Consistent VNF Forwarding Graphs Provisioning in Multi-domain Environments

Josué Castañeda Cisneros, Sami Yangui, Saul Pomares Hernandez, Julio Cesar Perez Sansalvador, Khalil Drira

Consistent VNF Forwarding Graphs Provisioning in Multi-domain Environments

Josué Castañeda Cisneros*, Sami Yangui*§, Saul E. Pomares Hernández*†, Julio César Pérez Sansalvador†‡, Khalil Drira*
*LAAS-CNRS, Université de Toulouse,
§INSA, F31400, Toulouse, France.
† INAOE, 72840, Santa María Tonantzintla,Puebla, Mexico.
‡ INAOE - Cátedra CONACyT, 72840, Santa María Tonantzintla,Puebla, Mexico.

Abstract

Network Function Virtualization (NFV) decouples the network functions from the underlying infrastructure and enables running Virtualized Network Functions (VNF) on top of any generic, Commercial On-The-Shelf (COTS) hardware. VNFs rely on the VNF-Forwarding Graph (VNF-FG) concept to describe and implement network topologies (e.g. connectivity schemes, forwarding rules and dependencies). VNF-FGs are provisioned and managed by appropriate orchestrators such as NFV-MANO. In multi-domain approach, and for complex and sophisticated network topologies, VNF-FGs are managed by multi-domain orchestrators. This improves on the drawbacks of traditional centralized orchestration such as limited availability, communication bottlenecks and lack of flexibility for services. However, distributed orchestration might lead to inconsistencies, and thus partial or total failure of services. Enabling such system is quite challenging due to inherent restrictions of multi-domain environments such as asynchronous communication and limited knowledge. This paper proposes the use of causal dependencies among VNFs to provision consistent VNF-FG reconfiguration in distributed multi-domain environments under asynchronous communication channels. The proposed model is implemented and compared with the traditional consistency used in centralized and distributed orchestration. Results indicate that, with this proposal, the inconsistencies due to asynchronous communication channels are prevented.

I. Introduction

Network Function Virtualization (NFV) is an European Telecommunications Standards Institute (ETSI) initiative[1] that decouples the Network Services (NS) from the underlying hardware. Physical network appliances are replaced with software-based Virtual Network Functions (VNFs) [1]. The ultimate goal is bring agility, dynamicity and cost-effectiveness when operating networks and network functions. To support a variety of users and requests for the NS, multiple reconfigurable VNF-Forwarding Graphs (VNF-FG) are created [2]. The VNF-FG specifies a topology of connectivity of multiple VNFs that compose the services and optionally forwarding rules applicable to the traffic conveyed over the topology [3]. It takes often tree topologies built on basic components such as line, bifurcation with single and different endpoints [4], [5]. This topology is a graph that defines the execution order among VNFs that compose a service based on their dependencies [6]. Implementation wise, VNFs connect with others using virtual links (VL) through interfaces called connection points (CP) [7]. It is important to note that a single network service can have multiple VNF-FGs and each specifies the execution and dependency order [3].

According to ETSI, the orchestrator is responsible for the end-to-end provisioning of network services, the composition of their VNF-FGs and theirs life-cycle management. Traditionally, the NS are created by a single orchestrator that has global knowledge of the domain it manages such as: technology, location of VNFs, domain policies [8]. This centralized solution selects VNFs based on functional requirements and places them in optimal locations to met non-functional requirements of the service. However, this single domain orchestration design poses some problems: (i) resource consumption is expected to reach 4.8 Zetabytes by 2022 [9], (ii) the advent of next generation network technologies such as 5G introduces tighter constraints for services such as a latency of 5ms, throughput of 10GB/s and service deployment of 90 minutes [10], (iii) the availability of services is compromised. To handle these problems, multi-domain orchestration has been proposed where many domains, that offer specific functionalities, can compose more sophisticated services [11].

Through coordination of multiple orchestrators, services are executed by using shared VNFs. The execution order is based on priorities and dependencies with respect to the VNF-FGs [12]. These VNF-FGs can be reconfigured to optimize profit by selecting a suitable strategy for resource reservation in different domains. This reconfiguration is a task of the VNF Forwarding Graph provisioning problem.

Despite the work done on the literature, reconfiguration has not been addressed thoroughly. Aside the offline provisioning where no changes in the VNF-FG are taken into account [2], [13], [14], few work have addressed the reconfiguration of the VNF-FG. In [15] the online placement of VNF-FG for content delivery networks is presented. The authors propose an Integer

[1]https://www.etsi.org/technologies/nfv

Linear Programming model where VNFs are reused to place the VNF-FG. Similarly, in [16] the authors propose a model that takes into account VNF migration where VNF-FG can be reconfigured to obtain optimal performance. They consider the case of cooperative multi-domain orchestration. These works consider the placement of the VNF-FGs that can change, their objective is to minimize cost and maximize performance. Nevertheless, they do not consider the inconsistencies that can be created during reconfiguration through the coordination of multiple orchestrators. Some inconsistencies can be created due to asynchronous channels in the network: messages sent by the orchestrators can arrive in a different order that they were sent. This leads to inconsistencies in the dependency relations of the VNF-FG. Traditionally, only eventual consistency model is considered for the reconfiguration, however there are cases were stronger guarantees are necessary.

In this work, we study the consistency update problem for VNF-FG in federated multi-domain environments. Our research questions are the following (i): What kind of inconsistencies can occur during the VNF-FG composition for shared network services in a distributed multi-domain federation? (ii) How can consistent VNF-FGs be achieved in multiple domains while facing asynchronous communication and partial information among the multiple orchestrators that compose a distributed multi-domain federation environment? Our main contribution is the identification of inconsistencies created during the reconfiguration of VNF-FGs through the coordination of orchestrators in a distributed environment. We propose the use of causal orderings to prevent inconsistencies. After an evaluation we discuss the trade-offs between consistency guarantees, performance, extendibility and limitations of the proposed consistency model in the domain of NFV.

The rest of the paper is structured as follows. Section II describes a use case to illustrate the inconsistencies due to asynchronous channels and limited information of the orchestrators. Section III introduces the definitions and system model. Section V discusses the proposed solution. Section VI presents the implementation details, as well as, the performed experiments and the obtained results. Section VII reviews the related work. Section VIII concludes the paper.

