A Near Optimal Reliable Orchestration Approach for Geo-Distributed Latency-Sensitive SFCs

Dmitrii Chemodanov*⋆, Prasad Calyam*, Flavio Esposito†,
Ronald McGarvey*, Kannappan Palaniappan*, Antonio Pescapé‡
*University of Missouri-Columbia, USA; †Saint Louis University, USA; ‡University of Napoli Federico II, Italy.

***Abstract*—Traditionally, Network Function Virtualization uses Service Function Chaining (SFC) to *place* service functions and *chain* them with corresponding flows allocation. With the advent of Edge computing and IoT, a *reliable orchestration* of latency-sensitive SFCs is needed to *compose* and *maintain* them in geo-distributed cloud infrastructures. However, the optimal SFC composition in this case becomes the NP-hard integer *multi-commodity-chain* flow (MCCF) problem that has no known approximation guarantees. In this paper, we first outline our novel practical and near optimal SFC composition scheme which is based on our novel *metapath composite variable* approach, admits end-to-end network QoS constraints (e.g., latency) and reaches 99% optimality on average in seconds for practically sized geo-distributed cloud infrastructures. We then propose a novel metapath-based SFC maintenance algorithm that guarantees a distributed control plane consistency without use of expensive consensus protocols. Using trace-driven simulations comprising of challenging disaster-incident conditions, we show that our solution composes twice as many SFCs and uses ∼10x less control messages than state-of-the-art methods. Finally, experimental evaluations of our SFC orchestration prototype deployed on a realistic cloud/edge computing testbed show significant speed-ups (up to 3.5x) for our case-study geo-distributed latency-sensitive object tracking pipeline w.r.t. its IP-based cloud computing alternative.**

***Index Terms*—NFV, geo-distributed latency-sensitive SFCs, reliable SFC orchestration, multi-commodity-chain flow problem**

I. Introduction

Nowadays, Network Function Virtualization (NFV) is an attractive paradigm for network operators to dynamically *place* virtualized network functions (*e.g.*, firewalls, load-balancers, etc.), *chain* them for a service flow routing and allocate corresponding compute/network resources in a cloud infrastructure by utilizing SFCs [1]. Recently, areas such as Microservices [2], Mobile Edge Computing [3] and Computer Vision Analytics [4] have also shown benefits of adopting the SFC technology. With the advent of Edge (or Fog) computing that augments cloud Application Programming Interfaces closer to the end-user IoT devices, SFCs can be now 'composed' from both core and edge cloud resources forming geo-distributed chains to satisfy geo-location and latency requirements of their functions [3], [5], [6]. Our *case study* example of a geo-distributed latency-sensitive SFC which is utilized for the computer vision of a real-time object tracking pipeline is shown in Figure 1. The pre-processing and Human-Computer Interaction analysis functions are placed for a low-latency access on edge servers, and the tracking function is placed on a cloud server for the compute-intensive processing. Microservices in this case allow users to have a low-latency access to processed data while waiting for new data from the main tracking pipeline. At the same time, using low-latency pre- processing functions allows the processing to exclude raw images without waiting for the main tracking pipeline to catch-up in cases such as e.g., if the desired target is missing, or if an image quality is poor [4].

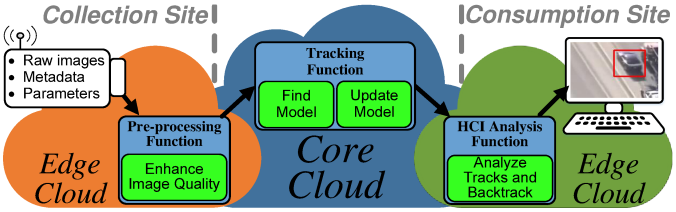


Fig. 1: Illustrative example of the geo-distributed latency-sensitive SFC used for the real-time object tracking pipeline [4].

However, geo-distributed SFCs can be subject to node failures and congested network paths leading to their frequent Quality of Service (QoS) demands violations [7]. In some specific cases of natural or man-made disaster-incidents, they can be subject to severe infrastructure outages [8]. Moreover, computation and network QoS demands of SFCs can fluctuate themselves [9], e.g., due to interference of co-located functions [6], [10]. Thus, a *reliable orchestration* of SFCs is needed to cope both proactively upon their *composition* as well as reactively during their lifespan *maintenance* with potential SFC demand fluctuations [9] as well as with possible infrastructure outages [7], [8].

At the same time, providing a continuously available and reliable service chain orchestration is hard [1], and it has engendered prior efforts on development of new algorithms to design special hardware accelerators [11]. First of all, this is because the optimal SFC composition has known approximation guarantees only in some special cases. For instance, when chaining of service functions (*e.g.*, satisfying bandwidth requirements between two consequent functions) [12], [13] and/or their ordering (*i.e.*, ensuring that function A precedes function B) [14], [15] are omitted. In the general case however, it requires solving of the NP-hard integer MCCF problem to align flow splits with supported hardware granularity [16]. It is also necessary to support cases when service functions or their associated flows are non-splittable. This problem has no known approximation guarantees and has been previously reported as the integer NFV service distribution problem [17]. Moreover, its complexity can be further exacerbated by incor-

⋆Corresponding and primary author (dycbt4@mail.missouri.edu).

porated reliability and geo-location/latency aware mechanisms. The former aims to cope proactively with both possible infrastructure outages as well as SFC demand fluctuations, whereas the latter is needed to satisfy QoS demands of geo-distributed latency-sensitive service chains.

Secondly, due to scale and nature of geo-distributed SFCs, having their orchestration with a single point of failure or congestion ($i.e.$, centralized) is too risky. On the one hand, having a SFC *maintenance* which is based on a distributed control plane system is crucial. On the other hand, distributed control plane-based Software Defined Networking (SDN) systems such as ONOS [18] require consistency guarantees to avoid various violations [19]. Examples of such violations tailored to the SFC orchestration can be double assignment of services, looping paths, QoS constraints violations and others. A common approach to guarantee consistency of a distributed control plane is by establishing a consensus. The latter can be done by running known consensus protocols such as Raft [20], etc. To the best of our knowledge, the only distributed SFC orchestration scheme that builds upon consensus literature is 'Catena' [21]. This algorithm ensures a consistency by running consensus protocols based on specified policies and can be used safely within distributed control plane. However, using consensus protocols for every service placement or its chaining in a SFC request is expensive.

Our approach: In this paper, we propose a new *reliable service chain orchestration* approach that can serve the needs of geo-distributed latency-sensitive SFCs at high-scale. Our orchestration approach assumes an initial SFC *composition* via a centralized control plane and its consequent *maintenance* during its lifespan within a distributed control plane.

Firstly, our approach involves ensuring reliability proactively. To do so, we *compose* SFCs with capacity chance-constraints (that handle both SFC demand fluctuations as well as infrastructure outages uncertainties) and with backup policies which further complicate solution of the NP-hard integer MCCF problem. We remark that our approach does not focus on the optimal composition of SFCs with their "guarantee" reliability. This is because this optimization problem scales poorly [22]. Instead, we propose a policy-based reliability mechanism that can trade-off SFCs' reliability and their composition optimality.

Secondly, to cope with this problem solution intractabilities, we outline our novel *metapath-based composite variable* approach that has been proposed in our prior work [23]. This approach is similar to other composite variable solutions in terms of its nature that aggregates multiple decisions within a single binary variable [24].

Finally, to *maintain* QoS demands satisfaction of SFCs during their lifespan, we propose a distributed control plane algorithm that builds upon metapaths and Simple Coordination Layer (SCL) [19].

Contributions: *In this paper, we augment our previously proposed practical and near optimal SFC composition approach with a novel SFC maintenance algorithm designed for a distributed control plane that guarantees the latter's consistency without use of expensive consensus protocols.* [23] Specifically, our contributions are the following:

(1) We formulate the (master) NP-hard integer MCCF problem previously adopted in NFV literature [17] but now with geo-location and latency constraints as well as with probabilistic capacity constraints for a reliable composition under uncertainty of geo-distributed latency-sensitive SFCs. (Section III)

(2) We outline our first-of-its-kind *metapath-based composite variable* approach that aggregates feasible mapping decisions of each single-link SFC segment as a set of k-constrained shortest metapaths. It then assigns SFC segments to their associated metapaths either optimally by using generalized assignment problem (GAP) [25] or suboptimally by using its (polynomial) Lagrangian relaxation counterpart. (Section IV)

(3) We also propose a new *metapath-based SFC maintenance* algorithm that operates in a distributed control plane and prove its *eventual correctness* property to use SCL. (Section V)

(4) Using trace-driven simulations of real US Tier-1 ($\sim$300 nodes) and regional ($\sim$600 nodes) infrastructure providers' topologies, we first show how our SFC composition approach achieves 99% optimality on average. In addition, we show that it only takes time on the order of seconds for practically sized problems in contrast with the master problem solution that takes several hours. By recreating challenging disaster incident scenarios as in [8], we lastly show how our approach can compose twice as many sequentially incoming SFC requests and better maintain them in terms of optimality and number of control messages due to use of metapaths and SCL than the state-of-the-art solutions [21], [22], [26]. (Section VII)

(5) Finally, using experimental evaluation, we implement our open-source *reliable service chain orchestration* prototype (available under GNU license at [27]) and show how it speeds-up the case study geo-distributed latency-sensitive object tracking pipeline by up to 3.5 times in comparison to its cloud computing alternative over IP networks. To this end, we deploy our prototype on a realistic core/edge cloud testbed in GENI infrastructure [28]. (Section VII-B)

II. Related work

SFC is traditionally used in NFV to place a set of middleboxes and chain relevant functions to steer traffic through them [1]. Existing SFC solutions either separate the service placement from the service chaining phase [12], [14], [15], or jointly optimize both the two phases [9], [17].

SFC Optimality. In some special cases the optimal SFC is shown to have approximation guarantees [12], [13], [14], [15]. For instance, Cohen et al. [14] and Sang et al. [15] provide near optimal approximation algorithms for the SFC problem without chaining and ordering constraints. Tomassilli et al. [12] propose the first SFC solution with the approximation guarantees which admits ordering constraints, but still omits chaining constraints. Similarly, omitting chaining constraints allows Zhang et al. [13] to achieve in the worst case a 2x approximation bound of SFC in polynomial time. Q. Zhang et al. [6] propose an interference-aware approximation algorithm for 5G network services placement of linear, split and split-and-merged SFCs, however, without considering their chaining constraints. Guo et al. [29] show approximation guarantees for SFCs with both ordering and chaining constraints, but only under assumptions that available service chaining options are of polynomial size. In the general case however, when service functions need to be jointly placed and chained in a geo-distributed cloud infrastructure with a corresponding compute/network resource allocation, possible SFC compositions

are of exponential size. Thus, it becomes a linear topology Virtual Network Embedding (VNE) [3], [30] and can be formulated as the (NP-hard) MCCF problem with integrality constraints with no known approximation guarantees [17]. Thus, Feng et al. [17] propose a heuristic algorithm whose preliminary evaluation results in a small-scale network settings (of $\sim$10 nodes) shows promise for providing efficient solutions to the integer MCCF problem in practical settings.

In this paper, we propose the first to our knowledge practical and near optimal SFC composition approach in the general case of joint service function placement and chaining in a geo-distributed cloud infrastructure that also admits end-to-end network QoS constraints such as latency, packet loss, etc. To this aim, we propose a novel *metapath composite variable approach* which reduces a combinatorial complexity of the (master) integer MCCF problem. As a result, our approach achieves 99% optimality on average and takes seconds to compose SFCs for practically sized problems of US Tier$-$1 ($\sim$300 nodes) and regional ($\sim$600 nodes) infrastructure providers' topologies, where master problem solution takes hours using a High Performance Computing cloud server.

