Towards QoS-aware Provisioning of Chained Virtual Security Services in Edge Networks

Mircea M. Iordache-Sica, Christos Anagnostopoulos, and Dimitrios P. Pezaros
School of Computing Science, University of Glasgow, G12 8QQ, UK
Email: m.iordache-sica.1@research.gla.ac.uk, {christos.anagnostopoulos, dimitrios.pezaros}@glasgow.ac.uk

***Abstract*—Future networks are expected to deliver low-latency, user-specific services in a flexible and efficient manner. Operators have to ensure infrastructure resilience in the face of such challenges, while maintaining service guarantees for subscribed users. One approach to support emerging use cases is through the introduction and user of virtualised network functions (VNFs) at the edge of the network. While placement of VNFs at the network edge has been previously studied, it has not taken into account services comprised of multiple VNFs and considerations for network security.**

In this paper we propose a mathematical model for latency-optimal on-path allocation of VNF chains on physical servers within an edge network infrastructure, with special considerations for network security applications and operator's best practices. We acknowledge the challenges of employing optimal solutions in real networks and provide the *Minimal Path Deviation Allocation* algorithm for placement of security-focused network services in a distributed edge environment, minimising end-to-end latency for users.

We then evaluate our placement results over a simulated nation-wide network using real-world latency characteristics. We show that our placement algorithm provides near-optimal placement, with minimal latency violations with respect an optimal solution, whilst offering robust tolerance to temporal latency variations.

***Index Terms*—Network Function Virtualisation, Network Service Placement, Network Service Chaining, Context-based Security, Network Security**

I. Introduction

Network Infrastructure Security implemented by Telecommunication Service Providers (TSPs) has traditionally revolved around deployment of specialised, proprietary Hardware Appliances (HAs). Such HAs are inflexible with respect to functionality and require strict placement in the network. Even slight changes in the security requirements involve manual and time-consuming reconfiguration of HAs, replacement or deployment of additional HAs.

Network Function Virtualisation (NFV) [9] has been put forward as a potential solution to address the operational challenges and minimise the costs of managing proprietary HAs in an ad-hoc manner. The underlying principle of NFV is the softwarisation of network functions (e.g., intrusion detection systems, anomaly detectors, firewalls etc) based on proprietary HAs. These newly-created Virtual Network Functions (VNFs) can be deployed and executed within virtual machines on a wide array of commodity devices. In decoupling software functionality and hardware requirements, the approach enables any network function to be flexibly deployed on any server through an automated and logically centralised management system.

978-3-903176-32-4

NFV Management and Orchestration (NFV MANO) can dynamically provision complex network services in the form of sequences of VNFs (also called service function chains (SFCs)). Network Service Chaining enables a subset of network traffic to traverse a given SFC. For example, a typical deployment would consist of a firewall, followed by an Intrusion Prevention System (IPS). Combining NFV with SFC concepts enables flexible, dynamic management of service chains to meet real-time network demands.

An approach to support the emerging use cases of next-generation networks (e.g., 5G) is to provision and manage VNFs not only within the provider's internal NFV infrastructure, but also at the distributed Multi-Access Edge Computing (MEC) infrastructure. The nodes of the network can include home routers managed by the TSP, IoT or enterprise gateways, micro-clusters or compute-enabled next-generation base stations. The usage of services at the mobile edge can significantly improve end-to-end latency and prevent unnecessary utilisation of the core network infrastructure.

One application for NFV and SFC is providing infrastructure resilience through network security. Chains of Virtualised Security Services (VSSs), security-specific VNFs such as firewalls or IDS, can be dynamically created and configured to inspect, filter or monitor network traffic. Compared to specialised HAs, VSSs have a significant impact on the performance of the network and Quality of Service (QoS) experienced by end-users. Overheads from virtualisation, server oversubscription rates and MANO techniques employed are the most significant factors that influence QoS degradation.

As presented in [2], the placement of security services within the network requires special considerations when compared to generic VNF placement. Namely, VSS generally demand increased resources to operate in real-time fashion, and are highly susceptible to oversubscription from multiple network flows, as evidenced in [1]. In order to facilitate these vulnerabilities, a placement and scheduling mechanism, where a per-flow service chain is emplyed, is required.

