Trust-Aware Service Chain Embedding

Nariman Torkzaban
Department of Electrical
and Computer Engineering &
Institute for Systems Research
University of Maryland
College Park, Maryland, USA
narimant@umd.edu

Chrysa Papagianni
Nokia Bell Labs
Antwerp, Belgium
chrysa.papagianni@nokia-bell-labs.com

John S. Baras
Department of Electrical
and Computer Engineering &
Institute for Systems Research
University of Maryland
College Park, Maryland, USA
baras@isr.umd.edu

***Abstract*—Network function virtualization (NFV) decouples network functions (NFs) from dedicated hardware, leading to significant cost reduction in network service provisioning. With NFV, a network service is represented by a series of interconnected virtual network functions (VNFs), forming a service function chain (SFC). The problem of placing the VNFs on the NFV infrastructure (NFVI) and establishing the routing paths between them, according to the service chain template, is termed as SFC embedding. The objectives and constraints for the optimization problem formulation of SFC embedding may vary depending on the corresponding network service. We introduce the notion of trustworthiness as a measure of security in SFC embedding and thus network service deployment. We formulate the resulting trust-aware SFC embedding problem as a Mixed Integer Linear Program (MILP). We relax the integer constraints to reduce the time complexity of the MILP formulation, and obtain a Linear Program (LP). We investigate the trade-offs among the two formulations, seeking to strike a balance between results accuracy and time complexity.**

I. Introduction

With the emergence of network function virtualization (NFV) and software-defined networking (SDN), networks are becoming increasingly agile. Supported features such as programmability, IT virtualization and open interfaces, enable operators and infrastructure providers to launch a network service rapidly and in a more cost and operation efficient manner, allowing for shorter time-to-market cycles, while driving down both operational and capital costs.

Essentially, NFV implements NFs through software virtualization techniques (e.g., running on containers or virtual machines) and runs them on commodity hardware [1]. Examples of such VNFs include firewalls, load balancers, intrusion detection systems (IDS), caches etc. and more recently mobile network functions [2]. These virtual appliances can be instantiated and scaled on demand, using cloud scaling technologies, to match the service demand requirements and utilize the underlying distributed compute, storage, and networking resources in the most efficient manner. To support complex services, a sequence of NFs may be stitched together, creating an SFC, and can be represented by a graph containing VNFs as nodes and the traffic demand between those VNFs, termed VNF Forwarding Graph (VNF-FG) [3].

Towards the deployment of SFCs, the SFC embedding problem becomes of paramount importance. The problem entails the placement of VNFs on the physical hosts and establishing the routing paths between them, according to the SFC template. Appropriate approaches are needed for allocating network and computing resources to the requested SFC. Solving the service chain embedding problem requires the identification and adoption of appropriate resource allocation policies, resulting from the identified strategic objective(s) to optimize. Depending on the network service, such objective(s) may be related to QoS, profit maximization, fault tolerance, load balancing, energy saving, security etc. Many of these can be expressed as hard constraints in the problem formulation e.g., end-to-end delay in [4]. In this paper, we also consider constraints and objectives motivated by security recommendations laid out to protect systems supporting multi-tenancy.

In particular, similar to [5], we capture the notion of security by integrating "trust weights" into the service chain embedding problem. These weights indicate the trustworthiness of a host system, based on its interactions with the other network entities, location (e.g., physically secured location), the level of hardening for the host, etc. Thus, by using such weights, we enhance the trustworthiness of the service chain deployment and its resilience to attacks. We assume that such trust weights are developed based on monitoring and configuration/deployment data and are disseminated via appropriate methods so that they are timely available to the entity (e.g., orchestrator, domain controller) that performs the embedding process.

Trust weights are also considered for the VNFs in the service chain, expressing their respective requirements in terms of security, e.g., VNFs performing mission critical operations must be hosted on a trusted infrastructure. Hence, the proposed trust-aware service chain embedding approach ensures matching the requested security level of each VNF in the chain, with the security level offered by the servers that will host them.

Though the incorporation of trust seems intuitively more fit for resource allocation problems spanning multiple administrative domains, due to the uncertainty in dealing with multiple infrastructure providers (InPs), we believe that it befits also the case of a single InP, due to the distributed heterogeneous nature of the NFV Infrastucture (NFVI). For example, edge NFVI Points of Presence (PoPs) are less reliable/secure than the NFVI in the central office or core Data Centers (DCs); in a single NFVI PoP the security levels of the hosts may not be homogeneous (e.g., adoption of security zones, as collections of resources that share a common security exposure or security risk). In this paper we provide the basic formulation of the problem considering a single InP.