II. Motivating Use Case: Inconsistency in Multi-Domain Environments

In this section we describe the problem of inconsistent updates of VNF-FGs under a distributed multi-domain. We consider a set of users that consume services which has data that originates from many sources and flows trough multiple VNFs as defined in the VNF-FGs [17]. We consider the case of a close federation where a fixed number of trustful orchestrators share their VNFs to support services. We let the case of dynamic and distrustful orchestrators for future work. Next, we describe the problem of consistent VNF-FG provisioning with a use case.

Consider the case of a smart city where a monitoring system is deployed to prevent disasters such as [18], [19]. In case of an emergency, a small council of people must make decisions to coordinate the city's resources such as police, firemen and health services to mitigate the damage and reduce impact to a community [20]. The interest is to locate possible hazards such as fires, wounded persons or terrorists. To make informed decisions, multiple sources of information are considered despite the complete or partial shutdown of the monitoring infrastructure. To this end, data sent by multiple devices is aggregated and key information is extracted. This data is obtained by crowd-sensing citizens when they start live video/audio streams using multiple devices such as their smartphones, microphones or digital cameras [21], [22]. Locating hazards is the first objective of the mission. The second one is to gather contextual information based on multiple sources in the area. One critical requirement of the mission is synchronization between the key videos and text; thus services provided in the mission have latency and delay constraints. To achieve the goals Network Function Virtualization paradigm is considered.

Due to the partition of the network during disasters, the supporting services that extract key information from live-data are distributed among multiple domains domains that create a federation. The federation covers an entire area of interest as shown in Figure 1. In it, each domain provides different network functions based on their capacities. Network services can be instantiated using VNFs from multiple domains. If a VNF is used by multiple services, it is said that the VNF is $shared$ such as the *Decoder* shown in Figure 1.

The main goal of the orchestrators in the federation is to coordinate themselves to achieve the required network services while satisfying multiple functional and non-functional requirements of their users [23], [24]. To support these requirements, a forwarding path from the source until the target is done as shown in the top part of Figure 1. However, due to the limited knowledge of each domain; is necessary to obtain information of VNFs that enable services such as topology, address space and connection points [25]. This can be achieved by message passing among the orchestrators that manage resources and services that execute on top of them.

The exchange of information done by messages must include the negotiation of resource allocation, flow steering, policies and updates for the VNF and VNF-FG [23]. After the negotiation phase, VNFs are instantiated and the VNF-FG is created as an ordered set of VNFs that the network service executes to fulfil the services' attributes [26] as shown in Figure 1. The orchestrators are responsible for creating and sharing the VNF-FG information among them by message passing.

However, due to asynchronous channels in the network, messages sent can be received in a different order that they were sent. Consider the exchange of messages between orchestrators in Figure 1 to signal an update of VNFs in the VNF-FG of the service that delivers information to foreign tourists shown in the green VNF-FG of (A) Figure 2. Due to internal operations, two orchestrators in the federation need to change connections points of the *Encoder* and *Decoder* VNFs. After the update,

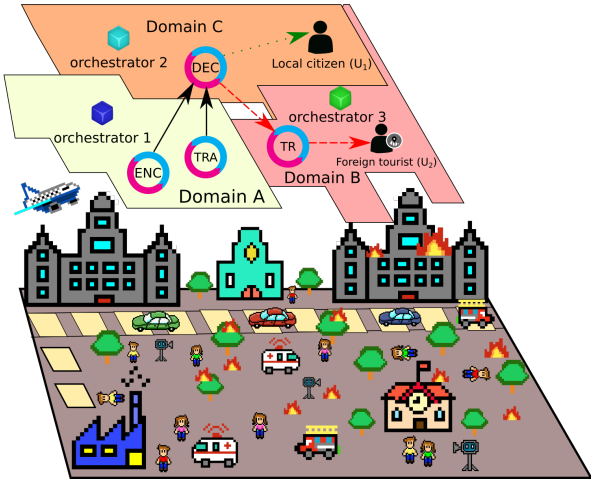

Fig. 1. Federation covers the city with multiple domains with different orchestrators. A VNF-FG is setup using different VNFs from multiple domains and brings functionality to multiple users such as local and foreign tourists to identify and prevent disasters.

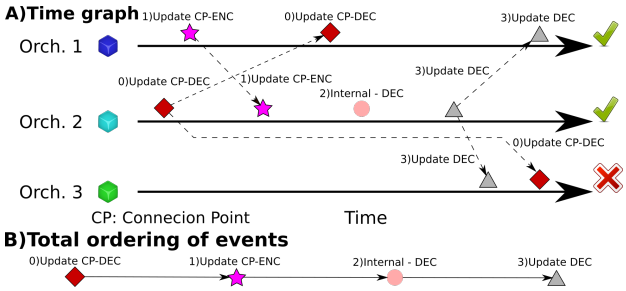

Fig. 2. Inconsistency VNF-FG update due to asynchrony. Two orchestrators update connection points of the VNFs (*Encoder*, *Decoder*) and send messages to the respective affected orchestrators. Due to asynchrony the first update of connection point, denoted with 0, arrives to the orchestrator after the last update with numbering 3. At the end, this orchestrator will update the connection based on old values, creating an inconsistency in the network service. The inconsistency yields a service partial failure.

they must inform the affected orchestrators as shown in the VNF-FG; in the case of *Encoder* both the 1st and 3rd orchestrator are concerned while for the VNF *Encoder* only the 2nd orchestrator.

Due to asynchrony in the network messages arrive to orchestrators in a different order they were sent: First the update of *Decoder* connection points is done at 2nd orchestrator and the message is sent to the other two orchestrators. Sometime after, the 1st orchestrator updates the connections points for the *Encoder* VNF and sends it to the 2nd one. Then, the 2nd orchestrator receives the message of update from the 1st orchestrator which triggers an internal event and setups another configuration of connection points for the *Decoder* that is also sent to two orchestrators. That messages arrives to the 3rd orchestrator who updates the connection point. Finally, the first message arrives out of order, and the 3rd orchestrator updates again the connection point. This leaves an inconsistent state, the VNF-FG does not work anymore and it does not satisfy the ordering as shown in Figure 2 (B). The service works partially, only delivering information to the local citizen.