We argue that, for wide adoption of VSS, provisioning should not only take into consideration the security policies, but also account for specific QoS needs of applications. Omitting the latter may lead, for instance, to a strategy that blindly forces network flows to traverse an unneeded SFC.

As a result, computationally demanding services, such as IDS may cause a noticeable performance degradation to latency-sensitive applications (e.g. augmented reality systems). On the other hand, beyond the overall QoS and resource allocation requirements, the allocation strategy should also consider operator-specific best practices and policies. Ignoring such aspects can result in breaches of infrastructure security by e.g., inappropriate placement of VSS within the network.

In this paper we present a novel approach to provisioning and allocation of security services by composing VSS chains according to specific QoS needs of user applications and security policies defined by TSPs. The security policies (an input to our proposed provisioning model) include: the VSS type (e.g., firewall, IDS, ACL, etc) that should be deployed for a specific class of applications, their order (e.g., firewall followed by IDS).

We define an Integer Linear Programming (ILP) formulation and associated heuristic algorithm to tackle the provisioning problem in dynamic networks, where service requests are not known in advance. The *a priori* knowledge of requests is an assumption made by a majority of related works. Although the focus of our proposal revolves around security-specific services, the proposed approach can also be applied to more generic scenarios, where heterogeneous network services provided by generic VNFs coexist (e.g., security, content caching, video transcoding, etc).

The remainder of this paper is structured as follows: Section II gives the relevant background information that this work builds upon and the related work present within the field. Section III details the motivation behind this work, with example use-cases. Section IV provides insight into the mathematical formulation of the ILP model, and Section V describes the heuristic approximation that we implemented to solve the problem. In Section VI we evaluate our work on real-world topologies. Finally, conclusions are presented in Section VII.

II. Background and Related Work

The emergence of "softwarised" networks led to a multitude of research projects in the last few years. A subset of these research initiatives address the optimal placement of chained VNFs. A popular approach to the problem revolves around the use of linear programming techniques and heuristic algorithms to address scalability issues. In this section, we analyse and classify the most relevant works in this domain.

A. Context-based Security Policies

Context-based Security Policies are a way to tailor a security deployments to individual user's requirements. This paradigm, is presented in [19]. It proposes a trade-off by dedicating individual VSSs and chains to a specific user, instead of grouping flows for batch inspection. This paradigm, made accessible through NFV and SFC is gaining adoption in enterprise environments where critical infrastructure must be protected. The concept allows for flexible definitions of security requirements and reduced configuration overhead. The placement of proposed services however, is not considered with respect to NFV infrastructure.

B. QoS-aware VNF Placement

An approach revolving around network properties is presented by Gupta et al [10]. The authors present a linear programming problem that minimises bandwidth consumed by routing traffic through selected paths. Placement of VNFs is performed at optimal locations in the a programmable network environment. The term "Network-enabled Cloud" (NeC) is also introduced, which describes a cloud environment with extensions provided by programmable packet and optical network nodes. Results presented show a reduction in network resource consumption.

In [15] a formulation for optimal use of network resources is provided for the placement of VNF chains. Their simulation results show how the ILP approach can accommodate more flows and reduce the overall network resource usage. However, the presented simulation only features two types of SFC requests, with no operator-defined composition strategy.

In [17], the resource allocation architecture includes two models whose goals are optimising energy consumption and number of flow entries. The presented ILP formulation and associated heuristic algorithm favour the minimisation of the flow entries and number of servers used to host VNFs, leading oversubscription of computational resources.

The model proposed in [6] focuses on placement of individual VNFs, considering bandwidth and latency constraints to minimise the end-to-end latency. However, the focus is on individual VNFs that act as traffic endpoints (e.g., caches, video transcoders, etc) and service chains from source to destination are difficult to achieve under the proposed architecture.

