

Experimental Evaluation of Secure Digital Twin Model Exchange Using Distributed Ledger Technologies

P. González, M. Ruiz, and L. Velasco*

Optical Communications Group - Universitat Politècnica de Catalunya, Barcelona, Spain - e-mail: luis.velasco@upc.edu

Abstract—Secure intra-domain Digital Twin model exchange for multi-domain end-to-end modelling is realized through a Distributed Ledger Technology network. A workflow is proposed and experimentally demonstrated. Results show the impact of models' size on the performance of the system.

Keywords—End-to-end Digital Twin, Secure Model Exchange

I. INTRODUCTION

The use of digital twins (DT) for modelling optical networks can support multiple use cases e.g., quality of transmission estimation for lightpath provisioning, misconfiguration detection, and degradation detection, among others [1]. Traditionally, DTs target at modelling single domain optical networks, as they need full knowledge of the characteristics of optical devices supporting a lightpath. They can be also applied to model unknown behaviors in applications like failure identification and multivendor scenarios [2]. In addition, they could cover multidomain scenarios, where optical connectivity needs to be established between two end-points belonging to different administrative domains, each controlled by a different software-defined networking (SDN) controller.

The solution to create end-to-end (e2e) models for the entire lightpath is that the domains exchange partial models of the segments in their domain. However, that solution is subject to several vulnerabilities like tampering and model forging. A possible solution is to create a federation of domains based on distributed ledger technologies (DLT) [3]. In this paper, we follow that approach and use DLT with smart contracts (SC) for the secure exchange of DT models among optical domains. An SC is a program running in the DLT that might include a collection of code (its functions) and data (its state). Specifically, we assume OCATA as the network DT [2], since their models are based on deep neural networks (DNN) that allow for direct concatenation, which facilitates creating e2e models. However, because the significant size of the DNN models, different options, including protocol buffers, are evaluated to upload them in the specific SC created for each lightpath.

II. PROPOSED ARCHITECTURE AND WORKFLOW

Fig. 1 presents an illustrative reference network scenario with three network domains and a lightpath connecting domain D1 and domain D3 through intermediate domain D2. In the control plane, a hierarchy of systems is used with a network orchestrator on top of the domain controllers, each with an SDN controller and a DT. Note that the domain SDN control plane

can be also a hierarchy of SDN controllers with a parent SDN controller and controllers for the optical line systems. In any case, we assume that the DT in each domain has full visibility of the underlying optical resources, i.e. optical transponders (Tp) and line systems, so it is able to accurately model optical transmission within the domain. Domain DTs are then able to train models for every lightpath or segment within the domain. Those DT models can afterwards be used, e.g., for accurately tracking the differences between the expected lightpath performance and the measured once, so degradations can be detected at very early stages.

However, when a lightpath that spans through more than one domain is established, such models cannot be used directly since each domain has not fully visibility of the e2e lightpath. In such cases, e2e models need to be created, so the end domains can track differences between the expected and measured performances. To that end, controllers of domains supporting a multi-domain lightpath need to exchange domain models, so end domains can compose the needed e2e models. DLT and SCs provide the needed security so such models can be securely exchanged. Note that DT models might include details of the underlying domain network topology and the configuration of optical devices, and that information, in case of eavesdropping, can be used afterwards to craft specific attacks. Therefore, a DLT network with several DLT nodes is used for the domain controllers to exchange DT models using a specific SC created for the lightpath.

The proposed workflow is sketched in Fig. 2, which is carried out as part of the multi-domain lightpath provisioning procedure. We assume that the network orchestrator and the SDN controllers have been previously configured with the needed credentials in the DLT network, and that the credentials of the SDN controllers are known by the network orchestrator. The workflow consists of two phases: *i) SC deployment*; and *ii) model exchange*. When the workflow starts, the network orchestrator compiles the SC that it will be used to store the DT models for that lightpath (0 in Fig. 2) and it deploys the SC on the DLT network (1). Next, the network orchestrator requests the DLT to associate the credentials of the SDN controllers to interact with the SC (2), so it can control the access to the SC and revoke permissions if needed (e.g., if the lightpath is rerouted through some different domains). Once the addresses are associated, the network orchestrator distributes the details to access the SC among the domain SDN controllers involved in the lightpath (3). At this point, the SDN controllers can