SFC Reliability. With the advent of edge networking, 5G and growing number of latency sensitive services, recent works consider problems of geo-distributed [31] and edge SFC [5], [6]. Although these works mainly focus on the load balancing [32] and latency optimization techniques [5], they omit an important reliability aspect of geo-distributed latency-sensitive SFCs. The closest works related to ours are [9] and [22]. Fei et al. [9] propose a prediction-based approach that proactively handles SFC demand fluctuations. Similarly, Xiao et al. [33] propose a deep reinforcement learning approach *viz.*, 'NFVdeep' to cope with both SFC demands and network traffic uncertainties. However, their approach does not account for network/infrastructure outages that mainly cause service function failures [7]. At the same time, Spinnewyn et al. [22] propose a SFC solution that ensures a sufficient infrastructure reliability, but neither proactively nor reactively handles SFC demand fluctuations.

In contrast to [9] and [22], our reliable composition scheme uniquely ensures reliability of geo-distributed latency-sensitive SFCs by using of chance-constraints and backup policies to cope with both SFC demand fluctuations and infrastructure outages [23]. It then also *maintains* them during their lifespan to ensure reliability of already composed service chains ($i.e.$, reactively) by utilizing a distributed control plane to avoid a single point of failure or congestion. We also found only one SFC orchestration approach that guarantees a distributed control plane consistency during service chain orchestration — Catena [21]. In contrast to Catena that uses expensive consensus protocols, we prove our metapath-based maintenance algorithm *eventual correctness* to use SCL [19].

III. Modeling Reliable Service Chain Composition

In this section, we define the problem of joint *SFC composition* that can be formulated as the integer MCCF problem for an augmented cloud infrastructure graph [17] which is a generalization of a well-known multi-commodity flow problem [30]. To proactively ensure reliability of a SFC composition, we use backup policies as well as probabilistic 'chance' capacity constraints instead of deterministic ones. Thus, we use a chance-constrained programming [34]. We also extend this problem with geo-location and latency constraints to satisfy all QoS demands of geo-distributed latency-sensitive SFCs.

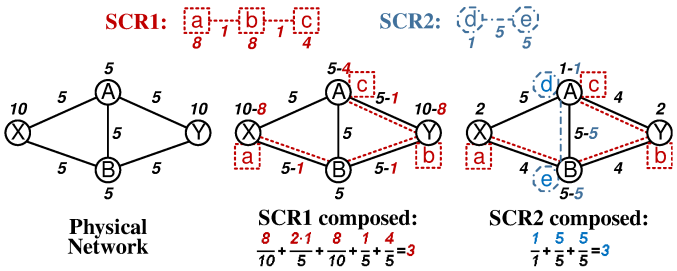


Fig. 2: Illustrative example of the *online composition* of service chain requests (SCR) on top of the capacitated physical network (numbers indicate service demands and corresponding resource capacities).

Objective and example of the online chain composition. Based on providers' policies, the service chain composition problem can be used to minimize (expected) values of different fitness functions $F_{\mathbb{E}}$. One example of common fitness functions is an additive function of ratios of service chain demands and corresponding physical resource capacities. Such function is known to best balance the physical network load [30]. In most cases service chain requests can be unknown in advance, and using the load balancing fitness function allows one to increase the acceptance ratio of these requests. Such optimization is also known as the 'online optimization' [26], [30].

Figure 2 shows an example of the online SFC composition that minimizes the network load balancing function: by minimizing a sum of SFC demands and corresponding physical resource capacity ratios, for $a-b-c$ service chain we achieve its minimum value $F_{\mathbb{E}} = \frac{8}{10} + \frac{2\cdot1}{5} + \frac{8}{10} + \frac{1}{5} + \frac{4}{5} = 3$. As a result, we compose this service chain request with X, Y and A physical nodes ($e.g.$, servers) to place a, b and c services, respectively. To enable service communications $a-b$ and $b-c$, we chain them with $X-B-Y$ and $Y-A$ physical paths, respectively. This also allows us to compose the subsequent $d-e$ request.

In the rest of this paper, our objective is to minimize the expected value of the load balancing fitness function $F_{\mathbb{E}}$ formally defined below (see Equation 1) for the case of SFC demand and available physical resource uncertainties.

Service chain composition sets and variables. We model each SFC $a \in A$ as a chain graph $G^a = (N_V^a, E_V^a)$. A service G^a is composed by a set N_V^a of services and a set E_V^a of corresponding service communications (or service links) representing logical network connectivities among elements in N_V^a. Moreover, each SFC a has a set of primary and backup resources denoted as B^a. Note that we assume here full backup policy, $i.e.$, each individual SFC replica should be able to handle SFC workload by itself. We then model the physical infrastructure on which the service functions run as a physical network graph $G = (N_S, E_S)$, composed by a set N_S of substrate nodes and a set E_S of substrate edges.

We define two types of binary variables: one for the service chain link mapping, and another for (node) service mapping. Particularly, let binary variable $f_{ij}^{st}(b,a) = 1$ if a flow for $st \in E_V^a$ service link of a backup $b \in B^a$ of a SFC $a \in A$ is assigned to the physical edge $ij \in E_S$, $i.e.$, $f_{ij}^{st}(b,a) = 1$, or 0 otherwise. Furthermore, let binary variable $x_i^s(b,a) = 1$

if a service $s \in N_V^a$ of a backup $b \in B^a$ of a SFC $a \in A$ is assigned to the physical node $i \in N_S$, *i.e.*, $x_i^s(b,a) = 1$, and 0 otherwise. Having sets and variables defined, we now can formulate the online service chain composition problem under uncertainty using integer MCCF problem.

Problem 1 (online SFC composition under uncertainty)**.** *Given a set of SFCs represented as graphs $G^a = (N_V^a, E_V^a)$ and a physical network graph $G = (N_S, E_S)$, the online service chain composition problem under SFC demands and available physical resources uncertainties can be formulated as follows:*

$$\min F_{\mathbb{E}} = \sum_{a\in A}\sum_{b\in B^a}\left(\sum_{s\in N_V^a}\sum_{\tau\in T}\sum_{i\in N_S}\mathbb{E}\left[\frac{\mathbf{D}_s^{a\tau}}{\mathbf{C}_i^{\tau}}\right]x_i^s(b,a) + \sum_{st\in E_V^a}\sum_{ij\in E_S}\mathbb{E}\left[\frac{\mathbf{D}_{st}^{a}}{\mathbf{C}_{ij}}\right]f_{ji}^{st}(b,a)\right) \tag{1}$$

subject to

Service Placement Constraints:

$$\sum_{i\in N_S} x_i^s(b,a) = 1, \forall s \in N_V^a, b \in B^a, a \in A \tag{2}$$

$$\mathbb{P}\left[\sum_{a\in A}\sum_{b\in B^a}\sum_{s\in N_V^a}\mathbf{D}_s^{a\tau}x_i^s(b,a) \leq \mathbf{C}_i^{\tau}\right] \geq R, \forall i \in N_S, \tau \in T \tag{3}$$

Service Chaining Constraints:

$$\sum_{j\in N_S} f_{ij}^{st}(b,a) - \sum_{j\in N_S} f_{ji}^{st}(b,a) = x_i^s(b,a) - x_i^t(b,a), \quad \forall i \in N_S, st \in E_V^a, b \in B^a, a \in A \tag{4}$$

$$\mathbb{P}\left[\sum_{a\in A}\sum_{b\in B^a}\sum_{st\in E_V^a}\mathbf{D}_{st}^{a}f_{ij}^{st}(b,a) \leq \mathbf{C}_{ij}\right] \geq R, \forall ij \in E_S \tag{5}$$

Specific QoS Constraints (Geo-Location, Latency, etc.):

$$gd_{si}^a x_i^s(b,a) \leq \mathcal{GD}_s^a, \forall i \in N_S, s \in N_V^a, b \in B^a, a \in A, \tag{6}$$

$$\sum_{ij\in E_S} w_{ij}^k f_{ij}^{st}(b,a) \leq \mathcal{K}_{st}^{ak}, \forall st \in E_V^a, b \in B^a, a \in A, k \in K \tag{7}$$

Additional Policy Constraints (*e.g.*, No-Consolidation):

$$\sum_{b\in B^a}\sum_{s\in N_V^a} x_i^s(b,a) \leq 1, \forall i \in N_S, a \in A \tag{8}$$

where symbols and notations of sets, parameters, variables and functions are summarized in Table I.

SFC composition constraints discussion. Minimization of $F_{\mathbb{E}}$ in Equation 1 is subject to a set of constraints which contains both — basic composition constraints and constraints specific to the geo-distributed latency-sensitive SFCs. The basic constraints include service *placement* for a specified number of duplicates (Equation 2), service *chaining* or well-known multi-commodity flow constraints (Equation 4). Additional policy constraints for service chain composition problem are also acceptable. One such example is a common 'no consolidated service placement' constraint that prohibits placement of two or more different services (or their backups) belonging to the same service chain onto one physical node (Equation 8). Note that this policy further complicates a combinatorial complexity of the (NP-hard) integer MCCF problem. In contrast to prior SFC composition problems [17], [22], we now use probabilistic physical node and link capacity constraints to ensure that physical resources satisfy SFC QoS demands given some acceptable risk (Equations 3 and 5).

The specific geo-distributed latency-sensitive SFC constraints include physical node geo-location and service communication end-to-end network QoS constraints such as latency, packet loss, etc. (Equation 7).

SFC reliability and chance-constraints discussion. It is know that even infrastructures with very high availability are subject to traffic loss and outages [16]. Thus, both the reliability of SFCs and provided QoS are naturally coupled. To address this, we use chance-constrained programming [34] that