In [4] an optimisation problem for mobile core networks to deploy VNFs is presented. Placement onto nodes of the physical substrate network and optimal traffic routing between nodes is achieved through the work. As many other works in the field, the objective is to minimise the cost of occupied links and node resources. Simulation results over two nation-wide network topologies outperform traditional Virtual Network Embedding optimisation approaches.

Vizarreta et al. [18] propose a cost-focused approach that minimises estimated Capital and Operational Expenditure. As a result, the QoS aspect becomes a secondary objective in their proposed model, and is impacted by other factors within the network.

C. Security-oriented VNF Placement

The authors in [3] address placement of security services in virtualised environments. The problem is modelled as a Mixed-Integer Linear Programming (MILP) model on the ISP environment, with the goal of minimising costs of network operators. The work revolves around individual VSS, and does not consider the possibility of using SFC concepts to create complex chains.

[12] put forward two strategies for service placement: tenant-centric and service-centric. The former results in co-location of multiple functions on the same server, leading to

improvement of bandwidth usage and reduction of incurred network delays. Service-centric tries to group together network flows towards the same VNF to facilitate it's reuse. This approach assumes infinite processing capabilities of host devices and incurs significant overheads from rerouting of network traffic.

Doriguzzi-Corin et. al [8] propose the progressive provisioning of security services (PESS) model. It estimates the processing delay based on residual computing resources and factors it into allocation of services. The objective function aims to minimise cumulative usage of physical resources. Acknowledging the limitations of an ILP formulation, the authors also propose a heuristic solution. Both approaches propose sharing of security services between multiple users, which increases related management overheads when user migrations occur within the network. Furthermore, the evaluation is performed with the main goals of minimising residual bandwidth of the networks and VNF host CPU availability. The resource availability of the aforementioned hosts is akin to that of a TSP's core infrastructure, with limited consideration towards edge networks.

The model proposed in [2] is different, being presented as a variable cost - variable sized bin packing problem (VSBPP). The placement is influenced by server resource availability and aims to increase overall requests. The solution proposed, aimed at multi-tenant cloud data centre environments, does not take into consideration QoS metrics, and assumes only individual requests with no complex services by chaining together of VSSs.

III. Motivation

Our work is motivated by two use-case scenarios that are becoming of increased relevance in next-generation networks where TSPs exploit SFC and NFV technologies to provide infrastructure resilience through network security services.

Public Infrastructure: Multiple mobile devices within the network wish to interact with IoT Sensors and their Gateways to query the status of various measurements (e.g., air quality, parking spaces). An IDS might be used to detect possible threats to these devices due to malicious intent or deliberate misuse (e.g., rootkits, botnets, etc). The network may contain some Augmented Reality (AR) data to enhance users' experience within the physical environment. As a result, an expectation for low latency by the application is created, in order to provide optimal user experience.

Autonomous Vehicles: In this scenario, vehicles traversing the network require timely communication with their neighbours to exchange important information (e.g., nearing a traffic junction). Other auxiliary services required by the vehicle (e.g., navigation) may not be as strict in their timing requirements. An added consideration is the low network lifetime of such a device, while requiring rapid service delivery.

In line with the motivations provided above, we summarise the high-level goals for our proposed approach: (i) a user should experience minimal degradation in Quality-of-Service because of VSS operations, (ii) placement of VSS chains should respect the service provider's best practices.

IV. Latency-based Inline VNF Chain Placement

In this section we introduce and formalise the mathematical model to allocate and route service requests, formed by one or multiple VSS, on a physical network infrastructure as an Integer Linear Program (ILP).

The model takes as input a model of the physical network, including the current resource availability of computing resources of servers alongside one or more security service requests and additional security policies of the TSP (in the form of additional constraints within the model). The output is a mapping of VSS onto servers within the network, the recommended latency-optimal route between the traffic source and destination traversing the VSS chain, and an updated model that takes into account the resulting allocation.

