype deployed in operational wind park," *Transactions on Emerging Telecommunications Technologies*, vol. 27, no. 9, pp. 1281–1288, 2016, ett.3057.

[25] N. Bjorner and A.-D. Phan, "Z - maximal satisfaction with z3," in *SCSS 2014. 6th International Symposium on Symbolic Computation in Software Science*, ser. EPiC Series in Computing, T. Kutsia and A. Voronkov, Eds., vol. 30. EasyChair, 2014, pp. 1–9.

[26] M. Yu, Y. Yi, J. Rexford, and M. Chiang, "Rethinking virtual network embedding: Substrate support for path splitting and migration," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 17–29, Mar. 2008.

[27] X. Cheng, S. Su, Z. Zhang, K. Shuang, F. Yang, Y. Luo, and J. Wang, "Virtual network embedding through topology awareness and optimization," *Computer Networks*, vol. 56, no. 6, pp. 1797 – 1813, 2012.

[28] I. N. Fovino, A. Carcano, M. Masera, and A. Trombetta, "An experimental investigation of malware attacks on scada systems," *International Journal of Critical Infrastructure Protection*, vol. 2, no. 4, pp. 139 – 145, 2009.

[29] T. Camilo, J. S. Silva, A. Rodrigues, and F. Boavida, "Gensen: A topology generator for real wireless sensor networks deployment," in *Software Technologies for Embedded and Ubiquitous Systems*, R. Obermaisser, Y. Nah, P. Puschner, and F. J. Rammig, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 436–445.

[30] A. Basta, W. Kellerer, M. Hoffmann, H. J. Morper, and K. Hoffmann, "Applying nfv and sdn to lte mobile core gateways, the functions placement problem," in *Proceedings of the 4th Workshop on All Things Cellular: Operations, Applications, &#38; Challenges*, ser. AllThingsCellular '14. New York, NY, USA: ACM, 2014, pp. 33–38.

[31] T. Benson, A. Akella, and D. A. Maltz, "Network traffic characteristics of data centers in the wild," in *Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement*, ser. IMC '10. New York, NY, USA: ACM, 2010, pp. 267–280.

[32] N. Spring, R. Mahajan, D. Wetherall, and T. Anderson, "Measuring isp topologies with rocketfuel," *IEEE/ACM Trans. Netw.*, vol. 12, no. 1, pp. 2–16, Feb. 2004.