TABLE I: Symbols and Notations

Service Chain Composition: Sets	
A	≜Set of SFCs that needs to be composed
B^a	≜Set of primary and backup replicas for the SFC a
N_V^a	≜Set of service functions composing the SFC a
E_V^a	≜Set of service communications in the SFC a that create a service chain
$\mathcal{P}_a^{st}$	≜Set of metapaths for the st (single-link) service chain segment of SFC a
T	≜Set of QoS demands for service functions such as CPU, memory, storage, etc
K	≜Set of end-to-end network QoS demands for SFC communications such as latency, losses, jitter, etc
N_S	≜Set of physical nodes within the infrastructure
E_S	≜Set of physical links within the infrastructure
Service Chain Composition: Variables	
$x_i^s(b,a)$	≜Binary variable that equals to 1 if the service s backup b for the SFC a is placed on the physical node i
$f_{ij}^{st}(b,a)$	≜Binary variable that equals to 1 if the communication backup b between services s and t for the SFC a is placed on the physical link ij
$f_{ijk}^{st}(b,a)$	≜Binary variable that equals to 1 if the single-link chain segment st is assigned to the metapath P_{ijk}^{st} of the SFC a backup b
$\mathbf{D}_s^{a\tau}$	≜Random variable that corresponds to the SFC a service s QoS demand $\tau \in T$
$\mathbf{D}_{st}^{a}$	≜Random variable that corresponds to the communication bandwidth demand between services s and t for the SFC a
$\mathbf{C}_i^{\tau}$	≜Discrete random variable that corresponds to the possible physical node i capacity of $\tau \in T$ resource, *i.e.*, $\mathbf{C}_i^{\tau} \in \{0, C_i^{\tau}\}$
$\mathbf{C}_{ij}$	≜Discrete random variable that corresponds to the possible physical link ij capacity, *i.e.*, $\mathbf{C}_{ij} \in \{0, C_{ij}\}$
Service Chain Composition: Parameters	
gd_{si}^a	≜Parameter that corresponds to the geographical distance between the desired location of the service s in the SFC a and the physical node i location
$\mathcal{GD}_s^a$	≜Parameter that corresponds to the maximum allowable geographical distance between the desired location of the service s for the SFC a and some physical node location
w_{ij}^k	≜Parameter that corresponds to the additive weight of the physical link ij (or multiplicative when composed with *log* function) of the service communication end-to-end $k \in K$ QoS constraint (*e.g.*, latency)
$\mathcal{K}_{st}^{ak}$	≜Parameter that corresponds to the communication end-to-end $k \in K$ QoS constraint (*e.g.*, latency) between services s and t of the SFC a
R	≜SFC reliability probability that a chance-constraint will be satisfied
K_α	≜Constant in a standard Normal distribution table corresponding to desired α probability
$\mu_s^{a\tau}$	≜Expected (or mean) value of the SFC a service s QoS demand $\tau \in T$
μ_{st}^{a}	≜Expected (or mean) value of the the communication bandwidth demand between services s and t for the SFC a
$\sigma_s^{a\tau}$	≜Variance of the SFC a service s QoS demand $\tau \in T$
σ_{st}^{a}	≜Variance of the the communication bandwidth demand between services s and t for the SFC a
C_i^{τ}	≜Physical node i capacity of $\tau \in T$ resource (*e.g.*, CPU)
C_{ij}	≜Physical link ij capacity

guarantees QoS satisfaction at a given infrastructure segment with a set reliability level. To this aim, we use a policy-based reliability for the SFC composition, i.e., we allow for policy specifications of chance-constraints acceptable risks and service backups.

For instance, by decreasing an acceptable risk and/or increasing number of backups, we can leverage the overall probability of a SFC disruption that requires its re-composition ($e.g.$, migration of virtual resources) during its maintenance. For example, given a risk of 5%, $i.e.$, $R = 0.95$, and 5 services for a single SFC, the lower bound probability that its demands will be satisfied is $P_{lb} = R^5 = 0.95^5 \approx 0.77$ not considering inter-service communication demands and not allowing backup resources. Thus, approximately in 1 out of 5 cases the service chain needs to be re-composed. Alternatively, if we at least duplicate the service chain physical resources ($i.e.$, compose 2 service chain backups), the lower bound probability that SFC demands will be satisfied by at least one of the duplicates becomes $P_{lb} = 1 - (1 - R^5)^2 \approx 0.95$. As a result, the service chain needs to be re-composed only in 1 out of 20 cases. However, with tighter reliability policies (i.e., the lower acceptable risk or the higher number of backups), fewer feasible solutions are available, and the optimal solution achieves a poorer objective value of the optimal solution, and thus, worse performance of the online service chain composition is realized. We show such reliability/performance trade-offs of our approach using trace-driven simulations in Section VII.

MCCF-based SFC composition intractabilities. When deterministic equivalents of the objective in Equation 1 as well as of capacity chance-constraints are known, we can use any integer programming solver ($e.g.$, CPLEX [35]) for Problem 1 to reliably compose all (known at a time) service chain requests. However, due to NP-hardness of this composition, the solution can be intractable for large-scale cloud/edge infrastructures. To improve its scalability limitations, existing column generation [26], heuristic [22] and metaheuristic [22] approaches can be used (often at expense of the master problem optimality). In the next section, we propose a near optimal *metapath composite variable* approach that simplifies a combinatorial complexity of the SFC composition outlined in Problem 1.

IV. Service Chain Composition via Metapaths

In this section, we outline our previously proposed *metapath-based composite variable* approach that aims to simplify the combinatorial complexity of the integer MCCF-based SFC composition problem [23]. Thus, similarly to existing composite variable schemes [24], we create a binary variable that composes multiple (preferably close to optimal) decisions. To this end, we build upon a known result in optimization theory: all network flow problems can be decomposed into paths and cycles [36]. We first introduce our notion of *metapath* and its relevance to the constrained shortest path problem [37], [38], [39]. We then use the constrained shortest metapaths to create variables with composite decisions for the SFC composition problem and discuss scalability improvements of this approach.

Metalinks and metapaths. Before defining the metapath, it is useful to introduce the idea of 'metalinks'. Metalinks have been widely adopted in prior NFV/VNE literature to solve optimally graph matching problems [17], [26]. A metalink is an augmentation link in a network graph. In our case, it represents the (potential) *feasible* placement of some service a on some physical node A, as shown in Figure 3. Formally, we have:

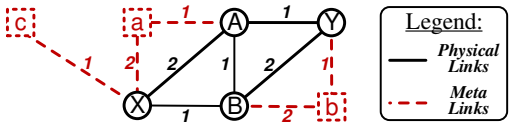


Fig. 3: Illustrative example of the augmented with metalinks physical network which represent feasible service a, b and c placements; numbers indicate fitness function values - red and black values annotate service placement and service chaining via some physical link, respectively.

Definition 1 (metalink)**.** *A link si for SFC $a \in A$ belongs to the set of metalinks E^a_M if and only if the service $s \in N^a_V$ of SFC a can be placed onto the node $i \in N_S$.*

Building on the definition of a metalink, we can define a metapath as the path that connects any two services through the physical network augmented with metalinks. For example, consider following metapaths $a-A-Y-b$ and $a-A-B-b$ shown in Figure 3. Formally, we have:

Definition 2 (metapath)**.** *The path P^{st}_{ij} is a metapath between services s and t for SFC $a \in A$ if and only if $\forall kl \in P^{st}_{ij}$: $kl \in E_S \vee kl \in \{si, tj\}$.*

Intuitively, metapath P^{st}_{ij} is formed by exactly two metalinks that connect s and t to the physical network and an arbitrary number of physical links $kl \in E_S$.

Constrained shortest metapaths. Having defined metapaths, let us consider a simple case of the SFC composition problem - composition of a single-link chain ($i.e.$, two services connected via a single virtual link): the optimal composition of a single-link chain can be seen as the *constrained shortest (meta)path* problem that connects two services via the augmented physical network, where all physical links have arbitrary fitness values of a service chaining (virtual link mapping) and all metalinks have arbitrary fitness values of a service placement divided by the number of neighboring services ($i.e.$, by 1 for a single-link chain). In our example, shown in Figure 3, the optimal single-link SFC $a - b$ composition can be represented by the constrained shortest metapath $a-A-Y-b$ that satisfies all SFC composition constraints with the overall fitness function of 3. Further, we prove our intuition formally:

Theorem IV.1. (The optimal single-link SFC composition) *The optimal single-link chain composition is the constrained shortest metapath.*

Proof. Assume the contrary. Let $L_1(s,t)$ be the optimal objective value of the single-link service chain st composition and $L_2(s,t)$ be a length of the constrained shortest metapath P_2. We need to show that $L_1 \neq L_2$:

Case 1 *($L_1 < L_2$):* In this case, $L_1(s,t)$ solution is mapping of services s and t to physical nodes i and j, respectively, and a service link st to a physical path $P(i,j)$ as defined in the service chain composition problem. Without loss of generality, we can assume that the optimal solution of the service chain composition problem is feasible. Hence, s and t mappings are si and tj metalinks by Definition 1, respectively. Furthermore, let us define the path $P_1 = P(si, P(i,j), jt)$

which by Definition 2 is a metapath. As the optimal solution is feasible, P_1 satisfies all constraints of the single-link chain st composition. Hence, P_1 is a constrained metapath whose length $L_1(s,t)$ is shorter than $L_2(s,t)$ contradicting that P_2 is the constrained shortest metapath.

Case 2 ***($L_1 > L_2$):*** In this case, we can present metapath P_2 as $P_2 = P(si, P(i,j), jt)$, where si and tj are metalinks, and $P(i,j)$ is a physical path (see Definition 1). Let us map services s and t on physical nodes i and j, respectively, and service link st on a physical path $P(i,j)$. As P_2 is the constrained metapath, this mapping is feasible with the objective value $L_2(s,t)$ less than $L_1(s,t)$ contradicting that $L_1(s,t)$ is the optimal objective value of the single-link service chain st composition. ■

Corollary IV.1. (The optimal single-link SFC composition complexity) *The optimal single-link SFC composition has a pseudo-polynomial complexity.*

Proof. Based on Theorem IV.1, the optimal single-link SFC is the constrained shortest metapath which is by Definition 2 the constrained shortest path in the augmented network graph. However, it is known that the constrained-shortest path can be found in pseudo-polynomial time [39]. ■

We conclude that constrained shortest metapaths are good candidates to perform composite decisions, $i.e.$, to optimally decide on a single-link SFC composition in terms of its services placement and chaining with a single binary variable.

Multiple-link chain composition via metapath. While observing Figure 3, we can notice how using only a single constrained shortest metapath per a single-link segment of a multiple-link SFC $a-b-c$ can lead to an unfeasible composition: as the optimal $a-b$ composition is $a-A-Y-b$ metapath, and the optimal $b-c$ composition is $b-B-X-c$ metapath - service b has to be simultaneously placed on Y and B physical nodes. Thus, we cannot *stitch* these metapath, and we need to find more than one constrained shortest metapath per a single-link chain. *In our composite variable approach, we find k-constrained shortest metapaths (to create k binary variables) per each single-link segment of a multi-link service chain.* To find metapaths any constrained shortest path algorithm can be used [37], [38], [39]. In this paper, we build upon the path finder proposed in our prior work that is an order of magnitude faster than recent solutions [37].

To further benefit from constrained shortest metapaths and simplify the chain composition problem, we *offload* its constraints (either fully or partially) to either metalinks or the path finder. Specifically, *geo-location and an arbitrary number of end-to-end network ($e.g.$, latency) QoS constraints can be fully offloaded to metalinks and to the path finder, respectively*. At the same time, capacity constraints of the SFC composition problem are global and can be only partially offloaded. Once k-constrained shortest paths have been found for each single-link service chain segment, we can solve GAP problem [25] to assign each single-link chain segment to exactly one constrained shortest metapath and *stitch* these metapaths as described below.

Allowable fitness functions for metapath-based variables. In general, fitness functions qualify for our *metapath composite variable* approach if they are comprised from either additive or multiplicative terms. The above requirement fits for most SFC objectives [1], and other objectives can also qualify if well-behaved ($e.g.$, if their single-link chain fitness values can be minimized by a path finder). As the load balancing fitness function $F_{\mathbb{E}}$ in Equation 1 qualifies, we compute its single-link chain value $\mathbb{E}\left[F^{sta}_{ijk}\right]$ for k metapath as following:

$$\mathbb{E}\left[F^{sta}_{ijk}\right] = \sum_{\tau\in T}\mathbb{E}\left[\frac{\mathbf{D}^{a\tau}_{s}}{\mathbf{C}^{\tau}_{i}}\right]/deg(s)+ \\ + \sum_{st\in E^a_V}\sum_{\substack{\{uv\in E_S:\\ uv\in P^{sta}_{ijk}\}}}\mathbb{E}\left[\frac{\mathbf{D}^{a}_{st}}{\mathbf{C}_{uv}}\right] + \sum_{\tau\in T}\mathbb{E}\left[\frac{\mathbf{D}^{a\tau}_{t}}{\mathbf{C}^{\tau}_{j}}\right]/deg(t), \tag{9}$$

where $deg(s)$ (or $deg(t)$) corresponds to the service s (or t) degree, $i.e.$, $deg(s) = 1$ (or $deg(t) = 1$) for $s = in$ (or $t = out$) service that handles input (or processed output) data of SFCs; and $deg(s) = 2$ (or $deg(t) = 2$) otherwise. The first and the last terms represent the fitness values of metalinks, and the middle term corresponds to the sum of physical links' fitness values. Thus, this division by $deg(s)$ (or $deg(t)$) is needed to avoid counting the contribution of the intermediate service placement twice in the objective.