A. System model

We represent the physical network as an undirected graph $\mathbb{G} = (\mathbb{H}, \mathbb{E}, \mathbb{U})$ where $\mathbb{H}$ is the set of VSS-capable hosts, $\mathbb{E}$ denotes the network links between hosts, and $\mathbb{U}$ represents the users connected to the network. We assume that a VSS can be placed on any host in this graph, and all hosts have a finite hardware capacity (combined *cpu, memory, io, etc*) to host VSSs, denoted as $W_j, j \in \mathbb{H}$. Any link within the network is characterised by a latency value A_m. Because of the short-lived nature of flows, and limited bandwidth usage of applications used in our proposed environments, we omit the bandwidth physical limit of links.

We model flows between two users as (s, d) pairs and a set of paths $\mathbb{P}_{s,d}$ that connect the two hosts. A security service request takes the form $\mathbb{N}_{s,d}$ and consists of multiple VSSs $n^i_{s,d}$. Each service request is characterised by the upper latency bound $\theta_{s,d}$ of the associated SLA. This parameter is dictated by the application class associated with a given flow. Because of the different types of VSS types, we associate the *cpu, memory, io, etc* requirements with R_i for each VSS n_i.

We use the binary decision variables:

$$X^k_{s,d} = \begin{cases} 1 & \text{if the flow } (s,d) \text{ is routed on path } k \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

$$Y^j_i = \begin{cases} 1 & \text{if VSS } n^i_{s,d} \text{ is located on host } h_j \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

and

$$Z^{ij}_{(s,d)k} = \begin{cases} 1 & \text{if VSS } n^i_{s,d} \text{ is located on host } h_j, \\ & \text{belonging to path } p_k \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

The variable described in Equation (1) encompasses the path selection and routing decision of flows. Variable (2) is linked to the placement decision of individual VSS. We ensure consistency between the two variables using (3), which encompasses all of the relevant system parameters.

TABLE I: System Parameters

Network Parameters	Description
$\mathbb{G} = (\mathbb{H}, \mathbb{E}, \mathbb{U})$	Graph of network topology.
$\mathbb{H} = \{h_1, h_2, h_3 ... h_H\}$	VSS Hosts within the network.
$\mathbb{E} = \{e_1, e_2, e_3 ... e_E\}$	All network links.
$\mathbb{F} = \{u_1, u_2, u_3 ... u_U\}$	All flows associated with network functions.
$\Phi = \{(s_1, d_1), (s_2, d_2)(s_3, d_3) ... (s_U, d_U)\}$	All source and destination pairs of flows in the network.
$\mathbb{P}_{s,d} = \{p_1, p_2, p_3 ... p_P\}$	All paths in the network from source s to destination d.
W_j	Hardware capacity of $h_j \in \mathbb{H}$.
A_m	Latency on link $e_m \in \mathbb{E}$.
VSS Parameters	**Description**
$\mathbb{N}_{s,d} = \{n^1_{s,d}, n^2_{s,d} ... n^F_{s,d}\}$	Number of network functions to allocate where $n_i \in \mathbb{N}_{s,d}$ is associated to source and destination pairs $(s, d) \in \Phi$.
$\theta_{s,d}$	Upper latency bound for security service $N_{s,d}$.
R_i	The VSS host requirements of network function $n^i_{s,d} \in \mathbb{N}_{s,d}$.
Derived Parameters	**Description**
$l^k_{s,d} = \sum_{e_m \in \mathbb{E}} A_m$	Latency between the source-destination pair (s, d) using the path p_k. Derived from the physical topology and the VSS requests.
Variables	**Description**
$X^k_{s,d}$	Binary decision variable denoting if traffic between the (s, d) pair is going through path p_k or not.
Y^j_i	Binary decision variable denoting if network function $n^i_{s,d}$ is hosted on host h_j.
$Z^{ij}_{(s,d)k}$	Binary decision variable denoting if network function $n^i_{s,d}$ is located on host h_j of path p_k between (s, d).

We have summarised and synthesised the parameters used for the formulation of the problem and model in Table I

B. Problem Formulation for Latency-Optimal VSS Chain Placement

The placement problem is defined as follows:

Problem 1: Given the the network $\mathbb{G}$ described by set of users $\mathbb{U}$, the set of VSS Hosts $\mathbb{H}$, the set of network edges $\mathbb{E}$, with the traffic flow pairs Φ, the set of VSS chain requests $\mathbb{N}_(s, d)$, and a latency matrix l, we need to find an appropriate routing of all flows and placement for all VSSs that minimises the total expected end-to-end latency from all users to their destinations.