The spurious update can result in the violation of dependency order for the VNFs, which leads to an inconsistent VNF-FG. This leaves the network service in two possible states:

1) Partial Failure: This is the case when some VNF-FGs in the service still satisfy the order dependency of VNFs. This is the case for the use case, where the service delivered to *local citizens* is still running but the one delivered to *foreign tourists* is interrupted as shown in Figure 1.
2) Complete Failure: The inconsistency affects all VNF-FGs in the network service which brings it to an entire halt.

III. Definitions and System Model

In this section we introduce the required definitions and the System Model used in work. We summarize the notation in Table I.

A. System Model

The federation is composed of multiple domains denoted with the set $D = \{d_1, d_2, \cdots, d_n\}$. For each domain d_n there is an orchestrator o_n that belongs to the set $O = \{o_1, o_2, \cdots, o_n\}$. We consider that orchestrators can manage services that belong to the set $S = \{s_1, s_2, \cdots, s_m\}$. A service s_m is composed of a subset of VNFs that belong to the set $V = \{v_1, v_2, \cdots, v_p\}$ and virtual links belonging to the set $L = \{l_1, l_2, \cdots, l_q\}$. Each link l_q is composed by a pair of VNFs $v_p, v_p^{'} \in V$ such that $(v_p, v_p^{'}) \in L$. To represent the dependencies and order of execution of VNFs $V_m \subseteq V$ and virtual links $L_m \subseteq L$ of service s_m, a subset of VNF-Forwarding Graphs that belong to set $G_m \subseteq G = \{g_1, g_2, \cdots, g_r\}$ is considered. A VNF-FG is composed by an ordered sequence of VNFs $\alpha_0, \alpha_1, \cdots, \alpha_k, \alpha_{k+1}, \cdots, \alpha_u$ such that any α_k belongs to the set V. Moreover, each service s_m can have multiple VNF-FGs such that $|G_m| \geq 1$. We only consider the case of stateful VNFs which store information associated with sessions being served. To abstract the state resources for a VNF v_p, the operation $state(v_p)$ is defined. Connection points can be part of the *state* of a given VNF.

For a service s_m with at least one VNF-FG G_m, a reconfiguration operation on G_m is as follows: Let v_p be a VNF in G_m, the reconfiguration operation $\Delta : v_p \rightarrow v_p^{'}$ such that $state(v_p) \neq state(v_p^{'})$. The change of state be done by scaling, migration, update.

We define a binary relation *isManaged* between an orchestrator o_n and a service s_m if there is at least one VNF v_p that is part of a domain d_n:

$$isManaged(o_n, s_m) = \begin{cases} 1, & \text{If } \exists v_p \in s_m, s_m \in d_n \\ 0, & \text{Otherwise.} \end{cases} \tag{1}$$

The binary relation *depends* is defined between two VNFs $v_p, v_p^{'}$ that are part of a service s_m if during execution of the service, the first VNF v_p must be executed before the second VNF $v_p^{'}$ for any VNF-FG g_r that belongs to the service:

$$depends(v_p, v_p^{'}) = \begin{cases} 1, & \text{If } v_p, v_p^{'} \in V_m | V_m \in G_m \text{ and} \\ & \alpha_k \equiv v_p, \alpha_{k'} \equiv v_p^{'} | k' > k \\ 0, & \text{Otherwise.} \end{cases} \tag{2}$$

Two binary relations for services $multiDomain$ and $singleDomain$ are as follows:

$$multiDomain(s_m) = \begin{cases} 1, & \text{If } \exists o, o' \in O | \\ & isManaged(o, s_m), \\ & isManaged(o', s_m) \\ 0, & \text{Otherwise.} \end{cases} \tag{3}$$

$$singleDomain(s_m) = \begin{cases} 1, & \text{If } !multiDomain(s_m) \\ & \text{and } \exists o \in O \\ & | isManaged(o, s_m) \\ 0, & \text{Otherwise.} \end{cases} \tag{4}$$

The subset of VNFs V_p that can be impacted when a VNF v_p updates its properties denoted as $affected(v_p)$ is as follows:

$$affected(v_p) = V_p | \forall v \in V_p depends(v_p, v) \tag{5}$$

TABLE I
SYSTEM MODEL NOTATION

Variable	Notation
Domains	$D = \{d_1, d_2, \cdots, d_n\}$
Orchestrators	$O = \{o_1, o_2, \cdots, o_n\}$
Services	$S = \{s_1, s_2, \cdots, s_m\}$
VNFs	$V = \{v_1, v_2, \cdots, v_p\}$
Virtual Links	$L = \{l_1, l_2, \cdots, l_q\}$
VNF-FGs	$G = \{g_1, g_2, \cdots, g_r\}$
VNF-FGs of service s_m	$G_m = \{V_m, L_m\}$
Ordered chain in VNF-FG	$\alpha_0, \alpha_1, \cdots, \alpha_k, \cdots, \alpha_u | \forall \alpha_k \in V$
Messages	$M = \{m_1, m_2, \cdots, m_v\}$
Events	$E = \{e_1, e_2, \cdots, e_w\}$

B. Causal relations

We model the federation as a distributed system where different spatially separated entities communicate with each other by exchanging messages. For the particular case in NFV, only the orchestrators communicate among them since they have end-to-end knowledge of services. Each orchestrator $o \in O$ has knowledge of the domain it manages $d \in D$. The elements in the distributed system are as follows:

- Process. Programs or instances running simultaneously that communicate with other programs. For the NFV context, the processes are orchestrators that belong to the set of orchestrators O. They communicate with others processes by message passing over a network.
- Messages. Abstractions that represent packets in a communication network. They can contain complex data structures. Each message in the system belongs to the set $M = \{m_1, m_2, \cdots, m_v\}$
- Event. An action performed by a process. There are two types considered: *internal events* and *external events*. An internal event is an action that happens at the process in a local manner such has the selection, instantiation or reconfiguration of a VNF in the orchestrator's domain. An external event is also an action that is perceived by other processes and affects the global system state. Each event e belongs to the set $E = \{e_1, e_2, \cdots, e_w\}$. There are four types of events: *internal*, *send*, *receive*, and *delivery*.
  1) The internal event refers to an internal operation for VNFs such as update of connection points, scaling, migration.
  2) The send event denotes the emission event executed by a process.
  3) The receive event refers to the arrival's notification of a message in a process.
  4) The delivery event denotes the recipient's execution to communicate the received information to another process.