Remark: ***in*** *and* ***out*** *are services that handles input and processed output data of SFCs*, respectively.

Problem 2 (SFC composition via metapaths)**.** *Given a set of SFCs $a \in A$ represented as graphs $G^a = (N^a_V, E^a_V)$, a physical network graph $G = (N_S, E_S)$, and having set of k-constrained shortest metapaths $P^{sta}_{ijk} \in \mathcal{P}^{st}_a$ and their corresponding fitness function values F^{sta}_{ijk} found for each virtual link $st \in E^a_V$ in the SFC a, let a binary variable $f^{st}_{ijk}(b,a) = 1$ if the single-link chain segment st is assigned to the metapath P^{sta}_{ijk} of the backup $b \in B^a$ of SFC $a \in A$, or 0 otherwise. The SFC composition problem via metapaths can be formulated as follows:*

$$\min \sum_{a\in A}\sum_{b\in B^a}\sum_{st\in E^a_V}\sum_{P^{sta}_{ijk}\in\mathcal{P}^{st}_a}\mathbb{E}\left[F^{sta}_{ijk}\right] f^{st}_{ijk}(b,a) \tag{10}$$

subject to

Metapath Stitching (Assignment) Constraints:

$$\sum_{P^{sta}_{ijk}\in\mathcal{P}^{st}_a} f^{st}_{ijk}(b,a) - \sum_{P^{tsa}_{jik}\in\mathcal{P}^{ts}_a} f^{ts}_{jik}(b,a) = \begin{cases} -1, & t = in \\ 1, & t = out \\ 0, & otherwise \end{cases} \\ \forall t \in \{in, out\} \vee tj \in E^a_M, b \in B^a, a \in A \tag{11}$$

Node Capacity Chance-Constraints:

$$\mathbb{P}\left[\sum_{a\in A}\sum_{b\in B^a}\sum_{t\in N^a_V}\mathbf{D}^{a\tau}_{t}/deg(t)\left(\sum_{P^{sta}_{ijk}\in\mathcal{P}^{st}_a} f^{st}_{ijk}(b,a) + \right.\right. \\ \left.\left. + \sum_{P^{tsa}_{jik}\in\mathcal{P}^{ts}_a} f^{ts}_{jik}(b,a)\right) \le \mathbf{C}^{\tau}_{j}\right] \ge R, \forall j \in N_S, \tau \in T \tag{12}$$

Link Capacity Chance-Constraints:

$$\mathbb{P}\left[\sum_{a\in A}\sum_{b\in B^a}\sum_{st\in E_V^a}\sum_{\substack{\{P_{ijk}^{sta}\in\mathcal{P}_a^{st}:\\ uv\in P_{ijk}^{sta}\}}}\mathbf{D}_{st}^{a}f_{ijk}^{st}(b,a)\leq\mathbf{C}_{uv}\right]\geq R,\quad\forall uv\in E_S \tag{13}$$

where symbols and notations of sets, parameters, variables and functions are summarized in Table I.

We remark that service placement, chaining, geo-location, latency and no-consolidation constraints from the master Problem 1 have been fully offloaded to the constrained shortest path finder that generates metapaths composite variables. Thus, there is no need to specify them explicitly. At the same time, capacity constraints can be only partially offloaded, and still need to be specified in Problem 2. Note also that deterministic equivalents for the objective coefficients and capacity constraints in Equations 10, 12 and 13 are similar to deterministic equivalents of Problem 1, and their deterministic equivalent examples can be found in [23].

On complexity benefits of metapath composite variables. Both the master integer MCCF Problem 1 and its reduced with composite variables integer (GAP) Problem 2 are NP-hard [17]. However, the number of binary variables in master problem includes the following: the number of binary variables to place services is $\Omega_{N_V}(|A||B||N_V||N_S|)$; and the number of binary variables to chain these services is $\Omega_{E_V}(|A||B||E_V||E_S|)$. The total number of binary variables is $\Omega_{P_1}(|A||B|(|N_V||N_S|+|E_V||E_S|))$. At the same time the total number of composite variables in the reduced problem is $\Omega_{P_2}(|A||B||E_V||K|)$. As we show later in Figure 7, K can be empirically bounded as $|K|\sim|N_S|$. Thus, the composite variable approach at least halves the number of binary variables, $i.e.$, halves the exponent of time (and space) complexities of the Problem 1. This is because for linear topology SFCs $O(|N_V|)=O(|E_V|)$, and for infrastructure topologies $O(|E_V|)\geq O(|N_S|)$. As a result, $\Omega_{P_1}(|A||B|(|N_V||N_S|+|E_V||E_S|))\geq\Omega(2\cdot|A||B||N_V||N_S|)$, and hence, $\Omega_{P_1}\geq 2\cdot\Omega_{P_2}$. We demonstrate these complexity benefits of the proposed composite variable approach in Section VII.

A. SFC Composition via Lagrangian Relaxation

Aside from solving the NP-hard GAP Problem 2, we also propose its better scalable alternative. In particular, we solve the GAP using its polynomial Lagrangian relaxation by compromising both its optimality and feasibility guarantees [25].

Our approach. Problem 2 has two types of constraints - stitching (assignment) and capacity constraints. The assignment constraints (Equation 11) represent flow conservation constraints for metalinks $tj\in E_M^a$. Hence, these constraints form a totally unimodular constraint matrix. When having the linear objective function (Equation 10), this property allows us to relax integrality constraints on $f_{ijk}^{st}(b,a)$ variable in the uncapacitated service chain composition case (when capacity constraints are omitted). As a result, we can solve the above problem using polynomial Linear Programming (LP).

Lower Bound Algorithm. Similarly to [40], we use the unimodularity property benefits and push capacity constraints (see Equations 12 and 13) to the objective. To this end, let us denote $g_1^{\tau j}=R-\mathbb{P}_j^\tau$ and $g_2^{uv}=R-\mathbb{P}_{uv}$ functions for each constraint in Equations 12 and 13, respectively. Let us define $u_1^{\tau j}$ and u_2^{uv} as the Lagrangian multipliers specified for each iteration of the subgradient method [40]; we now can define (deterministic) Lagrangian weights as following:

$$\begin{aligned}w_{ijk}^{sta}=F_{ijk}^{sta}+u_1^{\tau j}\Big((\mu_s^{a\tau}+K_{\frac{R}{P_i}}\sigma_s^{a\tau})/deg(s)+\\ +(\mu_t^{a\tau}+K_{\frac{R}{P_j}}\sigma_t^{a\tau})/deg(t)\Big)+\sum_{uv\in P_{ijk}^{sta}}u_2^{uv}\Big(\mu_{st}^{a}+K_{\frac{R}{P_{uv}}}\sigma_{st}^{a}\Big)\end{aligned} \tag{14}$$

We then can solve the following linear program $\mathcal{L}$ with any LP solver:

$$\begin{aligned}\mathcal{L}=\min\Bigg(\sum_{a\in A}\sum_{b\in B^a}\sum_{st\in E_V^a}\sum_{P_{ijk}^{sta}\in\mathcal{P}_a^{st}}w_{ijk}^{sta}f_{ijk}^{st}(b,a)+\\ -\sum_{j\in N_S}\sum_{\tau\in T}u_1^{\tau j}.\begin{cases}C_j^\tau, R\leq P_j\\ 0, \quad R>P_j\end{cases}-\sum_{uv\in E_s}u_2^{uv}.\begin{cases}C_{uv}, R\leq P_{uv}\\ 0, \quad R>P_{uv}\end{cases}\Bigg)\end{aligned} \tag{15}$$

subject to constraints in Equation 11. Note that to improve LB while solving $\mathcal{L}$, we can also fix all variables $f_{ijk}^{st}(b,a)=0$ whose node (or link) mappings do not satisfy reliability, $i.e.$, if $R>P_i$ or $R>P_j$ (or if $\exists uv\in P_{ijk}^{sta}: R>P_{uv}$).

If solution of $\mathcal{L}$ satisfies GAP capacity constraints, we can stop and report optimal (or suboptimal) solution to GAP. However, if $\mathcal{L}$ solution is unfeasible to the primal GAP problem, we can project it back to the feasible space using some polynomial heuristic algorithm to get an upper bound (UB) of the primal GAP problem.

Upper Bound Algorithm. We propose a new (polynomial) greedy regret lower bound replication (GRLBR) algorithm that we found fast enough for our large scale GAP problem with flow assignment constraints. We build our GRLBR algorithm upon both lower bound replication and greedy regret algorithms proposed earlier in [40], and its pseudo code is outlined in Algorithm 1.

GRLBR starts by detecting the largest regret service chain segment st of SFC a' (lines 5-12), i.e., the segment with the largest difference between the first best and the second best corresponding lagrangian weights w_{ijk}^{sta} for its potential feasible assignments. If there are no feasible metapaths assignments for st of a that satisfy both assignment and capacity constraints (see Equations 11, 12 and 13), we stop and report no feasible solution (lines 6-8). Once, st of a' is found, we add it to the priority queue $Q_{a'}$ based on its langrangian weight $w_{ijk'}^{sta'}$ (line 13). We then retrieve and remove the head of this queue and try to map it to the LB metapath solution first (lines 19-20), or to the lowest lagrangian weight metapath $\hat{Psta'}_{ijk'}$ (lines 22-23), or report no feasible solution and terminate, otherwise (lines 17-18). Finally, we allocate corresponding metapath solution resources for the service chain st segment of a' and add all its adjacent segments (lines 25-26). Once $Q_{a'}$ is empty, all service chain segments of SFC a' for its backup b' have been placed. We then mark b' backup of SFC a' as mapped and remove it from further consideration by GRLBR (lines 28-31). Note that at any time $Q_{a'}$ contains only two elements due to a linear service chain topology.