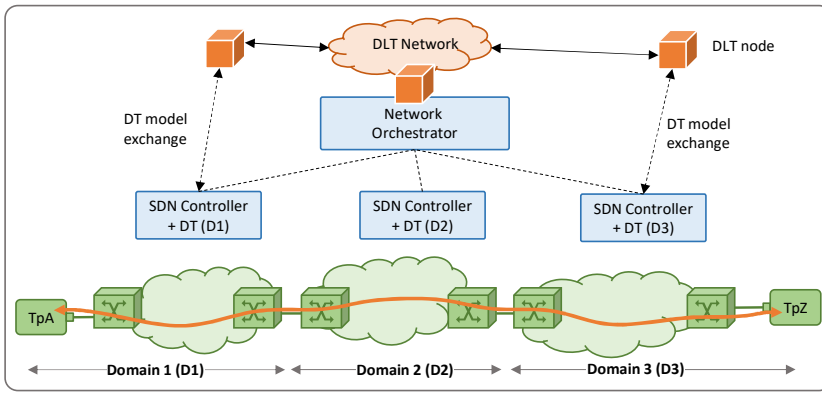


Fig. 1 Reference Network Scenario

communicate directly with a DLT node in the DLT network to perform transactions with the SC to exchange models, as well as to receive notifications when the SC has changed, i.e., a new DT model has been uploaded. When a DT model is available, the DT sends the models serialized and compressed to the related domain SDN controller (4), which uploads it to the SC for the lightpath in the DLT network (5). That transaction with the SC automatically triggers an event to the other domain controllers (6), which download the new DT model (7) and sends it to the DT, which in turn deserializes the model and uses it to compose the e2e model for the lightpath (8).

III. REDUCING THE SIZE OF DT MODELS

DLTs are typically designed to handle a large amount of transactions each of a limited size. The main reason is that DLT security is directly related to having a large number of DLT nodes that validate new transactions accurately and establish a consensus across the DLT network to ensure data consistency. However, in applications where the size of a transaction record is large, the amount of data that needs to be exchange among the DLT nodes to validate a transaction grows with the number of nodes in the DLT network, as well as it adds complexity to the DLT itself and largely limits its speed [4].

Assuming DT models based on DNNs, they consist of two parts: *i*) a dictionary that specifies the configuration of the DNN, like number of neurons, activation function, and other parameters for every layer in the model; and *ii*) a matrix of the coefficients (weights and bias) of the connections between the layers. These two parts are formatted, serialized and compressed before the model is uploaded to the DLT. In this paper, we review three ways to perform such serialization and exchange DT models through the DLT network: *i*) *minimal* formatting, where the original dictionary is augmented with the coefficients and then serialized; *ii*) *reduced* configuration, where the dictionary with the DNN configuration is reduced by removing unnecessary data, mostly used for re-training purposes. The resulting dictionary is then augmented with the coefficients and serialized; and *iii*) using *Protocol Buffers*, where the original dictionary is reduced (as in *ii*) and data is serialized using Protocol Buffers with a pre-defined schema. In all the cases, the serialized model is compressed before the exchange.

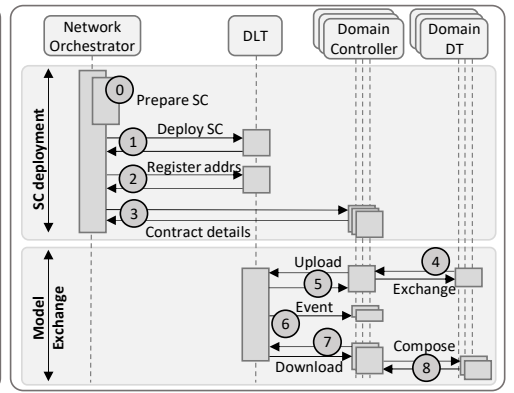


Fig. 2 Proposed workflow

IV. EXPERIMENTAL RESULTS AND CONCLUSIONS

The setup used to evaluate the proposed solution includes a DLT based on the Ethereum blockchain implementation written in Go, also known as Geth [5], which consists of four container-based nodes and a *bootnode* to provide an easy way to scale up the DLT network. An SC written in the Solidity programming language has been implemented to manage the DT models. The network orchestrator compiles and deploys the SC in the DLT network to allow the SDN controllers to interact with it directly and exchange DT models.