The remainder of the paper is organized as follows. Section II describes the trust-aware service chain embedding problem. In Section III we introduce a MILP formulation and its relaxed variant (LP). Section IV presents our evaluation results, whereas Section V provides an overview of related work. Finally, in Section VI, we highlight our conclusions and discuss directions for future work.

II. Network Model and Problem Description

Trust is widely adopted in various application domains such as e-commerce, peer-to-peer systems, cloud computing, social networks etc. Trust management systems are used to specify and collect trust related evidence, assess, establish and manage trust relationships between entities in distributed systems. Specification of the trust model and trust assessment are beyond the scope of this paper. We assume that the infrastructure provider has appropriate mechanisms in place to efficiently distribute trust evidence of the underlying infrastructure.

Network Model. We model the substrate network as a directed graph $G_S = (N_S, E_S)$. Servers, switches (e.g., nodes u, z in Fig. 1) and links are associated with their residual capacity, in terms of CPU r_u and bandwidth c_{uv}, respectively. We define trust for each node u in the substrate network as t_u.

There are various ways to quantify trust; in different trust schemes, continuous or discrete numerical values have been assigned to measure the level of trustworthiness for a network entity. For example, in [5] the trustworthiness of nodes in multi-hop wireless networks is defined as a continuous value in $[0,1]$. In [6] reputation-based trust for experimental infrastructures in a federated environment is defined in the same fashion. Following the same rationale, we denote that trust takes a continuous numerical value in $[0,1]$.

The trust estimate is updated periodically; we denote the update period as $T_{interval}$. During the update period, denoted by $[\tau - T_{interval}, \tau]$, the trust evaluation mechanism provides the trust estimate of node u denoted as t'_u. We use exponential weighted moving average (EWMA), to characterize the time-varying trustworthiness of the node over a period of time, as it reacts faster to recent updates in trust values. Hence, the new derived trust value for node u at time τ is given by

$$t_u(\tau) = (1-\alpha)t_u(\tau - T_{interval}) + \alpha\, t'_u \tag{1}$$

where $\alpha \in [0,1]$ is a constant weight indicating the preference between the updated and historic samples of trust values.

Request Model. We use a directed graph $G_F = (N_F, E_F)$ to express a service chain request (Fig. 1). The set of vertices N_F includes the VNFs that the traffic has to traverse. Each network function $i \in N_F$ can be hosted by a server at the substrate, while two VNFs are linked through a directional edge $(i,j) \in E_F$. We denote the CPU and bandwidth demands on the requested nodes and edges as g^i and d^{ij} respectively. We also denote as t^i the trust level required by each VNF $i \in N_F$.

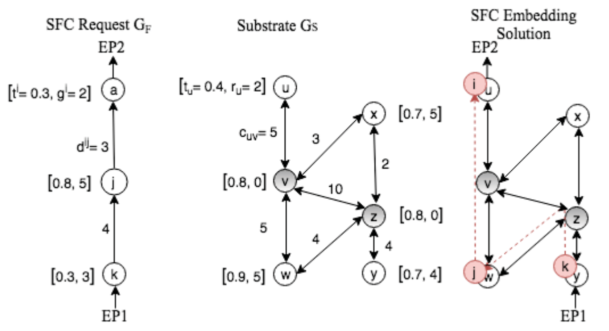

Fig. 1: Example of trust-based SFC embedding.

Trust-aware SFC Embedding. Considering trust-aware embedding of service chains, VNFs apart from their computing requirements can have also explicit requirements in terms of security, expressed via the respective trust values in the *Request*. Both can be expressed as constraints in the corresponding resource allocation problem. Thus, the trust-aware embedding solution in Fig. 1 ensures that: (i) the computing and bandwidth resources, allocated to the service chain, do not exceed the residual capacities of servers and links, respectively and, (ii) each VNF in the chain is assigned to a server that matches (at minimum) its requested trust level. For example, considering VNF j, even though servers w and x can both support its computing requirements, only server w can provide (at minimum) the required trust level. In other words the trustworthiness of the server should be equal or higher to the one requested. Binding the SFC End Points (EPs) to substrate ingress and egress nodes has been omitted for readability purposes.