A causal order establishes a precedence relation between two events in the following way: let e and $e^{'}$ be two events causally related, it is said that e *happened before* $e^{'}$ if there is a transference of information from e to $e^{'}$. Given such relation, e must be processed before $e^{'}$. The relation $\rightarrow$ is the smallest relation on a set of events E satisfying:

1) If e and $e^{'}$ are events belonging to the same process and e originated before $e^{'}$. then $e \rightarrow e^{'}$.
2) If e is the sending of a message by one process and $e^{'}$ is the receipt of the same message by another process, then $e \rightarrow e^{'}$.
3) If $e \rightarrow e^{'}$ and $e^{'} \rightarrow e^{''}$, then $e \rightarrow e^{''}$

Two events, e and $e^{'}$, are concurrent if there is not causal dependency between e and $e^{'}$ denoted as $e \not\rightarrow e^{'}$ and there is not causal dependency between $e^{'}$ and e and denoted as $e^{'} \not\rightarrow e$. We denote the concurrency of two VNFs e and $e^{'}$ as $e||e^{'}$

IV. Problem Definition

After a reconfiguration operation Δ for VNF α_k has taken place and there is a previous VNF α_{k-1} or next VNF α_{k+1}, a message is sent to other orchestrators. Formally, we denote this as $send(v, state(\alpha_k))$ if $affected(\alpha_k) \neq \emptyset$ and $\Delta(\alpha_k)$.

An inconsistency due to asynchronous channels between a pair of messages $m, m^{'} \in M$ for two orchestrators $o, o' \in O$ is created when event $send(m)$ is generated before $send(m')$ by orchestrator o denoted as $send(m) \prec send(m^{'})$ but the delivery to orchestrator o' is out of order denoted as $delivery(m') \prec delivery(m)$. Is important to note that reception can be out of order $receive(m') \prec receive(m)$ without creating an inconsistency.

The consistent VNF-FG graph reconfiguration problem consists in the following: For any VNF-FG G_m with a VNF α_k, a consistent reconfiguration through an exchange of states messages $m = state(\alpha_k)$, $m' = state(\alpha_k)$ is achieved if for all pair of events $(send(m), send(m'))$, if $send(m)$ happened before $send(m')$ then the delivery of m is done before the delivery of m'. Formally this is defined with the binary relation over the reconfiguration operation *isConsistent($\Delta(v)$)* for any VNF $v \in V$:

$$isConsistent(\Delta(v)) = \begin{cases} 1, & \text{If } m \equiv \Delta(v) = state(v) \\ & m' \equiv \Delta(v) = state(v) \\ & \forall (m, m') \in M \\ & \text{If } send(m) \prec send(m') \rightarrow \\ & delivery(m) \prec delivery(m') \\ 0, & \text{Otherwise.} \end{cases} \tag{6}$$

To achieve consistent VNF Forwarding graph provisioning, Equation 6 must hold true for all events that involve the reconfiguration of VNFs.

V. Proposed solution

In this section, first we define a series of relations that describe the properties between orchestrators, services, VNFs and VNF-FGs and introduce the causal dependency relation for the context of NFV. Then, we describe our proposed algorithm via pseudo code and illustrate the use of it via an execution of the use case presented in Section II.

Algorithm 1 Event management algorithm (Event e, Orchestrator o)

logicalClockVNF $\leftarrow$ e.getLogicalClock()
informationVNF $\leftarrow$ e.getInformationVNF()
currentVNF $\leftarrow$ e.getVNF()
if e is $internal$ **then**
  o.updateVNF(*informationVNF*)
  o.increaseLogicalClock(*informationVNF*)
  if $exists\ v \in V | depends(currentVNF, v)$ **then**
    currentClock $\leftarrow$ o.logicalClock(*informationVNF*)
    event $\leftarrow$ $send(currentClock, informationVNF)$
    o.sendToAll(event)
  end if
end if
if e is $received$ **then**
  o.increaseLogicalClock(*informationVNF*)
  currentClock $\leftarrow$ o.logicalClock(*informationVNF*)
  if currentClock $>$ logicalClockVNF **then**
    o.updateVNF(*informationVNF*)
    o.setLogicalVector(logicalClockVNF)
  end if
end if

The VNF-FG is enriched with the causal order relation in the following way: If two VNFs $v_p \in d_n, v_p^{'} \in d_{n'}$ that belong to a VNF-FG g_r have a *depends()* relation and $d_n \neq d_{n'}$, then there is a causal order among the the subet set of events $E' \subseteq E$ that pertain the two VNFs. Formally:

$$e \rightarrow e^{'} = \begin{cases} 1, & \text{If } \exists v_p, v_p^{'} \in V_m | depends(v_p, v_p^{'}), \\ & d_n \neq d_{n'} \text{ and} \\ & v_p, v_p^{'} \in e | send(e), receive(e), delivery(e) \\ 0, & \text{Otherwise.} \end{cases} \tag{7}$$

The pseudo code of our algorithm is shown in Algorithm 1. Next, we describe the algorithm using the notation previously presented. We suppose orchestrators in the federation f has a subset of services $S_f \subseteq S$. The services of S_f are supported by a subset of VNF-FGs $G_f \subseteq G$. Each VNF v in a VNF-FG $g \in G_f$ *isManaged* by an orchestrator $o \in O_f$ as defined in Equation 1. Services can be of *multiDomain* or *singleDomain*. Each orchestrator maintains a logical clock of each VNF that *isManaged*. These VNFs can be updated by the orchestrator at any time with an *internal* event. After an *internal* event of a VNF v_p, if there is VNF v_p' and $depends(v_p, v_p')$ is true, then a *send* event is sent to all orchestrators that have a VNF v_p' that is in the subset *affected(v_p)* orchestrators. If a *receive* event is obtained by orchestrator o, it will update its logical clock for the VNF concerned. After the increment, it will compare the internal logical clock and the received one. If the received one is greater or equal, it will update its internal clock to the received value and update the VNFs information.