Let us first start with the assessment of the workflow proposed in Section II, which is independent of the selected model format. The communication with the DLT network is based on WebSocket on top of TCP, whereas the SDN controller and the DT implement REST-APIs. Fig. 3 presents a capture with the exchanged messages, where the used numbering is consistent with that in Fig. 2. During the initial phase, the SC is compiled and deployed. Messages 1-2 are exchanged between the network orchestrator (labeled *Orch* in Fig. 3) and a DLT node (labeled *DLT*) in the network to perform such deployment and for registering the addresses of the SDN controllers to enable model exchange among them. Once the details of the SC are obtained, the network orchestrator sends them to the SDN controllers using their REST-API (message 3). The credentials are encapsulated in a JSON object, which includes the address and the identifier of the SC (see Fig. 4). With such details, the SDN controllers connect to the local DLT node to be able to receive events on the SC.

Next, the domain DT prepares the model to be exchanged, including serializing and compressing the model, and it encapsulates the result in a JSON object that is sent to the SDN controller through a REST-API (message 4); see details of the enclosed JSON object in Fig. 4. When the model is received in the SDN controller, a transaction with the local DLT node is performed to upload the model to the related SC (message 5). To this end, the SDN controller uses its address in the DLT network and signs the transaction with its private key. When the transaction is validated in the DLT network, the rest of SDN controllers receive an event (6) and download the new model from the DLT network (7). Finally, the SDN controller, sends the received DT model to the domain DT, which uses to create the e2e models for the lightpath (8).

	Source	Destination	Info	
①	Orch.	DLT	WebSocket Binary [FIN] [MASKED]	SC deployment
	DLT	Orch.	WebSocket Text [FIN]	
	Orch.	DLT	WebSocket Binary [FIN] [MASKED]	
②	Orch.	DLT	WebSocket Binary [FIN] [MASKED]	SC deployment
	DLT	Orch.	WebSocket Text [FIN]	
	Orch.	DLT	WebSocket Binary [FIN] [MASKED]	
③	Orch.	SDN-1	POST /contract HTTP/1.1 application/json	Model Exchange
	SDN-1	Orch.	HTTP/1.1 200 OK application/json	
④	DT-1	SDN-1	POST /model HTTP/1.1 application/json	Model Exchange
	SDN-1	DT-1	HTTP/1.1 200 OK application/json	
⑤	SDN-1	DLT	WebSocket Binary [FIN] [MASKED]	Model Exchange
	DLT	SDN-1	WebSocket Text [FIN]	
⑥	DLT	SDN-2	WebSocket Text [FIN]	Model Exchange
	SDN-2	DLT	WebSocket Binary [FIN] [MASKED]	
⑦	DLT	SDN-2	WebSocket Continuation [FIN]	Model Exchange
	SDN-2	DT-2	POST /model HTTP/1.1 application/json	
⑧	DT-2	SDN-2	HTTP/1.1 200 OK application/json	Model Exchange

Fig. 3 Exchanged messages

③	{ "SC_Details": { "contractAddr": "0x18F4546...8de81D7", "contractId": "LSP_2458_BCN-MAD.sol" } }
④	{ "LSP_Id": "LSP_2458_BCN-MAD", "Domain_Id": "D2", "model": "\xbbb\xee\xfb\xfd7\xbc\xca..." }

Fig. 4 Details of message 3 and 4

Let us now analyze the efficiency of the different formats for model exchange proposed in Section III. For the minimal and reduced configuration formats, the efficient binary serialization format Message Pack (*msgpack*) has been used to serialize/deserialize the models, whereas for the one using Protocol Buffers, a schema has been defined fitting the format of the reduced configuration dictionary and the coefficients. To compress the resulting serialized models, the *lzma* compression algorithm is used.