The solution to Problem 1 is given by:

$$min. \sum_{(s,d) \in \Phi} \sum_{p_k \in \mathbb{P}_{s,d}} X^k_{s,d} l^k_{s,d} \tag{4}$$

C. Constraints

Equation 4 looks for the values of $X^k_{s,d}$, while it is subject to the following constraints:

$$\sum_{p_k \in \mathbb{P}_{s,d}} X^k_{s,d} = 1, \forall (s, d) \in \Phi \tag{5}$$

$$\sum_{(s,d) \in \Phi} \sum_{p_k \in \mathbb{P}_{s,d}} X^k_{s,d} l^k_{s,d} < \theta_{s,d}, \forall n^i_{s,d} \in \mathbb{N}_{s,d} \tag{6}$$

$$\sum_{h_j \in \mathbb{H}} Y^j_{(s,d)i} = 1, \forall (s, d) \in \Phi, \forall n^i_{s,d} \in \mathbb{N}_{s,d} \tag{7}$$

$$\sum_{(s,d) in \Phi} \sum_{n_i \in \mathbb{N}_{s,d}} Y^j_i R_i < W_j, \forall h_j \in \mathbb{H} \tag{8}$$

$$Z^{ij}_{(s,d)k} = X^k_{s,d} Y^j_i \tag{9}$$

$$\sum_{p_k \in \mathbb{P}_{s,d}} Z^{ij}_{(s,d)k} >= 1, \forall (s, d) \in \Phi, \forall n^i_{s,d} \in \mathbb{N}_{s,d}, \forall h_j \in \mathbb{H} \tag{10}$$

Constraint (5) ensures that only one valid path may be used for any given flow. This means that all network traffic belonging to a given flow follows the same network path, which contains VNF allocations on physical servers. The constraint gives two guarantees: that there are no instances of VNFs performing the same task, and that the network traffic traverses all VNFs on the Service Chain.

Constraint (6) verifies that the selected end-to-end (E2E) path latency is below the upper bound specified for the flow at time of placement ($\theta_{s,d}$). That means that the traffic traversing selected path, and associated VNF allocations, will be below the threshold for the application traffic used. The calculation for all possible latency values is stored in the latency matrix $l^k_{s,d}$ where the s, d pair indicates the traffic source and destination and k denotes the network path used between the two points.

Constraint (7) states that each VSS must be allocated to exactly one host, while constraint (8) enforces that resource limitations of hosts are adhered to (CPU, memory, IO are finite and can only support a limited number of VSS). This constraint sums up all of the resource usage requirements R_i (selected by the binary decision variable Y^j_i) and ensures that for all hosts h_j the individual sums are below W_j, the total capacity of the host.

Algorithm 1 Minimal Path Deviation Allocation

```
def mpda_allocation(flows):
  for f in flows:
    chain = get_chain(f)
    paths = get_paths(f.src, f.dst)
    paths.sort(key=len)
    for p in paths:
      if can_allocate(chain, p):
        a = allocate(chain, p)
        update_resources(a)
        allocations.add(a)
```

Constraint (9) is used to guarantee that the allocation is valid with respect to the path (i.e., the chosen host is located on the path).

Finally, we enforce the chaining requirement through constraint (10), where all VNF belonging to the same Service Chain are on the same path. Doing so ensures that a chain does not reside on multiple paths, violating network traffic steering considerations by e.g., duplicating packets.