We illustrate our algorithm applied to the situation presented in Figure 2 from Section II. For the sake of simplicity, each orchestrator has a logical clock for all the VNFs in the system. Figures 3 and, 4 shows an example of execution divided in 9 setps. In step I, the *Decoder* VNF is updated and a *send* event is generated and the message is sent to both the 1st and 3rd orchestrator. In step II, the *Encoder* VNF is updated and a *send* event is generated to the 2nd orchestrator. In step III, a *delivery* event is obtained by the 2nd orchestrator, which updates the logical clock to 1 of its *Encoder* to reflect changes in the connection points. In step IV, a *delivery* event is obtained by the 1st orchestrator, which updates the logical clock of its *Decoder*. In step V, an *internal* event is generated by the 2nd orchestrator that is not sent to any orchestrator. In step VI, a *send* event is generated by the 2nd orchestrator and it is sent to the 1st and 3rd orchestrator. In step VII, the event generated at step VI is obtain and a *delivery* event is generated by the 3rd orchestrator and it updates the logical clock of its *Decoder* setting it to 3. In step VIII the message generated at step VI generates a *delivery* by the 1st orchestrator and it updates the logical clock of its *Decoder* to 3. Finally, in step IX the message generated at step I is *received* by the 3rd orchestrator which compares the value to its previous logical clock for *Decoder*. The received clock value is 1 and the current clock value is 3, since the received value is less than the current clock, it does not update it's logical clock and neither the associated VNF information. At the end, all orchestrators have the same view for dependent VNFs satisfying the causal dependency relation. It is important to note that some changes are not visible to all orchestrators as shown in the changes of *Encoder* which is only shared by the 1st and 2nd orchestrator. This is the case for concurrent events in the federation.

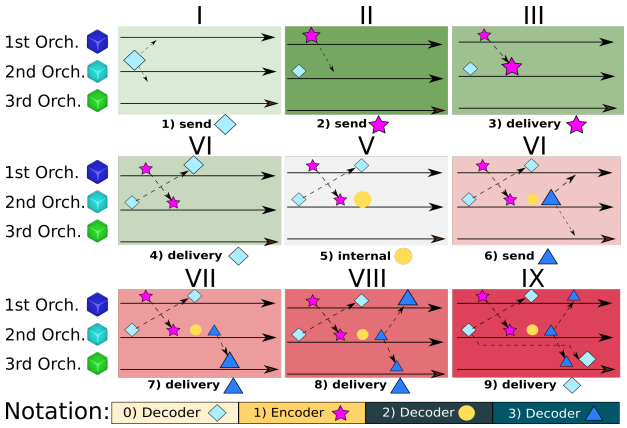

Fig. 3. Sequential execution of updates for the VNF-FG for each VNF. Each time an update triggers, the orchestrator will $send$ messages to all $affected$ orchestrators. Due to asynchrony, messages can arrive in a different order they were sent as shown by the delivery of the 1st updates in steps VI and IX.

VI. Implementation and Evaluation

To test our proposed algorithm, we use the multi-pop OSM test bed [27]. We set multiple domains each with an orchestrator that has information about only that particular domain. Each domain has it's policies, resources and VNFs. Services are requested by a client and the petition is sent to an orchestrator. The orchestrator sends request to all others in the federation to set up the VNF-FG for the shared network service based on the functionality and dependencies of the VNFs. We consider services of multiple size and across multiple domains. However, we do not consider services that could lead to cycles such as two VNFs being dependant on each other directly or transitively. The description of experimental values is shown in Table II. We evaluate both the traditional approach in federation and our proposed algorithm by measuring the following metrics:

- Communication bottleneck: The time waiting for an answer from the orchestrator. We measure for both the centralized and decentralized approach.
- Average number of inconsistencies by spurious updates: Reflects the errors when updating VNF-FGs that have VNFs with order dependency among them and one old update overwrites the most recent value due to asynchrony.
- Message Overhead: Measures the cost involved by introducing the causal dependencies among VNFs. In the worst case scenario where all VNFs have dependency order among them, the number of messages is equal to $\mathcal{O}(np)$ where n is the number of orchestrators and p is the number of VNFs in the federation.

A. Communication bottleneck

We evaluated the time spent waiting for the orchestrator to return an answer after an update for both the centralized and our decentralized approach. We considered 4 waiting periods of 1, 2, 4, 8 seconds. A waiting value was generated randomly for each class. Results are shown in Figure 5.

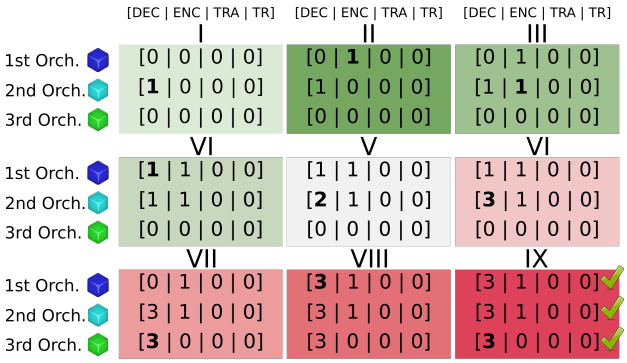

Fig. 4. Example of execution of our proposed algorithm to solve the inconsistency present in Figure 2. The orchestrators have a vector clock of size equal to each VNF that begins at zero. Each time there is an $internal$ or $send$ event, the clock of the VNF is increased by one. When there is a $delivery$ the maximum value of the clocks is taken. At the end of the update for the VNF-FG, all orchestrators have a consistent view based on the dependencies of the VNFs.

TABLE II
TABLE OF PARAMETER FOR EXPERIMENTS

Variable	Value
Number of orchestrators	4
Services per experiment	3
Number of VNFs	13
Number of VNFs per services	4, 5, 6
Random waiting period in seconds	0-20

TABLE III
NUMBER OF INCONSISTENCIES

Algorithm	Avg. Inconsistency	Std. Deviation
Traditional Size 3	37	6
Traditional Size 4	33	5
Traditional Size 5	28	3
Proposed	0	0
Proposed	0	0
Proposed	0	0

B. Average number of inconsistencies

We generated 10 random experiments with 100 updates done to the VNF-FGs for services of size 4,5,6. The previous sizes reflect the trade-off between performance and length of VNF-FG as a long service required more communication overhead and waiting time. Random waiting times were used to simulate asynchronous channels so messages sent by the orchestrators could be delivered in different times. We consider the worst case scenario where all VNFs have dependencies and orchestrators must coordinate with all others. We compared the traditional approach where dependency is given by a list and our proposed algorithm that introduces a causal dependency. We rounded up the values obtained. Results are shown in Table III.