Subgradient method. Having LB and UB algorithms outlined, we use them within the general subgradient method to

Algorithm 1: GRLBR

Input: $\hat{f}^{st}_{ijk}(a,b)$:= solution of $\mathcal{L}$; w^{sta}_{ijk}:= lagrangian weights; $P^{sta}_{ijk} \in \mathcal{P}^{st}_{a}$:= set of k-constrained shortest metapaths and their corresponding fitness values F^{sta}_{ijk} found for each virtual link $st \in E^{a}_{V}$

Output: UB := upper bound to GAP problem; $f^{st}_{ijk}(a,b)$:= feasible solution to GAP problem

begin
  /* **Step 0: initialize** */
  $A' \leftarrow A$
  $B'^{a} \leftarrow B^{a}, \forall a \in A$
  while $A' \notin \emptyset$ **do**
  /* **Step 1: find highest regret virtual link** sta' */
    forall $st \in E^{a}$ and $a \in A'$ **do**
      if $\nexists P^{sta}_{ijk} : P^{sta}_{ijk}$ is feasible **then**
        terminate and report no feasible solution
      end
      $ijk'_{sta} \leftarrow arg\min\{w^{sta}_{ijk} : P^{sta}_{ijk}$ is feasible$\}$
      $\rho_{sta} \leftarrow \min\{w^{sta}_{ijk} - w^{sta}_{ijk'(sta)} :$ P^{sta}_{ijk} is feasible, $ijk \neq ijk'_{sta}\}$
    end
    $sta' \leftarrow \underset{st \in E^{a}, a \in A'}{arg\max} \{\rho_{sta}\}$
  /* **Step 2: allocate all service chain segments that contains** sta' */
    Put sta' to the priority queue $Q_{a'} \leftarrow \{sta', w^{sta'}_{ijk'}\}$
    $b' \leftarrow \min\{B'^{a'}\}$
    while $Q_{a'} \notin \emptyset$ **do**
      $\hat{st} \leftarrow$ retrieve and remove $Q_{a'}$'s head
      if $\nexists \hat{P}^{\hat{sta}'}_{ijk} : P^{\hat{sta}'}_{ijk}$ is feasible **then**
        terminate and report no feasible solution
      else if $P^{\hat{sta}'}_{ijk} : \hat{f}^{\hat{st}}_{ijk}(a',b') == 1$ is feasible **then**
        $UB \leftarrow UB + F^{\hat{sta}'}_{ijk}$
      else
        $\hat{Psta}'_{ijk'} \leftarrow arg\min\{w^{\hat{sta}'}_{ijk} : P^{\hat{sta}'}_{ijk}$ is feasible$\}$
        $UB \leftarrow UB + F^{\hat{sta}'}_{ijk'}$
      end
      allocate corresponding physical resources for $\hat{st}$
      add adjacent virtual links of $\hat{st}$ and their best lagrangian weights to $Q_{a'}$
    end
  /* **Step 3: mark** b' **backup of** a' **SFC as allocated and go to Step 1** */
    $B'^{a'} \leftarrow B^{a'} - b'$
    if $B'^{a'} \in \emptyset$ **then**
      $A' \leftarrow A' - a'$
    end
  end
end

iteratively improve LB and UB as in [40]. To this end we start with zero u_1 and u_2 lagrangian multiplier vectors. At each iteration we track if LB solution is feasible, and if so we terminate our subgradient algorithm. Moreover, if LB has been improved, $i.e.$, if $LB_{new} > LB$, and LB is not feasible, we project LB solution back to the feasible space with our GRLBR algorithm to obtain new UB_{new} solution and update existing UB solution if $UB_{new} < UB$. If $\frac{||UB-LB||}{||LB||} < \epsilon$ or number of iterations is exceeded, we terminate the subgradient algorithm. At the end of each iteration u_1 and u_2 are calculated w.r.t. to their objective gradient. More implementation details as well as best practices on the subgradient method can be found in [40].

V. Service Chain Maintenance via Metapaths

Any network operating in a challenged scenario is subject to instabilities. Consider $e.g.$, a physical network after a natural disaster. In this section, we present a novel *metapath-based SFC maintenance* algorithm that utilizes a distributed control plane to cope with such network instabilities by allowing migration $i.e.$, reallocation of (part of) the service chain to

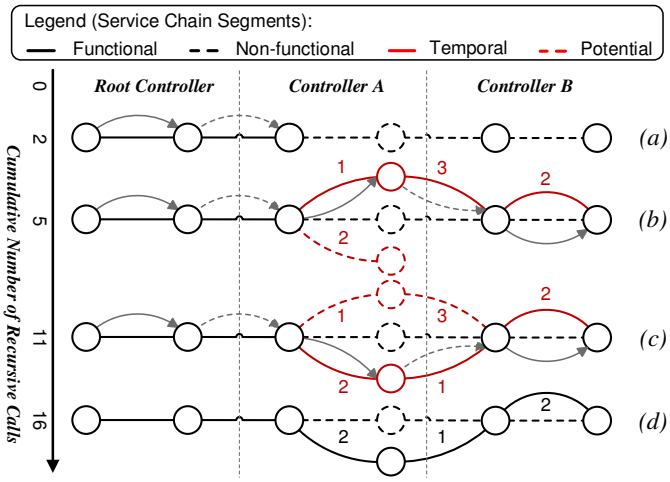


Fig. 4: Illustrative example of our distributed metapath-based SFC maintenance algorithm which provisions permanently the best new mappings of the failed SFC segments with the total fitness function of 5 once controllers A and B check all potential mappings.

maintain its services. We first outline our algorithm, and then prove its eventual correctness property to qualify for the SCL use that avoids expensive consensus protocols [19].

Distributed metapath-based SFC maintenance. Firstly, we assume that all demand increase requests of SFCs which QoS demands are not sufficient are handled proactively by the *root controller* – a controller associated with a physical node of the chain root service. By convention the root service is a service that outputs SFC processed data. We now present our metapath-based SFC maintenance algorithm for a distributed control plane whose logic is outlined in Algorithm 2, and Figure 4 illustrates its work. Upon a non-functional SFC segment detection (Figure 4a), the algorithm starts from the *root* controller and then, recursively, checks all service chain segments between the corresponding controllers to find all non-functional assignments (Step 1). In Step 2, the algorithm generates k-constrained shortest metapaths or finds them among the list of pre-computed, $e.g.$, during the service chain composition. When all service chain segments belonging to some controller are checked or temporally restored (Figure 4b), this controller requests the next chain segment (Steps 3 and 4; Figure 4c). Once the termination criteria is met (Figure 4d), the best found mappings (w.r.t. the fitness function) or a failed SFC error message are returned (Step 5).

The termination criteria is met when all combinations of possible failed segment allocations are checked or some heuristic number is reached, $e.g.$, the number of maximum recursive calls, used metapaths, etc.

To generate metapaths, we have two options - use metapaths which have been *pre-computed* during the service chain composition step, or find such metapaths *dynamically*. We evaluate different policy trade-offs in Section VII-A2.

On Algorithm 2 space/time complexity. Both time and space complexities of the maintenance algorithm for the particular SFC can be bounded as $O(k^{E_V})$. This is because Algorithm 2 uses k constrained-shortest metapaths at each recursive call with the maximum depth of the number of SFC segments E_V. To avoid potential exponential complexity of the Algorithm 2, we suggest to use a simple decay function for k policy, $i.e.$,

Algorithm 2: Metapath-based Service Chain Maintenance

Step 1: Upon receiving a message from controller l, start from the service s at the controller j and verify the next service chain link segment $st \notin$ checked segments
- *If* st is non-functional, stop and go to Step 2
- *Else if* $t \in j$, make $s = t$ and check next segment
- *Else*, send checked service chain segments to the controller $i : t \in i$

Step 2: If failed st found, generate k-constrained shortest metapaths set K for st (sorted in ascending order by their fitness function values) *dynamically* (to be discussed later) or among pre-computed metapaths during the composition step

Step 3: Iterate while $K \notin \emptyset$
- Retrieve and remove k metapath from K, then temporally allocate st on k and add it to checked segments
- Check current fitness function value:
  - *If* current fitness function values is worse then the best known objective value, skip this step
  - *Else if* $t \in j$, make $s = t$ and resume from the step 1
  - *Else*, send checked service chain segments to the controller $i : t \in i$
- If all segments have been checked and current fitness function value is the best known, track best physical resource mappings for non-functional segments
- Release st resources and remove it from checked segments

Step 4: Reply back to the controller l

Step 5: Once a termination criteria is met, permanently provision best known physical resources for non-functional service chain segments.

$k = round(k^{\frac{1}{d}})$, where d is a depth of a service chain segment from the root. Thus, if $k = 10$, we use up to 10 metapaths for the first segment, up to 3 for the second one, etc.

On Algorithm 2 'eventual correctness'. Based on the recent results in [19] it is possible to guarantee a distributed control plane consistency without expensive consensus protocols by introducing a simple layer for its coordination. However, in order to qualify for this layer use, control mechanisms have to have *eventual correctness* guarantees. We formally prove such guarantees of our Algorithm 2 below.

Lemma V.1. (The metapath maintenance eventual correctness) *Our metapath-based SFC maintenance outlined in Algorithm 1 has eventual correctness guarantees.*

Proof. To have eventual correctness guarantees, by definition control mechanisms needs to be *deterministic*, have *idempotent behavior*, *triggered recomputation* and be *proactive* w.r.t., to data plane [19]. We remark that once SFCs are functional, their demands become certain during some time slot τ. Hence, our metapath-based SFC maintenance outlined in Algorithm 1 is *deterministic* and features the *idempotent behavior*, *e.g.*, once a flow rule for some SFC link is setup by one controller this rule remains the same if setup by another one. Moreover, our Algorithm 2 needs to be *triggered* (*e.g.*, by a *root controller*) for a service chain elements migration which is done *proactively* by setting up new flow rules and provisioning new virtual machines to host service functions. ■

We conclude, that Algorithm 2 qualifies for the SCL use. In the rest of this paper, we first outline our reliable SFC orchestration prototype implementation. We then evaluate performance of our approach using both trace-driven simulations as well as experiments on a realistic core/edge cloud testbed.

VI. Service Chain Orchestration Prototype

In this section we describe the architecture of our *reliable service chain orchestration* prototype shown in Figure 5. Our prototype architecture includes four main logical components: (i) *control applications*, used to compose and maintain SFCs;

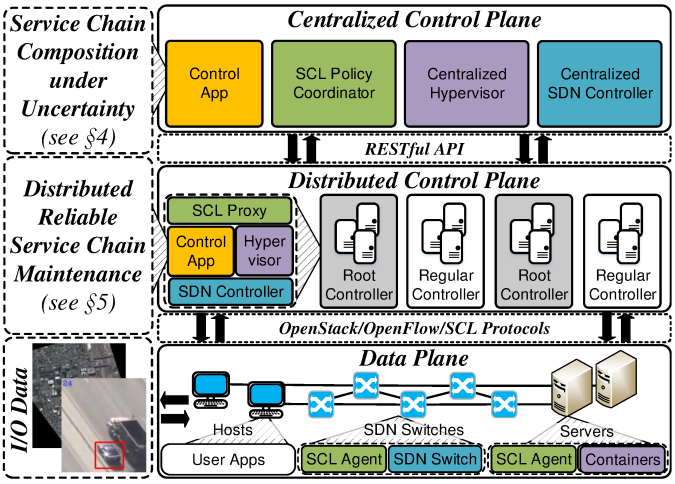


Fig. 5: System architecture of our reliable SFC orchestration prototype includes four main logical components: (i) *control application* is responsible for service chain composition in centralized control plane and its maintenance in distributed control plane; (ii) the Simple Coordination Layer (SCL) and *root controllers* are responsible for guaranteeing consistency in the distributed control plane; (iii) *SDN* is responsible for traffic steering in data plane; and (iv) *Hypervisor* is responsible for placing service functions. The prototype source code is publicly available at [27].

(ii) the SCL and *root controllers*, used to guarantee consistency of the distributed control plane; (iii) A *SDN*-based system with (iv) a *Hypervisor* to allocate mapped physical resources. Our source code is publicly available at [27]. In the rest of the section, we describe with some more details each of the four components of our prototype.

Control applications. We have two main types of control applications. The first type is responsible for the reliable service chain composition in the *centralized control plane* as discussed in Section IV. The second type is responsible for maintenance of composed service chains as discussed in Section V. We remark that to avoid both a single point of failure as well as congestion in the centralized control plane, we maintain SFCs in the *distributed control plane*.