Operator-Defined Policies: We propose two complimentary methods to integrate operator policies for best-practices within the model. The first approach is to provide additional system parameters and constraints for specific operator requirements. For example, an available link bandwidth network parameter b_m and an associated VSS β^i bandwidth requirement can be added alongside a derived parameter $B^k_{s,d} = \sum_{e_m \in \mathbb{E}} b_m$. To adhere to available network bandwidth resources a constraint, similar to the one stated in (11) can be added to the model:

$$\sum_{(s,d)\in\Phi} \sum_{p_k \in \mathbb{P}_{s,d}} X^k_{s,d}\beta^i < B_{s,d}, \forall n^i_{s,d} \in \mathbb{N}_{s,d} \qquad (11)$$

For policies that are related to positions in the network, the associated network graph can be divided to adhere to the operator's desires. 'Virtual' users can be defined for gateways between two sub-networks, and service chain requests can be split to reflect regional preferences as denoted by operators. Similarly, a sub-region of the network can be abstracted to a single host node to denote special availability (e.g., a dedicated processing cluster within an edge network).

V. Minimal Path Deviation Allocation heuristic

The model we propose in Section IV is designed to provide an optimal placement for service chain requests within a given network topology. It has been solved using a commercial Integer Linear Programming solver. However, given the complexity of the ILP model, the time required to produce solutions is outside the acceptable range for the dynamic scenarios considered. Furthermore, the model assumes a coherent and consistent snapshot of the network-level information (e.g., link-layer latencies) that are subject to temporal variations. As a result, we also implement a heuristic algorithm to find near-optimal solutions rapidly.

Algorithm 2 Evaluation for chain allocation

```
def can_allocate(chain, path):
  remaining_vss = chain.copy()
  for vss in chain:
    remaining_vss.pop()
    for server in path:
      if not can_host(server, vss):
        continue
      vss.potential_host = server
      break
  if remaining_vss:
    return False
  return True
```

The necessary network state shapshotting is difficult to achieve in large, multi-link edge networks. For general network edge environments, the distances between network nodes range between a few meters (in the case of dense small-cell networks: e.g., 5G, WiFi) to a few kilometers (for customer-provided equipment connected to a demarcation point). The short distance results in reduced propagation delays and can be leveraged for reduced latency in service placement. We decided that exchanging in-network measurements with network distances is a reasonable trade-off in placement of services in realistic edge environments.

As opposed to allocation in batches, as performed in the model presented in Section IV, we decided on an incremental approach that can be prioritised by the operator. As a result, our heuristic aims to place a service request independently of other requests, based on current infrastructure conditions and host capacity.

Consequently, we propose a Minimal Path Deviation Allocation heuristic that conforms to realistic network conditions. Our algorithm performs incremental allocation of service chain requests based on deviation from the shortest network path. Our allocation strategy still ensures that Constraints (5) - (10) are respected in an efficient manner, but relaxes the optimality requirement with respect to measured link-layer latency.

In Algorithm 1, we consider all possible paths for allocation: the function `get_paths` generating the possible noncyclic network paths between the source and destionation, sorting them based on path length. In our prototype implementation, we use the NetworkX Python Library to represent the network infrastructure and make use of the included functions for retrieving the previously mentioned paths.

Analysing the paths in order of length, we determine if a service chain can be allocated on the given path using the outline presented in Algorithm 2. For any given path and service chain, we consider placement of each constituent VSS within the chain on the closest server. We analyse the resource availability for the server using the `can_host` function, by comparing the free server resources to the requirements of the VSS, and mark the VSS for allocation onto the server if they

are met. If the service chain can be fully placed on a given network path, the allocation is performed by informing the servers of the deployment through the `allocate` method, and the model of network resource availability is updated accordingly.

Addition of operator-defined policies can be done by extending the proposed allocation framework to include additional information. For example, link bandwidth information can be introduced globally within the model, as it does not suffer from the same temporal variation challenges as latency. An additional pre-filtering step for allocation can be installed to preemptively remove paths that have lower bandwidth availability (based on the network resource utilisation model) before exhaustive allocation analysis is performed.

Similarly, policies related to network-wide positioning can be performed by partitioning service chain requests into segments and selection of desired network 'boundary points' (e.g., IoT Gateways, WAN Gateways) to act as intermediate 'virtual' users.