III. Problem Formulation

A. MILP Formulation

In order to formulate the trust-aware service chain embedding problem, we consider:

- the set of binary variables $\mathbf{x}$, where x^i_u expresses the assignment of VNF i to the substrate node u
- the set of continuous variables $\mathbf{f}$, where f^{ij}_{uv} expresses the amount of bandwidth from substrate link (u,v) allocated to VNF-FG edge (i,j) of the SFC.

The problem is formulated as follows:

Objective:

$$\text{Min.} \sum_{i \in N_F} \sum_{u \in N_S} t_u g^i x^i_u + \phi \sum_{ij \in E_F} \sum_{uv \in E_S} f^{ij}_{uv} \quad s.t. \tag{2}$$

Placement Constraints:

$$\sum_{u \in N_S} x^i_u = 1,\ \forall i \in N_F \tag{3}$$

Trust Constraints:

$$(t_u - t^i)x^i_u \geq 0,\ \forall i \in N_F, \forall u \in N_S \tag{4}$$

Flow Constraints:

$$\sum_{v \in N_S} (f_{uv}^{ij} - f_{vu}^{ij}) = d^{ij}(x_u^i - x_u^j) \qquad \forall ij \in E_F, u \in N_S \quad (5)$$

Capacity Constraints:

$$\sum_{i \in N_F} g^i x_u^i \le r_u, \ \forall u \in N_S \quad (6)$$

$$\sum_{ij \in E_F} f_{uv}^{ij} \le c_{uv}, \ \forall uv \in E_S \quad (7)$$

Domain Constraints:

$$f_{uv}^{ij} \ge 0, \ \forall ij \in E_F, uv \in E_S \quad (8)$$

$$x_u^i \in \{0,1\}, \ \forall i \in N_F, u \in N_S \quad (9)$$

Constraint (3) ensures that each VNF is placed on exactly one server. Constraint (4) ensures that each VNF is placed on a server that can support its requested trust level. Constraint (5) enforces flow conservation; i.e. the sum of all inbound and outbound traffic in switches and servers that do not host VNFs should be zero. Moreover, this condition ensures that for a given pair of assigned nodes i, j (VNFs), there is a path in the network graph where the VNF-FG edge (i, j) has been mapped. Constraints (6) and (7) guarantee that the allocated computing and bandwidth resources do not exceed the residual capacities of servers and links, respectively. Constraints (8), (9) express the domain constraints for the variables **x** and **f**.

The first term of the objective function (2) represents the amount of CPU resources multiplied by the trust estimate of each assigned server. This term is minimized, if servers with lower security levels are preferred. Coupled with the set of constraints (4), it leads to solutions where the trust level of the servers hosting the requested VNFs is close (but over) to the requested one. The term essentially is used to get a better mapping between the requested and substrate trust levels. The second term of the objective function expresses the accumulated bandwidth assigned to the VNF-FG edges. The term is minimized if all VNFs in the chain are collocated in the same host. The parameter $\phi = \frac{1}{\sum_{ij \in E_F} d_{ij}} \left(\sum_{i \in N_F} g^i \right)$ is used for the normalization of CPU and bandwidth units.

B. LP Relaxation and Rounding Algorithm

We derive the LP model from the original MILP model by relaxing the integrality of the binary **x** variables. Thus, the domain constraints in MILP are replaced by the following:

$$0 \le x_u^i \le 1 \ \forall i \in N_F, u \in N_S, \quad f_{uv}^{ij} \ge 0 \ \forall ij \in E_F, uc \in E_S$$

As the non-binary x_u^i values do not represent mappings between the VNFs and the servers, we use a deterministic rounding technique to obtain integer values for the variables **x** and embed the SFC requests.

The pseudo-code for the LP rounding is shown in Algorithm 1. The LP solver (**Solve_LP(..)**) is called iteratively. At each iteration (Lines 2-12), one dimension x_u^i of the LP solution is rounded. The selected dimension is the maximum fractional value among **x** that, if rounded to 1, still satisfies the capacity constraint of the corresponding substrate node. The rest of the $x_v^i, \forall v \in N_S \setminus \{u\}$ fractional solutions for the particular VNF will be zero, due to constraint (3). In the case that the capacity constraint can not be satisfied for any of the fractional solutions, the request will be rejected (*Sol=false*).