SCL and root controllers. To guarantee consistency in the distributed control plane and avoid various related violations (*e.g.*, looping paths, QoS violations, etc.), our control applications qualify to use the SCL [19]. SCL includes three main components: *SCL Agent* running on physical resources, *SCL Proxy Controller* running on controllers in the distributed control plane, and *SCL Policy Coordinator* running in the centralized control plane. The agent periodically exchanges messages with corresponding proxy controllers and triggers any changes in the physical resources. Proxy controllers send information about data plane changes to the service chain maintenance control application and periodically talk with other SCL Proxy Controllers. Finally, all policy changes (*e.g.*, in control applications, in SCL, etc.) are committed via 2-phase commit [19] by the policy coordinator.

In order to handle all service chain modification requests from application owners such as demand or latency sensitivity changes, we use *root controllers* - controllers that leverage physical resources associated with root services of service chains. In this paper, by root services we mean services that provide processed data to end-users.

SDN and Hypervisor. The last two logical components of our prototype are well-known *SDN* and *Hypervisor* systems.

Guided by control applications, both SDN and hypervisor are responsible for traffic steering and containers provisioning in the data plane, respectively. We use OpenFlow as our main SDN system [41] and Docker containers [42] as our hypervisor system to place services on the physical server.

VII. Performance Evaluation

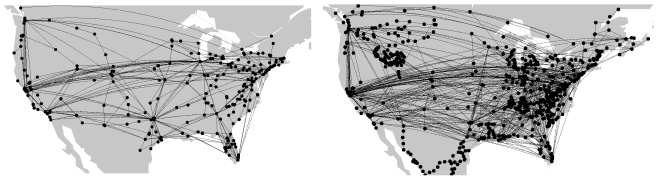
In this section, we evaluate performance of our reliable SFC orchestration approach under challenging disaster incident conditions that can cause severe infrastructure outages [8]. Thus, we first evaluate its performance against the state-of-the-art NFV/VNE solutions of the (master) integer MCCF problem. We then evaluate its maintenance performance w.r.t., the only existing to our knowledge consensus-based SFC orchestration scheme [21]. Finally, we evaluate its prototype implementation and benefits for our case study of object tracking using geo-distributed latency-sensitive SFCs w.r.t. the common core cloud computing.

Results. Our evaluation results can be summarized with the following thrusts: (i) *our metapath approach yields more than 99% optimality on average and is up to 3 orders of magnitude faster than the master problem solution*; (ii) *our metapath approach can secure up to two times more SFCs in comparison to the state-of-the-art NFV/VNE approaches under challenging disaster incident conditions*; (iii) *policies allow to trade-off between a SFC reliability and its composition optimality*; (iv) *our metapath approach enables better SFC maintenance for lesser control messages*; and (v) *geo-distributed latency-sensitive SFCs can improve real-time data processing*.

A. Numerical Evaluation of our Reliable SFC Orchestration

General Settings. For our simulations, we use an *HPC Cloud server* with two *Intel Xeon E5-2683 v3* 14-core CPUs at 2.00 GHz (total 56 virtual cores), 256GB RAM, and running the Ubuntu 16.04 allocated in NSF CloudLab platform [45]. We solve math programs with IBM ILOG CPLEX [35]. We use both Internet Topology Zoo [43] and Atlas [44] databases to re-create the US Tier1 and regional providers' networks as shown in Figure 6. We assume that each topology has nodes and links with uniformly distributed computation capacity from 5 to 50 TFlops and bandwidth from 1 to 10 Gbps, respectively. Note that the lowerbound 5 TFlops performance can simulate limited network edge servers, whereas 50 TFlops can simulate HPC cloud servers. Moreover, we assume that latency of each physical link is proportional to its propagation delay computed as its geographical length divided by the speed of light in fiber. Finally, we compute physical resources' outage risk w.r.t. to the geographical proximity to the disaster incident epicenter as discussed in [8]. All our results show 95% confidence intervals, and our randomness lays both in SFC requests and in disaster incident events.

SFC request settings. We generate a pool of 50 SFCs composed by 2 to 20 services, unless stated differently. Based on a common object tracking application [4], each SFC has equal chances to express its either High-Performance (HPC) or regular computing demands shown in Table II. As in [8], we assume a strong correlation between SFC demands and the disaster incident intensity. Using natural disaster data sets and their associated infrastructure outage risks specified in [8], we use the following *geo-location policies*: All services handling incoming raw data must be placed within a range of two disaster incident region radiuses from its epicenter. When there are no disaster incidents, these services as well as services that output processing data have to be placed within 200 miles out of the random geographic locations picked within the US. We remark that this geo-location policy is intended to simulate incident-supporting applications [4] that can benefit from use of geo-distributed latency-sensitive SFCs.



(a) (b)

Fig. 6: Simulation data sets: (a) network infrastructure that spans 7 Tier-1 US providers and comprises of 286 Point of Presence (PoP) nodes and 534 links; and (b) network infrastructure that spans 56 regional US providers and comprises of 596 PoP nodes and 1253 links.

TABLE II: An example set of demands for different SFC types

Demands Type	Expected Computation Demand per Function	Expected Data Collection Size	Expected Comp. Time	Expected Data Rate
Regular	0.5-5 TFlops	1-10 GB	10-100 ms/frame	10-100 Mbps
HPC	1-10 TFlops	10-100 GB	10-100 ms/frame	0.1 - 1 Gbps

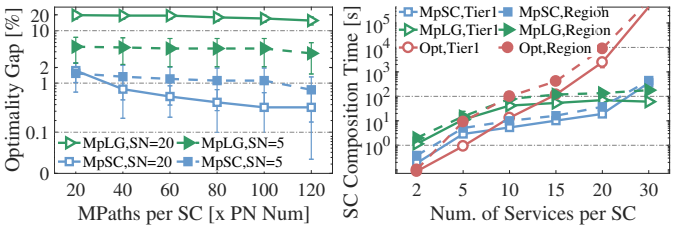
1) Service Chain Composition Evaluation:

Composition Metrics. We compare performance of our metapath-based SFC composition in Problem 2 (referred as $MpSC$) against its (polynomial) Lagrangian relaxation counterpart (referred as $MpLG$). We also compare $MpSC$ against the VNE/NFV state-of-the-art solutions of the (master) integer MCCF problem ($i.e.$, Problem 1): IBM CPLEX branch-and-bound version [35] (optimal, but has the highest combinatorial complexity), branch-and-price column generation [26] and recent isomorphism detection [22] approaches (suboptimal, but have lower combinatorial complexities). We refer to the branch-and-bound solution of the master problem as Opt, to the column (or path in case of SFCs) generation approach as $PgSC$, and to the isomorphism detection as $IsoSC$.

Assumptions. To evaluate the *online* optimization performance of our approach, we assume the most difficult case: all SFC requests arrive sequentially ($i.e.$, unknown in advance) and do not allow a service consolidation, $i.e.$, only one service in a chain can be placed onto the same physical server. Thus, we are forced to compose fractional traffic flows between all services in order to maximize the number of composition decisions that need to be made during SFC.

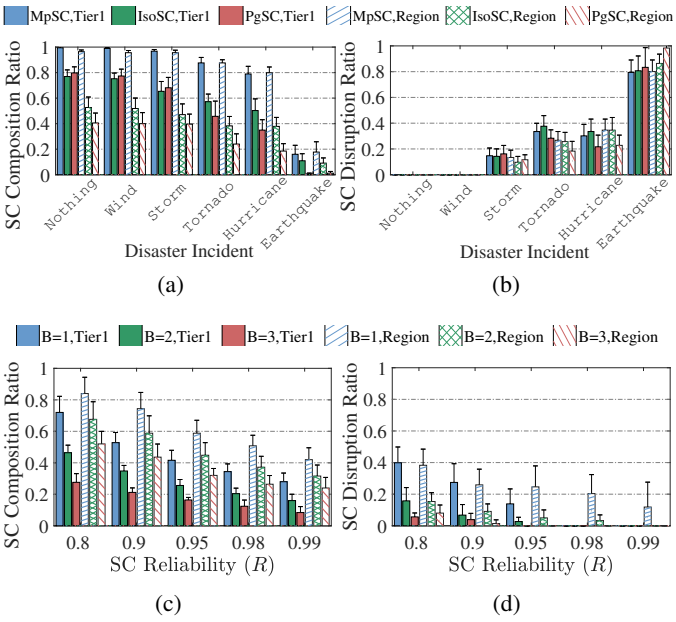
We assess the performance of our SFC composition algorithms by specifying the fraction of successfully composed SFCs over the total requested chains (*composition ratio*). Similarly, we assess the reliability of these algorithms by computing a fraction of the number of composed SFCs disrupted during disaster incidents over the total number of composed SFCs (*disruption ratio*). In addition, we also use an *optimality gap* metric which we define as a gap in % between Opt and all other algorithms. Finally, we use a *composition time* metric to access *scalability* performance of our metapath approach.

(i) MpSC gains more than 99% optimality on average and is up to 3 orders of magnitude faster than Opt. To



(a) (b)

Fig. 7: Service chain (SC) composition optimality gap (a) and time (b) results in presence of no disaster incidents ($R = 0$).



(a) (b) (c) (d)

Fig. 8: Service chain (SC) composition ratio (a,c) and disruption ratio (b,d) results under different natural disaster-incidents with reliability $R = 0.8$ (*first row*), and $MpSC$ results under hurricane disaster-incidents (*second row*).

estimate the baseline performance of our metapath approach, we first assume a scenario without disasters ($i.e.$, no outage risks and $R = 0$) where capacity chance constraints becomes deterministic yielding the largest feasible space ($i.e.$, leading to a higher combinatorial complexity of the master problem). Figure 7a shows how its optimality depends on the number of generated metapaths per single service chain (SC) request. We can see how, when this number exceeds $\geq$80 times the number of physical nodes (PNs), the performance of $MpSC$ flattens. On the other hand, when either the SFC service number (SN) increases or the reliability requirements get tighter in a disaster incident scene, $MpSC$ achieves optimality most of the time and shows 99% optimality on average. Note also how $MpLG$ shows significantly worse performance with respect to $MpSC$; this is due to use of greedy heuristics used to recover feasible solutions. However, $MpLG$ can be beneficial for large SFCs (of $\geq$ 25 services) as it is polynomial.

For the rest of our evaluation, we fix the number of metapaths generated per SFC request to 80 and 120 times the number of physical nodes for $MpSC$ and $MpLG$, respectively. These values of metapaths are picked to allow both $MpSC$ and $MpLG$ to compose large SFCs. For instance, with these settings $MpSC$ is almost three orders of magnitude faster than the optimal solution (Opt) as shown in Figure 7b. For small SFCs ($i.e.$, $\leq$ 5 services) we found, however, no significant scalability improvements of $MpSC$ and $MpLG$ over Opt. This is due to the fact that generating metapaths is time-consuming. Thus, for small service chains, it is recommended to avoid use of metapath-based composite variables and merely consider the Opt policy instead. For the rest of our evaluation, we only use the $MpSC$ service chain composition algorithm.

(ii) MpSC can secure up to 2 times more SFCs than PathGen and IsoSC under challenging disaster-incident conditions. Furthermore, we can see how our $MpSC$ outperforms $PgSC$ and $IsoSC$ by securing up to 2 times more SFCs under challenging disaster-incident conditions of tornadoes and hurricanes as shown in Figures 8a and 8b with the service chain reliability $R = 0.8$. This is due to the fact that $MpSC$ reaches the optimality most of the time while being sufficiently scalable. At the same time, $PgSC$ is limited by the performance of the SFC composition algorithm (that commonly uses a two-stage composition) to get the initial feasible solution [26]. Moreover, it is also known that column generation approaches such as $PgSC$ converge slowly to the optimal for integer problems [25]. In contrast to $PgSC$, $IsoSC$ doesn't need an initial feasible solution, but can fail to find one or not converge to the optimal solution for the predefined amount of iterations [22].