To give a more in depth look at the delay factor for the traditional solutions, we plot delays and count the number of inconsistencies for the different size of services. Delays range from 0 to 20 seconds. Greater delays, while possible in real scenarios, are improbable with current networks and applications. Results are shown in Figure 6.

C. Message Overhead

We generated 10 random experiments with 100 updates done to the VNF-FGs. Each orchestrator had a random probability assigned to update a VNF-FG. We measured the messages sent by orchestrators to coordinate with other orchestrators. We rounded up the values obtain. Results are shown in Table IV.

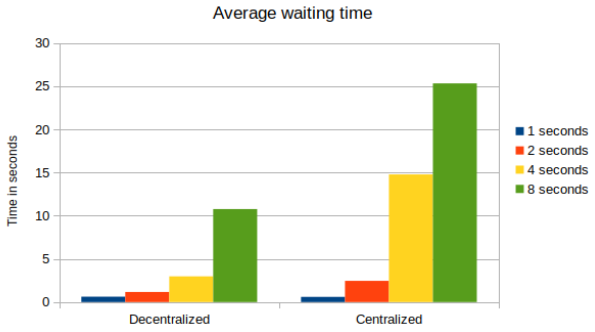

Fig. 5. Comparison between centralized and decentralized communication waiting time. The centralized approaches creates a bottleneck that can impact services

TABLE IV
NUMBER OF COORDINATION MESSAGES SENT

Algorithm	Avg. Messages	Std. Deviation
Proposed	23	11

D. Discussion and limitations

Based on the results obtained, our proposed algorithm eliminates the inconsistencies during the VNF-FG update of services in the federation created by asynchronous channels in the network. The traditional approach behaves worst as de the delay increases which is expected since messages have a higher change of arriving out of order. On the other hand, smaller VNF-FG have a higher inconsistency rate than bigger ones, this could be because in our experiment smaller VNF-FG tend to share more VNFs that are dependent in the setup which increases the probability of having an inconsistency. It is important to note however, that we only consider order dependent VNFs; this cannot be the case always as there are solutions which consider unique VNFs. The overhead communication cost was about 23% of the updates messages generated by the orchestrators. However, the number of messages is proportional to the changes done in the VNF-FG where there are dependent VNFs. This shows that despite the overhead, a decentralized solution is better than a traditional one in terms of delays as shown in the average waiting time. Based on this trade-off, we think that our proposed algorithm is a step towards provisioning consistent VNF-FG updates in distributed multi-domain environments.

There are however some limitations of our work: we require to have the services and VNFs in advance to setup the logical clocks for the federation, this could limit the applicability in open-federations where new participants are allowed to enter. When new providers can enter and setup services, security is a concern, techniques such as block-chain can be explored to include anonymity, security and consistency [28]–[30]. However, performance trade-offs must be taken into account due to the restrictions imposed by services such as latency.

We only consider inconsistency with the VNF-FG updates created by asynchronous channels in the network. More work is to be done on other kinds of inconsistencies that could arrive for multi-domain environments. For example, chain dependent reconfiguration: a single reconfiguration in an orchestrator could trigger the complete change of the service function chain. The number of inconsistencies could be greater since shared VNFs use multiple services, thus there is an interest to study this more general problem.

The impact of VNF-FG updates for services in the federation was not measured in our study. Since causal ordering introduces an overhead in services to prevent inconsistencies, a comparison in terms of performance such as delay or latency between consistency models for the context of NFV can be addressed in future work. This would bring more refine criteria to select stronger or weaker guarantees for deployment of virtual services in distributed scenarios.

Despite this limitations, the results obtained give insight for the two research questions we posed in the introduction: We identify the inconsistency in the updates of the VNF-FG due to limited knowledge of the orchestrators and asynchronous communication which was not considered previously in the literature in the context of NFV. We proposed the causal dependencies for updates in the VNF-FG to handle the identified inconsistency. However, we believe that there are other types of inconsistencies that need to be addressed for the provisioning of services with distributed multi-domain orchestration.

VII. Related Work

The literature for VNF-Forwarding Graph (VNF-FG) is mainly concerned with the initial placement of the requested service function chains [26], [31]. However, reconfiguration of the VNF-FG with operations such as: update, extension, migration among others, it is not considered broadly. An initial approach to handle reconfiguration operations in the VNF-FG is to consider a dynamic placement of VNFs while setting up the VNF-FG such that virtual functions are unknown beforehand. This approach was explored with a traffic steering scheme that allows flexibility during the setup of the VNF-FG [32].The migration operation

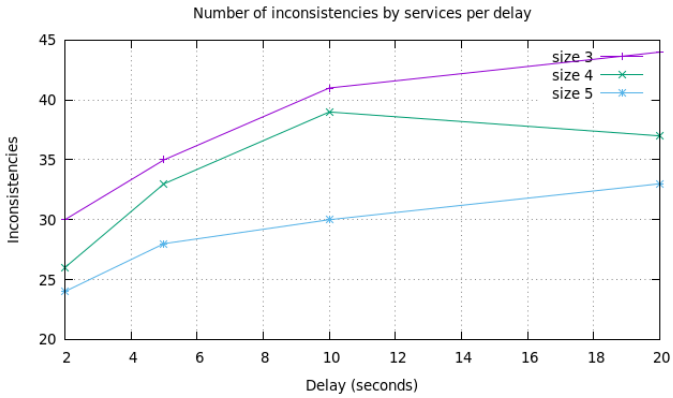

Fig. 6. Number of inconsistencies by services per delay. As the delay is lower, the traditional approach has few inconsistencies. The size of the VNF-FG does not appear to change significantly the number of inconsistencies

was partially covered with an approach that tried to optimally place the VNFs so that reconfiguration overhead is minimum. Their idea was to reuse VNFs to handle the online problem [15]. The extension of already deployed network services and function graphs to respond new demands while satisfying the services' constraints was proposed in [33]. Extensions can be triggered by new request or product of a life cycle management operation. The authors proposed an optimisation approach to minimise overhead during the extension. Update of VNF-FG was also proposed in [34] to insert, remove, replace or migrate new VNFs in existing forwarding paths. The authors proposed approximation algorithms to minimise the overhead required for these updates. In all of these previous approaches, the authors work on a single domain approach which limits the applicability to next generation services, thus, multi-domain approaches were proposed.