Given the mapping of the VNFs to the infrastructure (integral **x** values) we solve the multi-commodity flow allocation (MCF) problem, to determine the routing paths between them, as prescribed by the SFC (Lines 14-17). The algorithm will terminate if there is no solution found for the LP (*Sol=false*), or if there are no remaining dimensions of the solution to be rounded (*Sol=true*). The *Sol* flag determines the rejection/acceptance of the request (Lines 18-23).

Algorithm 1 LP Rounding

```
1: repeat
2:     $\{x_u^i, f_{uv}^{ij}\} \leftarrow$ Solve_LP(..)
3:     If solution exists Sol:= true, Otherwise Sol:= false
4:     $X \leftarrow \{x_u^i | x_u^i \notin \{0,1\}\}$
5:     if $X \neq 0$ then
6:         $\{i_0, u_0\} \leftarrow argmax_{\{i \in N_f, u \in N_s | g_i \le r_u\}} X$
7:         if $\{i_0, u_0\}$ exists then
8:             Add LP Constraint $x_{u_0}^{i_0} = 1$
9:         else
10:            Sol=false
11:        end if
12:    end if
13: until $(X = 0) \vee (Sol = false)$
14: if $Sol = true$ then
15:    $\{x_u^i, f_{uv}^{ij}\} \leftarrow$ Solve_MCF(..)
16:    If solution exists Sol:= true, Otherwise Sol:= false
17: end if
18: if Sol:= true then
19:    return $\{x_u^i, f_{uv}^{ij}\}$ {Accept the request}
20: else
21:    $\{\forall x_u^i := 0, \forall f_{uv}^{ij} := 0\}$
22:    return $\{x_u^i, f_{uv}^{ij}\}$ {Deny the request}
23: end if
```

Contrary to the MILP formulation that is computationally very expensive to solve or even intractable for large problem instances, the relaxed linear program can be solved in polynomial-time. The LP solver is invoked at most $|N_F| + 1$ times, as every iteration, up to $|N_F|$, leads (ideally) to the mapping of a VNF, while the MCF algorithm at the end provides us with the corresponding flow allocation.

IV. Performance Evaluation

In this section, we evaluate the efficiency of the proposed trust-aware SFC embedding method; we benchmark the LP algorithm against the MILP one. Specifically, we provide an overview of the performance evaluation setup (IV-A), and discuss the evaluation results (IV-B).

A. Performance Evaluation Setup

For the simulations, we use an event-based simulator implemented in Java ([8], [9]), including an SFC and DC topology generator. We use CPLEX for our MILP models based on the branch-and-cut method. We used the dual simplex method for the LP problem. Our tests are carried out on a server with an Intel i5 CPU at 2.3 GHz and 8 GB of main memory.

NFV Infrastructure. We have generated a 3-layer fat tree network topology for the DC, consisting of 16 pods. In our evaluations, we use only one portion (or zone) of the DC consisting of 4 (out of 16) pods, utilizing 2 switches per layer and the corresponding set of 2 servers per ToR switch. For each server we consider 8 cores running at 2 GHz. Each server's initial utilization is uniformly distributed $U(0.3, 0.6)$. The ToR-to-Server link capacity is 8 Gbps while the inter-rack link capacity is 16 Gbps. The initial trustworthiness of the substrate nodes is drawn from a uniform distribution $U(0.2, 1)$.

Service Chains. We generate VNF-FGs based on three service chain templates: (i) a chain handling traffic that needs to pass through a particular sequence of VNFs, *i.e.,* a Network Address Translation (NAT) and a Firewall (FW) followed by an IDS; (ii) the second template reflects the case where traffic in a service chain is split by a particular VNF, according to some predefined policy, *e.g.,* load balancing; (iii) the last template corresponds to a bifurcated path with a single endpoint reflecting cases where one part of the traffic needs to be encrypted while another part needs to pass through a firewall [10]. For each chain the number of requested VNFs, inbound traffic demands and requested trust level is uniformly distributed, $U(5,9)$, $U(50,100)$ Mbps and $U(0.2, 0.8)$ respectively. The CPU demand of each VNF is derived from the inbound traffic rate and the respective VNF profile (cycles per packet). Resource profiles are available for a wide range of VNFs [11], [12], while profiling techniques can be employed for processing workloads whose computing requirements are not known in advance [12].