(iii) Policy-based SFC reliability trade-offs. Further, to achieve a desired level of reliability during SFC composition ($i.e.$, proactively), the capacity chance-constraints acceptable risk ($i.e.$, $1 - R$) and/or the number of backups policies can be adjusted appropriately. As shown for $MpSC$ in Figures 8c and 8d, increasing either chance-constraints reliability R or the number of backups decreases the number of composed SFCs by either prohibiting more physical resources for allocation or utilizing more physical resources for SFC backups. On the other hand, such a strategy can significantly minimize the number of disrupted SFCs, therefore minimizing their outages.

2) Service Chain Maintenance Evaluation:

Maintenance Metrics. We compare our metapath-based SFC maintenance algorithm referred as $MpSM$ with the only existing (to the best of our knowledge) consensus-based SFC orchestration approach that can guarantee distributed control plane consistency – $Catena$ [21].

In this simulation scenario we have mainly assessed performance of SFC composition algorithm by specifying the fraction of times service chain events are successfully migrated over their total appearance number (*blocking ratio*). In addition, we also use an *optimality gap* and a *number of control messages* metrics to access the optimality and a complexity performance of our proposed solution.

(iv) Metapaths and SCL for better SFC migrations with lesser control messages. Figures 9a, 9b and 9c show how $MpSM$ with the pre-computed metapaths P policy can more optimally migrate SFCs with a lower blocking probability than $Catena$ and using an order of magnitude less control messages. The reason for these results is two fold. First of all, our algorithm simultaneously considers fitness functions of service placements and their chaining by recursively traversing possible migrations w.r.t. k-constrained shortest metapaths policy, whereas $Catena$ has approximation guarantees only for the service placement and uses k-shortest physical paths (not metapaths) to chain them in a best-effort manner. Secondly,

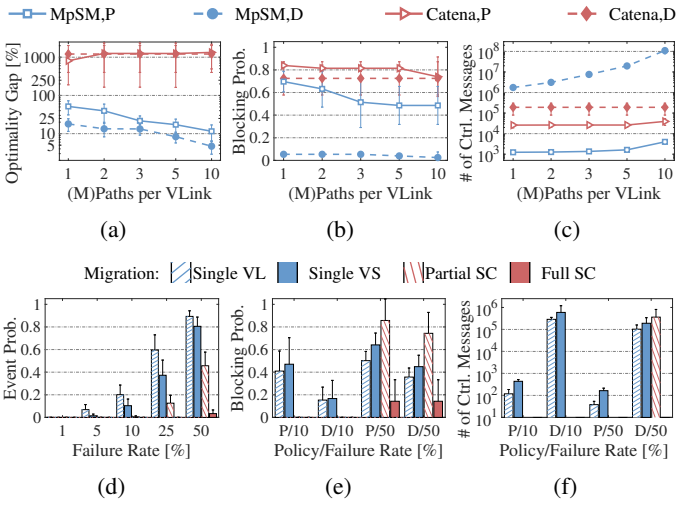


Fig. 9: Service chain (SC) migration optimality gap (a), blocking probability (b) and number of control messages used for this migration (c) results. Single virtual link (VL), single virtual function service (VS) and partial or full service chain (SC) event migrations probabilities (d), their blocking probabilities (e) and number of control messages used to migrate them (f) results with $k = 5$ metapath policy and different physical network failure rates.

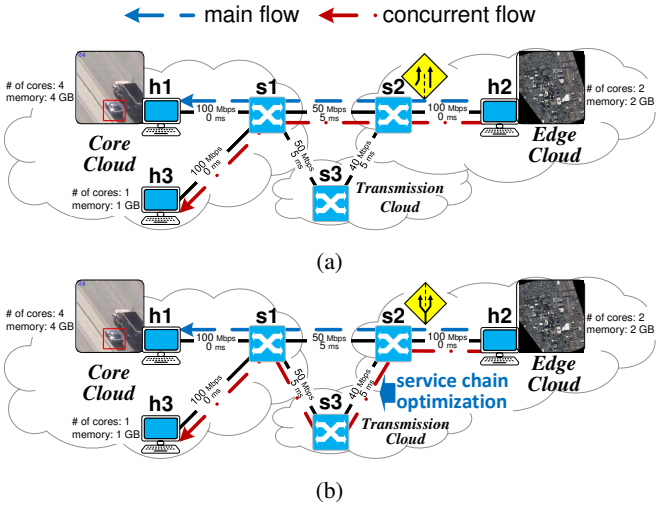


Fig. 10: Data flows in the allocated in GENI edge/core cloud testbed: (a) object tracking data flow interferes with concurrent flow on the $s2 \rightarrow s1$ link as regular network sends data through the best (the shortest) path; (b) by using our reliable service chain orchestration prototype, we chain tracking services with QoS guarantees which avoids congestion by redirection of concurrent flow through longer path $s2 \rightarrow s3 \rightarrow s1$. Furthermore, by allocating image pre-processing service function at the edge cloud (to $h2$ instead $h1$) it enables near *real-time* tracking.

our $MpSM$ uses SCL to avoid expensive consensus control messages that $Catena$ uses for providing the same distributed control plane consistency guarantees. We can also see how our $MpSM$ reaches optimality of $\geq$90% when number of traversed metapath candidates $k \geq 5$ for both dynamic D and pre-computed P metapath policies. Thus, for the rest of our simulation we fix $k = 5$.

Figures 9e and 9f illustrate how $MpSM$ can significantly reduce the blocking probability of various SFC migration events when finding metapaths dynamically D. However, the main side effect is that $MpSM$ with D policy demonstrates ≈ 4 orders of magnitude increase in control messages w.r.t. its P policy. Thus, we recommend use D policy if the following criteria are met: (i) single link or service migration events happen; (ii) the physical infrastructure experience severe failures, $i.e.$, $\geq 25\%$; and (iii) number of controllers is at least an order of magnitude less than the number of physical resources. Each of these criteria can decrease the number of control messages approximately by an order of magnitude. Particularly, (i) is due to the fact that single SFC segments are easier to recover, and these events are more common (see Figure 9d); (ii) is due to the fact that having more failed physical resources needs less number of control messages and significantly reduces a feasible space for SC migrations; and (iii) is due to the fact that the more physical resources are controlled by a single controller the more recursive calls of $MpSM$ can be done in memory, thus saving on control messages. As a result, number of control messages of $MpSM$ with D policy can be reduced up to 3 orders of magnitude w.r.t. P policy and approximately equal to $Catena$ with P policy.

B. Experimental Evaluation of our Object Tracking Case Study Geo-Distributed Latency-Sensitive Service Chain

In this subsection, we discuss our edge/core cloud testbed setup in GENI [28] and show improvements in data throughput and tracking time for our case study of object tracking using geo-distributed latency-sensitive SFCs w.r.t. the common core cloud computing. To this aim, we use our reliable service chain orchestration prototype implementation outlined in Section VI.

Setup. Multiple video resolutions in practice need to be processed because the input source imagery in surveillance typically spans a wide variety of sensor technologies found in mobile devices. In our experiments, we choose the VGA resolution data to express regular SFC demands (see Table II) and track pedestrians in a crowd [46]. The pre-processing step is performed for every image that needs to undergo adaptive contrast enhancement with Imagemagick tool before being used for tracking in the core cloud. The adaptive contrast enhancement requires global image information and thus needs to read in image data into memory, and operates on every pixel. All images are pyramidal tiled TIFF (Tagged Image File Format) and the pre-processing retains the tile geometry.

Our edge/core cloud testbed setup includes 6 virtual machines (VMs) in the GENI platform as shown in Figure 10, where three of these VMs emulate OpenFlow switches ($s1$, $s2$ and $s3$) and others are regular hosts ($h1$, $h2$ and $h3$). To consider disaster network scenarios that impact data transfer, we assume a 4G-LTE network configuration at the edge. Hence, each *host-to-switch* link has 100 Mbps bandwidth without delay, each *switch-to-switch* link has only 40 or 50 Mbps bandwidth (unless specified differently) and 5 ms transmission delay to emulate congested and damaged network infrastructure in a disaster scenario. Using our reliable SFC orchestration prototype, the tracking service function is allocated on $h1$ (quad-core CPU and 4GB of RAM) that acts at a *core cloud* site. At the same time, the pre-processing service function is allocated on $h2$ (double-core CPU and 2GB of RAM) that acts at a *edge cloud* site. Finally, $h3$ (single-core CPU and 1GB of RAM) consumes raw data from $h2$ by acting as a remote storage at *core cloud* site. The $h3$ is configured with cross-traffic flows consumption such that it interferes with our object tracking traffic. We call cross and object tracking traffic as the 'concurrent' and 'main' flows, respectively.

TABLE III: Object tracking case study results over limited network conditions.

Performance Metrics	Object Tracking in the Cloud	Object Tracking via SFC	Perceived Benefits
Preprocess time (s/fr)	0.1955 ± 0.0011	0.202 ± 0.023	**No significant difference**
App thr-put (Mbps)	10.50 ± 0.34	41.85 ± 0.24	Avoiding congestion with our reliable SFC orchestration maximizes object tracking throughput
Tracking time (s/fr)	0.4097 ± 0.0022	0.4229 ± 0.0024	No significant difference
Waiting time (s/fr)	0.902 ± 0.032	$\mathbf{0 \pm 0}$	Achieving maximum speed of tracking service function avoids waiting time and supports real-time computation
Total time (s/fr)	1.312 ± 0.034	0.4229 ± 0.0024	Real-time data processing via geo/latency-sensitive SFCs can produce $\sim$3.5X speedup over common cloud computing scenario

To differentiate between object tracking via the allocated geo-distributed latency-sensitive SFC and the cloud computing scenario, our experiment workflow is as follows: (i) start sending concurrent traffic from $h2$ to $h3$; (ii) start sending main traffic (imagery) from $h2$ to $h1$; (ii.a) while performing data processing via the allocated SFC, start pre-processing concurrently with step (ii); (iii) wait until at least the first frame has been transferred; (iii.b) in case of core cloud computing, start pre-processing before step (iv) (in this case the tracking service function has to wait for each frame when its pre-processing ends); (iv) start tracking; (v) wait until all main traffic has been transferred; and (vi) terminate both the service functions and data transfers.

(v.a) Geo-Distributed Latency-Sensitive SFCs can improve Real-Time Data Processing. Table III shows the final timing results computed with estimate 95% confidence intervals for SFC and cloud computing cases over limited edge network with 50 Mbps access bandwidth. For each trial, we used a 300 frame video sequence and measured several application performance metrics such as estimated throughout, tracking time, waiting time and total time. In our settings, we are able to pre-process frames faster in the core cloud computing scenario than when a geo-distributed latency-sensitive SFC is used. However, due to congestion in best-effort IP network and the absence of raw data at the *core cloud* site, we cannot track frames in real-time (i.e., with 0 waiting time) in the core cloud computing scenario. Whereas by composing a geo-distributed latency-sensitive SFC of our object tracking pipeline, we can track frames in near real time at $3-4$ Hz.

TABLE IV: Object tracking case study results over degrading network conditions.