Two DNN-based DT models from [1] have been used to compare the formats under evaluation. The first model (*small*) is a DNN to predict the distance of a lightpath and consists of 20 inputs, 2 hidden layers with 12 and 6 neurons, respectively, and a total of 337 coefficients. The second model (*large*) is a DNN modeling the lightpath segment in the domain, and it consists of 20 fully-connected layers with a total of 320 neurons and 5,020 coefficients (weights and bias).

Table 1 presents the size of each model before and after being serialized and compressed for the considered formats, as well as the obtained times for model exchange; rows are numbered for the sake of clarity. For both models, we observe that the size of the configuration dictionary is reduced noticeably (rows 1 in Table 1). However, when the coefficients are included (rows 3), the reduction in size between the minimal and the reduced configuration formats decreases to 31% in the case of the small model and to only 18% for the large one. The reason is that the coefficients matrixes remain unchanged (see rows 2). Comparing the minimal and the reduced formats, we observe that such reduction decreases to 24% and to 14% after serialization (rows 4) and just to 12% and 5% after compression (rows 4). However, both the reduction in size after serialization with Protocol Buffers and compression exhibits larger figures, achieving reductions of 33% for the small model and 21% for the large one, after compression. Let us now analyze the time to exchange a model, measured between the response to

Table 1. Sizes (Kbytes) and exchange times (ms) of the formats

		Minimal	Reduced	Proto Buff
Small model	1 Config. Dict.	2.18	0.21	0.21
	2 Coefficients	4.19	4.19	4.19
	3 Config + Coeff.	6.37	4.40	4.40
	4 Serialized	5.74	4.34	1.43
	5 Compressed	2.04	1.79	1.36
	6 Exchange times	219.6	216.9	212.9
Large model	1 Config. Dict.	12.30	1.07	1.07
	2 Coefficients	50.11	50.11	50.11
	3 Config + Coeff.	62.41	51.19	51.19
	4 Serialized	58.80	50.82	20.45
	5 Compressed	6.62	6.30	5.23
	6 Exchange times	517.9	491.4	488.3

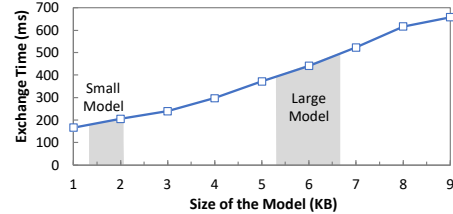


Fig. 5 Exchange Time vs Size of the model

message 5 is received at the SDN controller uploading the model and the event 6 that is received when the transaction has been validated (see workflow in Fig. 2). We observe large differences between the time to exchange the small and the large models (rows 6). Such differences are as a result of the size of the models, which supports our target to reduce their size as much as possible. To better show that relation, Fig. 5 plots the time to exchange a model as a function of its size for size ranging from 1 to 9 Kbytes. We observe a close-to linear relation in the studied size range. Note that, depending on the configuration of the DLT, models of higher size would require more than one transaction, which would introduce a high increment in the exchange times.

To conclude, a secure DT model exchange mechanism using DLT and SC has been experimentally demonstrated on a setup with three network domains. The size of the DT models to be distributed impacts on the exchange times through the DLT, so three different formats with specific serialization and data compression performance have been proposed aiming at reducing models' size. Evaluation was carried out using two realistic DNN models.

ACKNOWLEDGEMENT

The research leading to these results has received funding from the European Union's Horizon Europe research and innovation programme under G.A. No. 101092766 (ALLEGRO), the MICINN IBON (PID2020-114135RB-I00) project and from ICREA.

- [1] L. Velasco *et al.*, "Applications of Digital Twin for Autonomous Zero-Touch Optical Networking," ONDM, 2023.
- [2] D. Sequeira *et al.*, "OCATA: A Deep Learning-based Digital Twin for the Optical Time Domain," JOCN, 2023.
- [3] K. Antevsk and C. Bernardos, "Federation in Dynamic Environments: Can Blockchain be the Solution?" Comm. Magazine, 2022.
- [4] N. Kannengießer *et al.*, "Trade-offs between Distributed Ledger Technology Characteristics," ACM Computing Surveys, 2020.
- [5] Go-Ethereum. [On-line] <https://geth.ethereum.org/>