Evaluation Scenarios. We conducted two sets of simulations for two distinct cases. In the first case the trustworthiness of each substrate node does not change over time (*Experiment A: Static Trustworthiness*), while in the second one the trust estimate is updated at intervals of 500 time units, according to the description in Section III (*Experiment B: Dynamic Trustworthiness*). The purpose of the 1^{st} experiment is to benchmark the approximation algorithm against the optimal embedding approach, while the 2^{nd} one aims to showcase the impact of the trustworthiness updates in the embedding process. The new trust estimates (t'_u) are drawn from a uniform distribution $U(0.4, 1)$. The α parameter in formula (1) is tuned to 0.25 [13]. For both cases, requests arrive according to a Poisson process, with an average rate of 4 requests per 100 time units, each having an exponentially distributed lifetime with an average of 1,000 time units. Every simulation is executed for 6,000 requests and repeated for 10 iterations [9].

Evaluation Metrics. We use the following metrics for the evaluation of the two service chain mapping methods:

- **Request Acceptance Rate** is the rate of successfully embedded requests divided by the total number of requests.
- **Resource Utilization** is the amount of CPU or bandwidth units allocated for all the embedded requests.
- **Resource Revenue per Request** is the average amount of CPU or bandwidth units specified in the requests that have been embedded successfully.
- **Cumulative Resource Revenue** is the cumulative amount of CPU or bandwidth units in requests that have been embedded successfully, divided by the total number of embedding requests (successful or not successful).
- **Cumulative Resource Cost** is the cumulative amount of CPU or bandwidth units allocated to requests that have been embedded successfully, divided by the total number of embedding requests (successful or not successful).

B. Evaluation Results

1) Experiment A: Static Trustworthiness: Fig. 2 illustrates the rate of accepted requests for the LP and the MILP approaches. The acceptance rate of the LP, as expected, lags in steady state by approximately 5%. The trend in the acceptance rate is in accordance to the average CPU utilization statistics across DCs, depicted in Fig. 3. CPU utilization is below 80% in steady state for both approaches. The higher utilization levels (9%) achieved by the MILP stem from the higher request acceptance rate. Bandwidth utilization is on average less than 50%, hence not included here due to lack of space. The low link utilization can be attributed to the infrastructure properties and the fact that consecutive VNFs in a chain can be placed in the same host. Service request denials, are mainly caused due to the inability of the substrate to match the required trust values of the VNFs. In particular over time, trustworthy servers become saturated while less utilized servers cannot support the required trust levels of the SFCs, leading to service request denials.

Fig. 4 and Fig. 5 present the percentage increase of the cumulative CPU and bandwidth revenue for the MILP, as opposed to the approximation algorithm. In particular, the MILP generates in steady state approximately 10% more CPU and bandwidth revenue, compared to the LP. The result is in accordance to the above mentioned gap in the request acceptance rate between the two approaches.

Fig. 6 presents the percentage difference for the cumulative bandwidth cost between the MILP and the LP. The bandwidth cost for the MILP in steady state is approximately 16% higher than the LP. The percentage is not analogous to the percentage difference in accepted requests, as resources get fragmented over time, hence longer paths are used for SFC embedding. Since the MILP accepts more requests, the fragmentation is more severe. Consequently, higher per-request cost is associated with the MILP. Moreover, in transient state, the noted difference is due to the sub-optimal solutions of the LP. Collocation of the requested VNFs renders the bandwidth cost to zero; the approximation algorithm starts to employ multiple hosts for the embedding (hence not all VNFs of the SFC are collocated) at request #10 as opposed to the MILP that does so after request #25.

Fig. 7 and Fig. 8 present the minimum, first quartile, median, third quartile, and maximum of the CPU and bandwidth revenue per request, for the two approaches. Fig. 9 depicts the