Performance Metrics	Object Tracking in the Cloud	Object Tracking via SFC
Number of tracked frames	224 ± 7	892 ± 9
App thr-put (Mbps)	11.02 ± 0.36	43.9 ± 0.4
Tracking time (s/fr)	0.3904 ± 0.021	0.403 ± 0.004
Waiting time (s/fr)	0.86 ± 0.28	0 ± 0
Total time (s/fr)	1.25 ± 0.29	0.403 ± 0.004

(v.b) Geo-Distributed Latency-Sensitive SFCs can mitigate network QoS degradation. Further, to consider disaster network scenarios that can impact data transfer, we apply a bandwidth degradation profile to the (collection) edge network with an initial bandwidth of 100 Mbps (best case). For experimental purposes, the profile degrades the bandwidth at a rate of 20 Mbps per minute due to heavy cross-traffic load or candidate network path failures till it falls to zero (i.e., worst case disconnection scenario) with the total 5 minutes of tracking time until the edge network gets disconnected from the collection site. Table IV shows how using SFCs is beneficial in terms of the total number of tracked frames, $i.e.$, almost 4x more frames can be tracked than in the similar case of using tracking in the cloud. This is due to the benefit of low latency data access by pre-processing functions that can fetch required data faster from the collection site without been bottlenecked by the main tracking pipeline in the cloud.

VIII. Conclusion

In this paper, we presented the *reliable orchestration* approach to augment our previously proposed *composition* approach with the *maintenance* approach to not only compose but also support geo-distributed latency-sensitive SFCs during their lifespan [23]. To ensure reliability of SFCs, we handle both their demand fluctuations and possible infrastructure outages during the composition via use of capacity chance-constraints and service backups policies. We have addressed NP-hardness limitations of the (master) integer MCCF-based SFC composition problem by proposing a novel *metapath composite variable approach* that uses either (NP-hard) GAP or its (polynomial) Lagrangian relaxation counterpart. Using realistic trace-driven simulations with US Tier-1 and regional infrastructure topologies, we have shown that our metapath composite variable approach reaches 99% optimality on average, is up to 3 orders of magnitude faster than the master problem solution for practically sized problems and can compose twice as many SFCs than related NFV/VNE methods. Moreover, we have also proposed a novel metapath-based SFC maintenance algorithm that guarantees consistency of the distributed control plane without use of expensive consensus protocols. To this aim, we have proved its eventual correctness to utilize a prior Simple Coordination Layer concept [19]. As a result, our metapath-based maintenance solution reaches better optimality for less number of control messages than a recent consensus-based SFC orchestration scheme [21]. Finally, we were able to show almost a 3.5x speedup of our object tracking case study application when a geo-distributed latency-sensitive SFC is used for data processing with respect to a common cloud computing-based solution. To this aim, we deployed our reliable service chain orchestration prototype on a realistic edge/core cloud testbed allocated in GENI. The source code of our prototype is publicly available at [27].

Acknowledgements

This work has been supported by NSF awards CNS-1647084, CNS-1647182 and Coulter Foundation Translational Partnership Program. The information reported here does not reflect the views of the funding agencies.

References

[1] D. Bhamare, et al., "A survey on service function chaining", *Elsevier JNCA*, 2016.
[2] Y. Niu, et al., "Load balancing across microservices", *Proc. of IEEE INFOCOM*, 2018.
[3] R. Yu, et al., "Application provisioning in fog computing-enabled IoTs: a network perspective", *Proc. of IEEE INFOCOM*, 2018.
[4] R. Gargees, et al., "Incident-Supporting Visual Cloud Computing Utilizing Software-Defined Networking", *IEEE TCSVT*, 2017.
[5] R. Cziva, et al., "Dynamic, latency-optimal VNF placement at the network edge", *Proc. of IEEE INFOCOM*, 2018.

[6] Q. Zhang, et al., "Adaptive interference-aware VNF placement for service-customized 5G network slices", *Proc. of IEEE INFOCOM*, 2019.
[7] R. Potharaju et al., "Demystifying the dark side of the middle: a field study of middlebox failures in datacenters," Proc. of ACM IMC, 2013.
[8] B. Eriksson, et al., "Riskroute: a framework for mitigating network outage threats", *Proc. of ACM CoNEXT*, 2013.
[9] X. Fei, et al., "Adaptive VNF scaling and flow routing with proactive demand prediction", *Proc. of IEEE INFOCOM*, 2018.
[10] C. Zeng, et al., "Demystifying the performance interference of co-located virtual network functions", *Proc. of IEEE INFOCOM*, 2018.
[11] X. Li, et al., "DHL: Enabling flexible software network functions with FPGA acceleration", *Proc. of IEEE ICDCS*, 2018.
[12] A. Tomassilli, et al., "Provably efficient algorithms for placement of SFCs with ordering constraints," *Proc. of IEEE INFOCOM*, 2018.
[13] Q. Zhang, et al., "Joint optimization of chain placement and request scheduling for network function virtualization", *Proc. of IEEE ICDCS*, 2017.
[14] R. Cohen, et al., "Near optimal placement of virtual network functions," *Proc. of IEEE INFOCOM*, 2015.
[15] Y. Sang, et al., "Provably efficient algorithms for joint placement and allocation of virtual network functions," *Proc. of IEEE INFOCOM*, 2017.
[16] S. Jain, et al., "B4: Experience with a globally-deployed software defined WAN," *Proc. of ACM SIGCOMM CCR*, 2013.
[17] H. Feng, et al., "Approximation algorithms for the NFV service distribution problem", *Proc. of IEEE INFOCOM*, 2017.
[18] P. Berde, et al., "ONOS: Towards an Open, Distributed SDN OS", *Proc. of HotSDN*, 2014.
[19] A. Panda, et al., "SCL: Simplifying Distributed SDN Control Planes", *Proc. of NSDI*, 2017.
[20] D. Ongaro, J. K. Ousterhout, "In Search of an Understandable Consensus Algorithm", *Proc. of USENIX*, 2015.
[21] F. Esposito, "Catena: A distributed architecture for robust service function chain instantiation with guarantees", *Proc. of NetSoft*, 2017.
[22] B. Spinnewyn, et al., "Resilient application placement for geo-distributed cloud networks", *Elsevier JNCA*, 2017.
[23] D. Chemodanov, et. al., "A Near Optimal Reliable Composition Approach for Geo-Distributed Latency Sensitive Service Chains", *Proc. of IEEE INFOCOM*, 2019.
[24] C. Barnhart, "Airline fleet assignment with enhanced revenue modeling," *INFORMS Operations Research*, 2009.
[25] M. L. Fisher, "The Lagrangian relaxation method for solving integer programming problems", *Management science*, 2004.
[26] R. Mijumbi, et al., "A path generation approach to embedding of virtual networks", *IEEE TNSM*, 2015.
[27] Metapath-based service chain orchestration repository - https://bitbucket.org/duman190/mpsc_orchestration; Last accessed Feb. 2019.
[28] M. Berman, et. al., "GENI: A federated testbed for innovative network experiments", *Elsevier ComNet*, 2014.
[29] L. Guo, et al., "Joint placement and routing of network function chains in data centers", *Proc. of IEEE INFOCOM*, 2018.
[30] M. Chowdhury, et al., "Virtual network embedding with coordinated node and link mapping", *Proc. of INFOCOM*, 2009.
[31] X. Fei, et al., "Towards load-balanced VNF assignment in geo-distributed NFV infrastructure", *Proc. of IEEE/ACM IWQoS*, 2017.
[32] T. Wang, et al., "Multi-resource load balancing for virtual network functions", *Proc. of ICDCS*, 2017.
[33] Y Xiao, et al., "NFVdeep: adaptive online service function chain deployment with deep reinforcement learning", *Proc. of IEEE/ACM IWQoS* 2019.
[34] J.R. Birge, F. Louveaux, "Introduction to stochastic programming", *Springer Science & Business Media*, 2011.
[35] IBM CPLEX solver - http://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/index.html; Last accessed Feb. 2019.
[36] R. K. Ahuja, "Network flows: theory, algorithms, and applications", *Pearson Education*, 2017.
[37] D. Chemodanov, et. al., "A Constrained Shortest Path Scheme for Virtual Network Service Management", *IEEE TNSM*, 2018.
[38] X. Chen, et al., "Multi-criteria Routing in Networks with Path Choices", *Proc. of IEEE ICNP*, 2015.
[39] P. Van Mieghem, F. Kuipers, "Concepts of exact QoS routing algorithms", *IEEE/ACM TON*, 2004.
[40] V. Jeet, et al., "Lagrangian relaxation guided problem space search heuristics for generalized assignment problems", *Elsevier EJOR*, 2007.
[41] Floodlight sdn controller - http://www.projectfloodlight.org/floodlight/; Last accessed Feb. 2019.
[42] Docker containers - https://www.docker.com/; Last accessed Feb. 2019.
[43] S. Knight, et al., "The Internet Topology Zoo", *IEEE JSAC*, 2011.
[44] R. Durairajan, et al., "Internet atlas: a geographic database of the internet", *Proc. of ACM HotPlanet*, 2013.
[45] R. Ricci, E. Eide, "Introducing CloudLab: Scientific infrastructure for advancing cloud architectures and applications", *USENIX Login*, 2014.
[46] A. Ellis, et al., "Pets2009 and winter-pets 2009 results: A combined evaluation," *Proc. of IEEE PETS*, 2009.

Dmitrii Chemodanov received both his BS and MS degrees in applied math and physics from the Department of Computer Science at Samara State Aerospace University, Russia in 2012 and 2014, respectively. He received his PhD degree in Computer Science from the University of Missouri-Columbia in 2019. His current research interests include distributed and cloud computing, network and service management, and peer-to-peer networks.

Prasad Calyam received his MS and PhD degrees from the Department of Electrical and Computer Engineering at The Ohio State University in 2002 and 2007, respectively. He is currently an Associate Professor in the Department of Electrical Engineering and Computer Science at University of Missouri-Columbia. His current research interests include distributed and cloud computing, computer networking, and cyber security. He is a Senior Member of IEEE.

Flavio Esposito is an Assistant Professor in the Computer Science Department at SLU and a Visiting Research Assistant Professor in the EECS Department at University of Missouri-Columbia. He received his Ph.D. in CS at Boston University in 2013, and his MS in Telecommunication Engineering from University of Florence, Italy. His research interests include network management, network virtualization and distributed systems.

Ronald McGarvey is an Assistant Professor in the Industrial and Manufacturing Systems Engineering Department at University of Missouri-Columbia and at the Harry S Truman School of Public Affairs. He received his PhD in Industrial Engineering and Operations Research from Pennsylvania State University. His primary research interest is in applied optimization and its applications to public policy and resource management. McGarvey has more than 10 years of experience working as a researcher in Project AIR FORCE, a federally funded research and development center operated by the RAND Corporation that is tasked with performing policy analysis on behalf of Air Force leadership.

Kannappan Palaniappan received his PhD from the University of Illinois at Urbana-Champaign, and MS and BS degrees in Systems Design Engineering from the University of Waterloo, Canada. He is a faculty member in Computer Science at the University of Missouri where he directs the Computational Imaging and VisAnalysis Lab and helped establish the NASA Center of Excellence in Remote Sensing. His research is at the synergistic intersection of image and video big data, computer vision, high performance computing and artificial intelligence to understand, quantify and model physical processes with applications to biomedical, space and defense imaging.

Antonio Pescapé is a Full Professor at the Department of Electrical Engineering and Information Technology of the University of Napoli Federico II (Italy). His research interests are in the networking field with focus on Internet Monitoring, Measurements and Management and on Network Security. Antonio Pescapé has coauthored over 200 journal and conference publications and he is co-author of a patent. For his research activities he has received several awards, comprising a Google Faculty Award, several best paper awards and two IRTF (Internet Research Task Force) ANRP (Applied Networking Research Prize).