The multi-domain approach is fairly recent in literature and not many approaches are considered for the update of VNF-FG. However, there are some works that are considering this approach. An adaptive allocation for the embedding of the VNF-FG is proposed in [16]. The authors try to reallocate the VNF-FGs and minimise the time overhead using both centralized and decentralized approaches. A distributed chaining approach was considered in [17] where orchestrators coordinate themselves; however their approach is limited to the initial placement and execution of VNF-FGs. A deep reinforcement learning approach to achieve VNF-FG embedding is proposed in [12]. Blockchain has also been explored for the placement of VNFs in multi-domain orchestration [35]. A collaborative content delivery network through blockchain is studied in [36]. The authors present a simulation highlighting the advantages of decentralized solutions. VNF placement in the edge was addressed in [37]. Through a smart contract the orchestrators try to select the best PoPs for the services. Brokering was also studied in [38] where a proof of concept gives insight in the use of ledger technologies to NFV and 5G. Despite presenting a compelling case for the use of blockchain in multi-domain orchestration focused on security, these works only present the placement of VNFs and do not address the reconfiguration of VNF-FGs and the consistency properties required. To the best of our knowledge, only one work addresses consistency in the context of NFV [39]. The authors proposed a formal method for verification of NFV-enabled network services. They identify consistency properties such as policy and loop-free VNF-FGs. However, they don't consider the inconsistencies that can ocurr during the reconfiguration of VNF-FGs such as in scaling, migration or update with shared VNFs for multi-domain environments.

VIII. Conclusion

This paper highlights and addresses the existing limitations of Network Function Forwarding Graph (VNF-FG) management with distributed orchestrators in VNF multi-domain setting. Specifically, within a close network federation, this proposal focuses on preventing any consistency that might occur during VNF-FG update due to asynchronous channels and/or limited knowledge in the federation. To that end, VNF-FG model were extend to support additional causal dependencies among the VNFs according to the VNF-FG they belong to. The proposed algorithm was implemented using NFV open source tools. Several simulations were defined and implemented to evaluate and compare this proposal to the traditional eventual consistency guarantees. The number of inconsistencies, the message overhead count and average waiting time are among the considered metrics. The obtained results show that the proposed algorithm eliminates the inconsistencies in dependent VNFs with low message overhead as compared to the eventual consistency.

References

[1] R. Mijumbi, J. Serrat, J. Gorricho, N. Bouten, F. De Turck, and R. Boutaba. Network Function Virtualization: State-of-the-Art and Research Challenges. *IEEE Communications Surveys Tutorials*, 18(1):236–262, Firstquarter 2016.

[2] Jiuyue Cao, Yan Zhang, Wei An, Xin Chen, Jiyan Sun, and Yanni Han. Vnf-fg design and vnf placement for 5g mobile networks. *Science China Information Sciences*, 60(4):040302, 2017.

[3] NFVISG ETSI. Network Functions Virtualisation (NFV); Network Service Templates Specification. *ETSI GS NFV-IFA*, 14:V2, 2016.

[4] Menglu Zeng, Wenjian Fang, and Zuqing Zhu. Orchestrating Tree-Type VNF Forwarding Graphs in Inter-DC Elastic Optical Networks. *J. Lightwave Technol.*, 34(14):3330–3341, Jul 2016.

[5] M. C. Luizelli, L. R. Bays, L. S. Buriol, M. P. Barcellos, and L. P. Gaspary. Piecing together the NFV provisioning puzzle: Efficient placement and chaining of virtual network functions. In *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pages 98–106, May 2015.

[6] Deval Bhamare, Raj Jain, Mohammed Samaka, and Aiman Erbad. A survey on service function chaining. *Journal of Network and Computer Applications*, 75:138–155, 2016.

[7] P. T. A. Quang, A. Bradai, K. D. Singh, G. Picard, and R. Riggio. Single and Multi-Domain Adaptive Allocation Algorithms for VNF Forwarding Graph Embedding. *IEEE Transactions on Network and Service Management*, 16(1):98–112, March 2019.

[8] Nathan F. Saraiva de Sousa, Danny A. Lachos Perez, Raphael V. Rosa, Mateus A.S. Santos, and Christian Esteve Rothenberg. Network Service Orchestration: A survey. *Computer Communications*, 142-143:69 – 94, 2019.

[9] Cisco Visual Networking Index. Global mobile data traffic forecast update, 2016–2021 white paper. *Cisco: San Jose, CA, USA*, 2017.

[10] Mamta Agiwal, Abhishek Roy, and Navrati Saxena. Next generation 5G wireless networks: A comprehensive survey. *IEEE Communications Surveys & Tutorials*, 18(3):1617–1655, 2016.

[11] R. V. Rosa, M. A. S. Santos, and C. E. Rothenberg. MD2-NFV: The case for multi-domain distributed network functions virtualization. In *2015 International Conference and Workshops on Networked Systems (NetSys)*, pages 1–5, March 2015.

[12] Quang Tran Anh Pham, Abbas Bradai, Kamal Deep Singh, and Yassine Hadjadj-Aoul. Multi-domain non-cooperative VNF-FG embedding: A deep reinforcement learning approach. In *IEEE Infocom 2019-IEEE International Conference on Computer Communications*, 2019.

[13] Oussama Soualah, Marouen Mechtri, Chaima Ghribi, and Djamal Zeghlache. A green vnf-fg embedding algorithm. In *2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft)*, pages 141–149. IEEE, 2018.

[14] Quang Tran Anh Pham, Yassine Hadjadj-Aoul, and Abdelkader Outtagarts. Evolutionary actor-multi-critic model for vnf-fg embedding. In *IEEE Consumer Communications & Networking Conference (CCNC 2020)*, 2020.

[15] Narjes Tahghigh Jahromi, Somayeh Kianpisheh, and Roch H Glitho. Online VNF placement and chaining for value-added services in content delivery networks. In *2018 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN)*, pages 19–24. IEEE, 2018.

[16] Pham Tran Anh Quang, Abbas Bradai, Kamal Deep Singh, Gauthier Picard, and Roberto Riggio. Single and multi-domain adaptive allocation algorithms for vnf forwarding graph embedding. *IEEE Transactions on Network and Service Management*, 16(1):98–112, 2018.