VI. Evaluation

We have implemented the Minimal Path Deviation Allocation algorithm as described in Section V. To show the properties of such a system, we have designed a series of experiments. After presenting our proposed experimental environment (Section VI-A) that was modelled using a real-world topology and network-level measurements, we evaluate placement of our proposed algorithm with respect to the Optimal Solution (as presented in Section IV) under static network conditions and show the emerging latency benefits of using in-path placement of service chains (Section VI-B). We further analyse the properties of the proposed algorithm with respect to traffic steering in Section VI-C. We finish our evaluation by introducing temporal latency variation and present the robustness of the placement scheme in Section VI-D.

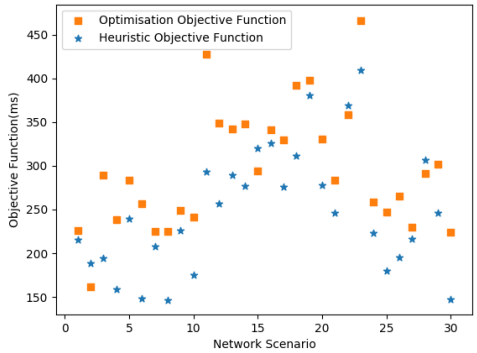

Fig. 1: Cumulative path latency comparison

A. Experimental Setup

1) Network Topology: As a basis for the network topology used, we have used the nation-wide Jisc NREN backbone network, as reported on Topology Zoo[1]. In order to approximate edge resources, we have assumed finite resource availability at all points of presence in this network topology, with each server able to host a limited number of VSS. The deployment where computational availability is present alongside each network device is in close alignment with the ETSI MEC suggested scenario [14]. We have introduced two Cloud Data Centres to be used, to represent the internal NFV infrastructure with unlimited capacity for hosting of VSS.

Application type	**Latency SLA**
Hard Real-time (e.g., autonomous vehicles)	5ms
Soft Real-time (e.g., AR/VR)	10ms
Near real-time (e.g., conferencing)	30ms
Non real-time (e.g., data transfer)	100ms

TABLE II: Application classes and their latency requirements

2) Application Latency Requirements: We have categorised the applications based on the expected end-to-end latency of packets, giving our summary in Table II. We assign a total latency $\theta_{s,d}$ for each request $\mathbb{N}_{s,d}$ representing the maximum allowed latency for the flow, beyond which the user notices application performance degradation. This information is derived from work showcasing the benefits of next-generation networks and the envisioned applications [5], [13], [16].

3) Latency Modelling: End-to-end latency has been modelled using millions of end-to-end latency measurements from real-world applications. Data has been collected from the New Zealand research and education wide-area network provider REANNZ using Ruru [7]. Modelling has been performed in a similar manner to the process described in [6], with individual link latency values sampled from a Gamma distribution ($k = 2.2, \theta = 0.22$) to create a representative time series.

B. Static Allocation

The optimisation problem described in Section IV has been formulated as an ILP model and implemented using the Gurobi solver [11]. The solver calculates the optimal service chain placement (and total source-to-destination end-to-end latency for all flows as an objective function) for our problem. Our Minimal Path Deviation Allocation heuristic provides similar outputs as the solver, with both placement of VSS and cumulative latency.

In this experiment we assign users to edge locations in a randomised order and assign one flow per user in a round-robin fashion. We assume a chain of 2 VSS per user (mimicking a firewall and IDS setup commonly encountered) and assign computing capabilities to all edge nodes with a total capacity of 100 VSS (ca. 4 VSS per edge node).

In Figure 1 we look into the cumulative latencies for both allocation strategies when placing requests for multiple numbers of flows. We notice that, even without network-wide information regarding individual link latencies, the proposed heuristic Minimal Path Deviation Allocation provides overall lower cumulative latencies. We attribute this to preference

[1]http://www.topology-zoo.org

of shorter paths which have smaller delays in propagation and switching. This initial metric gives an overall appropriate estimation on placement performance, but can fine-grained generate latency violations.