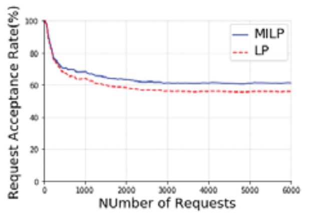

Fig. 2: Exp-A:Acceptance Rate

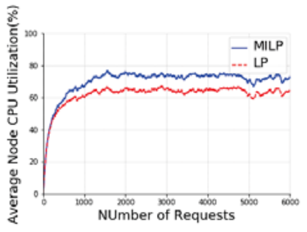

Fig. 3: Exp-A:CPU Utilization

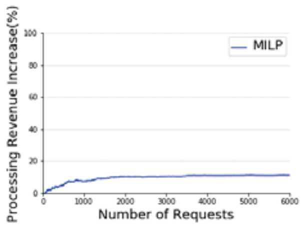

Fig. 4: Exp-A:Incremental Cum. CPU Revenue to LP

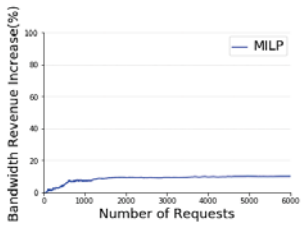

Fig. 5: Exp-A:Incremental Cum. BW Revenue to LP

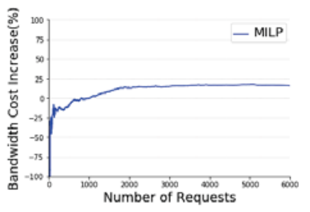

Fig. 6: Exp-A:Incremental Cum. BW Cost

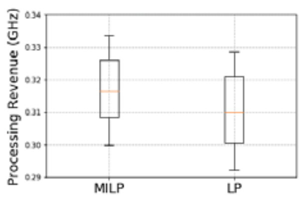

Fig. 7: Exp-A:CPU Revenue per Request

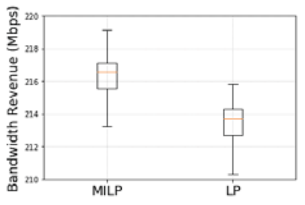

Fig. 8: Exp-A:BW Revenue per Request

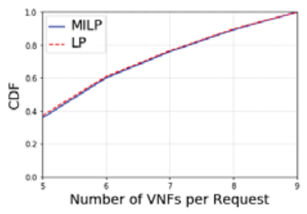

Fig. 9: Exp-A:CDF of Accepted Requests

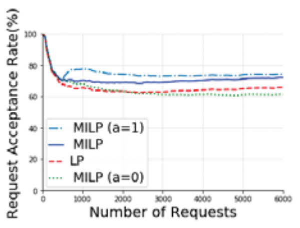

Fig. 10: Exp-B:Acceptance Rate

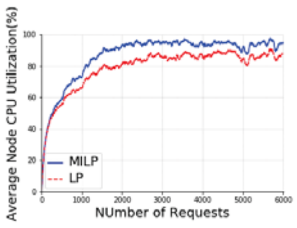

Fig. 11: Exp-B:CPU Utilization

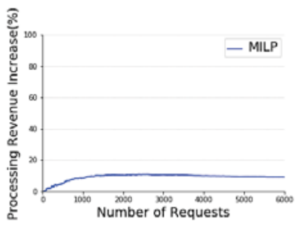

Fig. 12: Exp-B:Incremental Cum. CPU Revenue to LP

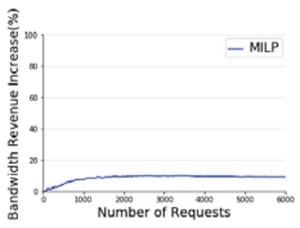

Fig. 13: Exp-B:Incremental Cum. BW Revenue to LP

CDF of the service chains' size (number of VNFs) successfully embedded by the two programs. We notice that the LP follows the same trend as the MILP, in terms of the the sizes of the requests it accepts. Moreover, the difference in revenue per request is less than 2%, for CPU and bandwidth. Both results prove the efficiency of the approximation algorithm.

2) Experiment B: Dynamic Trust Estimates: Fig. 10 illustrates the rate of accepted requests for the LP and MILP, for dynamic trust estimates. Two additional α values are used for the MILP to showcase two extreme cases; in the first case the trustworthiness of a node relies on the historical data (α=0), while in the second case it is equal to the trust estimate for the particular time interval (α =1). Fig. 11 denotes the average CPU utilization of the substrate servers for the LP and MILP. Regarding the different α values for the MILP, we notice no difference prior to the first trust update. Thereupon, since on average the trustworthiness of the servers increases, the graphs corresponding to (α = 0.25) and (α = 1) diverge from the static-trust one (α = 0) leading to higher acceptance rates. The increase, as expected, is gradual for α = 0.25, while over time converges to the graph corresponding to (α = 1).