[17] Milad Ghaznavi, Nashid Shahriar, Shahin Kamali, Reaz Ahmed, and Raouf Boutaba. Distributed service function chaining. *IEEE Journal on Selected Areas in Communications*, 35(11):2479–2489, 2017.

[18] Flávio EA Horita, João Porto de Albuquerque, and Victor Marchezini. Understanding the decision-making process in disaster risk monitoring and early-warning: A case study within a control room in Brazil. *International journal of disaster risk reduction*, 28:22–31, 2018.

[19] Kenneth B Wells, Jennifer Tang, Elizabeth Lizaola, Felica Jones, Arleen Brown, Alix Stayton, Malcolm Williams, Anita Chandra, David Eisenman, Stella Fogleman, et al. Applying community engagement to disaster planning: developing the vision and design for the Los Angeles County Community Disaster Resilience initiative. *American Journal of Public Health*, 103(7):1172–1180, 2013.

[20] Jayanrtha Withanaarachchi and Sujeeva Setunge. Influence of decision making during disasters and how it impacts a community. 2014.

[21] Thomas Ludwig, Christian Reuter, Tim Siebigteroth, and Volkmar Pipek. Crowdmonitor: Mobile crowd sensing for assessing physical and digital activities of citizens during emergencies. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, pages 4083–4092, 2015.

[22] Jinwei Liu, Haiying Shen, Husnu S Narman, Wingyan Chung, and Zongfang Lin. A survey of mobile crowdsensing techniques: A critical component for the internet of things. *ACM Transactions on Cyber-Physical Systems*, 2(3):1–26, 2018.

[23] Raphael Vicente Rosa, Mateus Augusto Silva Santos, and Christian Esteve Rothenberg. MD2-NFV: The case for multi-domain distributed network functions virtualization. In *2015 International Conference and Workshops on Networked Systems (NetSys)*, pages 1–5. IEEE, 2015.

[24] Xi Li, Josep Mangues-Bafalluy, Iñaki Pascual, Giada Landi, Francesca Moscatelli, Kiril Antevski, Carlos J Bernardos, Luca Valcarenghi, Barbara Martini, Carla Fabiana Chiasserini, et al. Service orchestration and federation for verticals. In *2018 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, pages 260–265. IEEE, 2018.

[25] Yoonseon Han, Thomas Vachuska, Ali Al-Shabibi, Jian Li, Huibai Huang, William Snow, and James Won-Ki Hong. ONVisor: Towards a scalable and flexible SDN-based network virtualization platform on ONOS. *International Journal of Network Management*, 28(2):e2012, 2018.

[26] Juliver Gil Herrera and Juan Felipe Botero. Resource allocation in NFV: A comprehensive survey. *IEEE Transactions on Network and Service Management*, 13(3):518–532, 2016.

[27] M. Peuster, S. Draxler, H. R. Kouchaksaraei, S. v. Rossem, W. Tavernier, and H. Karl. A flexible multi-pop infrastructure emulator for carrier-grade MANO systems. In *2017 IEEE Conference on Network Softwarization (NetSoft)*, pages 1–3, July 2017.

[28] Bruno Rodrigues, Thomas Bocek, and Burkhard Stiller. Multi-domain ddos mitigation based on blockchains. In *IFIP International Conference on Autonomous Infrastructure, Management and Security*, pages 185–190. Springer, 2017.

[29] Gabriel Antonio F Rebello, Igor D Alvarenga, Igor J Sanz, and Otto Carlos MB Duarte. BSec-NFVO: A blockchain-based security for network function virtualization orchestration. In *ICC 2019-2019 IEEE International Conference on Communications (ICC)*, pages 1–6. IEEE, 2019.

[30] Igor D Alvarenga, Gabriel AF Rebello, and Otto Carlos MB Duarte. Securing configuration management and migration of virtual network functions using blockchain. In *NOMS 2018-2018 IEEE/IFIP Network Operations and Management Symposium*, pages 1–9. IEEE, 2018.

[31] Ghasem Mirjalily and LUO Zhiquan. Optimal Network Function Virtualization and Service Function Chaining: A Survey. *Chinese Journal of Electronics*, 27(4):704–717, 2018.

[32] Cristina K Dominicini, Gilmar L Vassoler, Rodolfo Valentim, Rodolfo S Villaca, Moisés RN Ribeiro, Magnos Martinello, and Eduardo Zambon. KeySFC: Traffic Steering using Strict Source Routing for Dynamic and Efficient Network Orchestration. *Computer Networks*, page 106975, 2019.

[33] Omar Houidi, Oussama Soualah, Wajdi Louati, Djamal Zeghlache, and Farouk Kamoun. Virtualized network services extension algorithms. In *2018 IEEE 17th International Symposium on Network Computing and Applications (NCA)*, pages 1–5. IEEE, 2018.

[34] Selma Khebbache, Makhlouf Hadji, and Djamal Zeghlache. Dynamic Placement of Extended Service Function Chains: Steiner-based Approximation Algorithms. In *2018 IEEE 43rd Conference on Local Computer Networks (LCN)*, pages 307–310. IEEE, 2018.

[35] Raphael Rosa and Christian Esteve Rothenberg. Blockchain-based decentralized applications for multiple administrative domain networking. *IEEE Communications Standards Magazine*, 2(3):29–37, 2018.

[36] Nicolas Herbaut and Nicolas Negru. A model for collaborative blockchain-based video delivery relying on advanced network services chains. *IEEE Communications Magazine*, 55(9):70–76, 2017.

[37] He Zhu, Changcheng Huang, and Jiayu Zhou. EdgeChain: Blockchain-based multi-vendor mobile edge application placement. In *2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft)*, pages 222–226. IEEE, 2018.

[38] Boubakr Nour, Adlen Ksentini, Nicolas Herbaut, Pantelis A Frangoudis, and Hassine Moungla. A blockchain-based network slice broker for 5G services. *IEEE Networking Letters*, 1(3):99–102, 2019.

[39] Myung-Ki Shin, Yunchul Choi, Hee Hwan Kwak, Sangheon Pack, Miyoung Kang, and Jin-Young Choi. Verification for NFV-enabled network services. In *2015 International Conference on Information and Communication Technology Convergence (ICTC)*, pages 810–815. IEEE, 2015.