We observe a similar trend to *Experiment A*, with regards to the acceptance ratio between the two programs. The difference in steady state is around 6%. Moreover we notice an increase in the acceptance rate for both programs over time, due to the gradual increase in the trustworthiness of the servers, according to the experiment setup. This leads to an increase in the acceptance rate by 10% compared to the static scenario. This increase is also witnessed by the corresponding increase in utilization; increased trustworthiness of the infrastructure leads to higher acceptance rate and saturation of computing resources, as trust constraints are more easily satisfied.
Figs. 12 and 13, present the percentage increase of the cumulative CPU and bandwidth revenue for the MILP, as opposed to the approximation algorithm. Results exhibit a similar trend as *Experiment A* and are in accordance with the above mentioned gap between the acceptance rate of the two approaches.

V. Related Work

In this section we briefly discuss related work on service chain embedding in general and then we focus on studies pertaining to security and trust in VNF-FG embedding.

The definition and approaches for the VNF-FG embedding problem vary, depending on a number of design decisions such as: (i) embedding objectives such as QoS [14], profit maximization [15], fault tolerance [16], load balancing [17], energy efficiency [18] etc.; (ii) solution strategy such as exact [19], heuristic [20] or meta-heuristic [21]; (iii) the technology domain where VNF-FG embedding is applied such as radio access networks [22], LTE/EPC [8], mobile core network [4], cloud networks [23]; (iv) type of resources such as CPU and bandwidth [4], processing time and buffer capacity [24], ternary content-addressable memory [9]; (v) single administrative [25] domain or multi-domain approaches[26]. Authors in [7] provide a taxonomy of SFC embedding approaches.

Trust in NFV has been considered relatively recently, with studies such as [27] that discusses the challenges associated with incorporating trust into NFVI, mentioning also the need to apply policies for binding VNFs to certain (trusted) NFV platforms depending on the platforms' configurations. The requirement for placement of NFVs onto trusted platforms/hosts is more important given the emergence of 5G for supporting vertical markets (e.g, healthcare). The problem of security awareness in resource allocation has only been addressed in the context of virtual network embedding (VNE). Liu at el. [28] abstract the security requirements of virtual networks to quantifiable security levels and formulate the problem with security constraints related to the security demands of the virtual resources and the minimum security level required for a virtual node to be hosted on a substrate node. Similar to the above we consider trust values of the substrate nodes and trust demands for the VNFs for the placement of VNFs on trusted servers. To the best of our knowledge, the notion of trust in the service chain embedding process is introduced for the first time.

VI. Conclusions

In this paper, we introduce the trust-aware service chain embedding problem, for delivering secure network service deployment on a trustworthy infrastructure. We introduce a MILP formulation, enforcing trustworthiness through appropriate trust-related constraints and trust weights in the objective function of the optimization problem. We then relax the integer constraints and use a deterministic rounding approach to obtain a polynomial-time solution algorithm.

Apart from the trustworthiness of the substrate hosts, it is also important to take into consideration the trustworthiness of the substrate network paths used; these can be potentially expressed as constraints bounded by a minimum path trust threshold. However, the aggregate trust value of a path can be expressed as the product (in the simplest formulation) of the corresponding trust values along the path, thus the corresponding terms introduce non-linearity in the problem formulation. This problem is a topic of future research.

References

[1] B. Han et al.,. "Network function virtualization: Challenges and opportunities for innovations". IEEE Communications Magazine, 53(2), pp.90-97., 2015.

[2] K. Katsalis, et al., "Network Slices toward 5G Communications: Slicing the LTE Network," in IEEE Communications Magazine, vol. 55, no. 8, pp. 146-154, Aug. 2017.

[3] ETSI NFV Architectural Overview, [Online] http://www.etsi.org/

[4] D. Dietrich et al., "Network function placement on virtualized cellular cores," IEEE COMSNETS, Bangalore, 2017, pp. 259-266.

[5] E. Paraskevas, T. Jiang, and J.S. Baras, "Trust-aware network utility optimization in multihop wireless networks with delay constraints." In IEEE Control and Automation Conf. 2016, pp. 593-598.

[6] A. Kapoukakis et al., "Reputation-Based Trust in federated testbeds utilizing user experience," IEEE CAMAD, Athens, 2014, pp. 56-60.

[7] J. Gil Herrera and J. F. Botero, "Resource Allocation in NFV: A Comprehensive Survey," in IEEE Transactions on Network and Service Management, vol. 13, no. 3, pp. 518-532, Sept. 2016.

[8] C. Papagianni, P. Papadimitriou, and J. Baras, "Rethinking Service Chain Embedding for Cellular Network Slicing", IFIP Networking 2018, pp 253-261.

[9] C. Papagianni, P. Papadimitriou, and J. Baras, "Towards Reduced-State Service Chaining with Source Routing", IEEE CNSM 2018, pp 438-443.

[10] M. C. Luizelli et al., "Piecing together the NFV provisioning puzzle: Efficient placement and chaining of virtual network functions," IEEE IFIP, Ottawa, ON, 2015, pp. 98-106.

[11] M. Dobrescu, K. Argyarki, and S. Ratnasamy, "Toward Predictable Performance in Software Packet-Processing Platforms," USENIX NSDI, San Jose, CA, USA, March 2016.

[12] A. Abujoda, and P. Papadimitriou, "Profiling packet processing workloads on commodity servers", IFIP WWIC, Russia, June 2013.

[13] J. M. Lucas, M. S. Saccucci, Exponentially Weighted Moving Average Control Schemes: Properties and Enhancements, Technometrics 32, 1-29, 1990

[14] M. Huang et al., "Throughput Maximization of Delay-Sensitive Request Admissions via Virtualized Network Function Placements and Migrations," IEEE ICC, Kansas, MO, 2018, pp. 1-7.

[15] Y. Ma, et al., "Profit Maximization for Admitting Requests with Network Function Services in Distributed Clouds," in IEEE Trans. on Parallel and Distributed Systems, vol. 30, no. 5, pp. 1143-1157, 1 May 2019.

[16] B. Yang et al., "Algorithms for Fault-Tolerant Placement of Stateful Virtualized Network Functions," 2018 IEEE International Conference on Communications (ICC), Kansas City, MO, 2018, pp. 1-7.

[17] F. Carpio, S. Dhahri and A. Jukan, "VNF placement with replication for Load balancing in NFV networks," IEEE ICC, Paris, 2017, pp. 1-6.

[18] V. Eramo, M. Ammar and, F. G. Lavacca, "Migration Energy Aware Reconfigurations of Virtual Network Function Instances in NFV Architectures," in IEEE Access, vol. 5, pp. 4927-4938, 2017.

[19] I. Jang et al., "Optimal net- work resource utilization in service function chaining," in Proc. IEEE NetSoft, Seoul, South Korea, Jun. 2016, pp. 11–14.

[20] V. Eramo et al., "An Approach for Service Function Chain Routing and Virtual Function Network Instance Migration in Network Function Virtualization Architectures," in IEEE/ACM Trans. on Networking, vol. 25, no. 4, pp. 2008-2025, Aug. 2017.

[21] C. Wang, et al., "Toward Optimal Resource Allocation of Virtualized Network Functions for Hierarchical Datacenters," in IEEE Transactions on Network and Service Management, vol. 15, no. 4, pp. 1532-1544, Dec. 2018.

[22] C. Chang, N. Nikaein and T. Spyropoulos, "Radio access network resource slicing for flexible service execution," IEEE INFOCOM 2018 - IEEE INFOCOM WKSHPS, Honolulu, HI, 2018, pp. 668-673.

[23] J. Soares and S. Sargento, "Optimizing the embedding of virtualized cloud network infrastructures across multiple domains," in Proc. IEEE ICC, London, U.K., Jun. 2015, pp. 442–447.

[24] R. Mijumbi et al., "Design and evaluation of algorithms for mapping and scheduling of virtual network functions," in Proc. of IEEE NetSoft, London, 2015, pp. 1-9.

[25] R. Cohen et al., "Near Optimal Placement of Virtual Network Functions", IEEE INFOCOM, Hong Kong, China, April 2015.

[26] D. Dietrich et al., "Multi-Provider Service Chain Embedding With Nestor," in IEEE Trans. on Network and Service Management, vol. 14, no. 1, pp. 91-105, 2017.

[27] S. Ravidas et al., "Incorporating trust in NFV: Addressing the challenges," 2017 20th Conference on Innovations in Clouds, Internet and Networks (ICIN), Paris, 2017, pp. 87-91.

[28] S. Liu et al., "Security-aware virtual network embedding," 2014 IEEE ICC, Sydney, NSW, 2014, pp. 834-840.