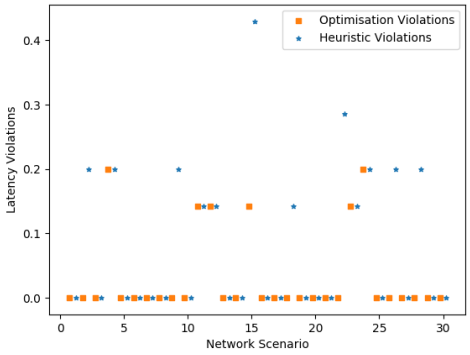

Fig. 2: Initial placement efficiency

We compare the efficiency of initial allocations using the proposed algorithm with the optimal solution. As can be seen in Figure 2, in the majority of cases, our proposed Minimal Path Deviation Allocation algorithm performs optimal or near-optimal placement of service requests. The number of application requests that initially cause latency violations is 23.3%. Of particular interest is the scenario where an allocation causes 42.85% of the applications to suffer from latency violations. We have identified that the later service requests were allocated on longer paths because of the prioritisation model employed. In real-world situations, these types of issues can be mitigated by sorting the incoming requests by application latency requirements prior to allocation.

C. Deviation from shortest and optimal path

Our Minimal Path Deviation Allocation algorithm calculates the shortest path which contains the necessary hardware resources to fulfil a service request. In Figure 3 a comparison between the path lengths of the optimal and heuristic allocation models is provided. We normalise these results with respect to the shortest path between the user and destination. Using longer paths for traffic routing in the optimal solution gives improved initial latency. However, in large networks, we argue that hardware limitations of switches become apparent, increasing the switching delay and leading to overpopulation of forwarding tables.

D. Dynamic Network Behaviour

The link latencies change over time (due to user utilisation), the placement of VSS is subject to temporal variations. These deviations from a previously optimal allocation may result in latency violations which note application performance degradation. We analyse the resulting violations to provide key insights into dynamic rescheduling of service chains.

This experiment expands on the one presented in Section VI-B by introducing a temporal component that updates

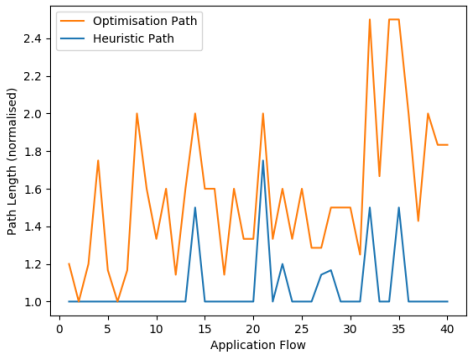

Fig. 3: Path lengths compared to shortest network path

the latency matrix l every time instance t (such a time instance can be e.g., 1 minute in a real-world network). We then analyse the number of latency violations that occur over the given time period.

We run the simulation over 100 time instances. We present the behaviour of our placement system. As shown in Figure 4, the number of latency violations occurring for the optimal placement is significantly higher than the violations generated by the Minimal Path Deviation Allocation heuristic.

We attribute this increased robustness to temporal latency variations to the shorter paths used in placement of services. The optimal allocation, on the other hand, attempts to minimise the end-to-end latency at a given time instant, thus introducing multiple delay-sensitive elements within the chosen path. When also factoring in the time required for finding an optimal solution for the allocation problem, the heuristic solution is preferable, even if certain latency violations are still encountered.

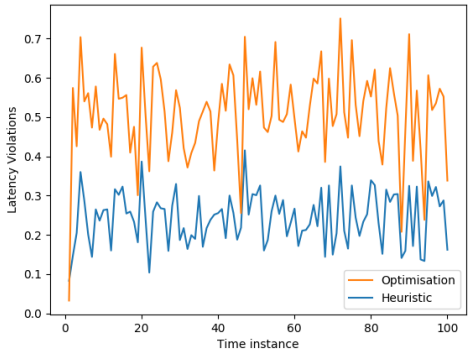

Fig. 4: Latency violations per time instance

E. Robustness of Heuristic

The Minimal Path Deviation Algorithm and Latency-Optimal VSS Chain Placement Optimisation Problem have been designed with network edge scenarios in mind. We however consider the possibility of harsh network conditions

where the network latency is similar to core and Internet scenarios.

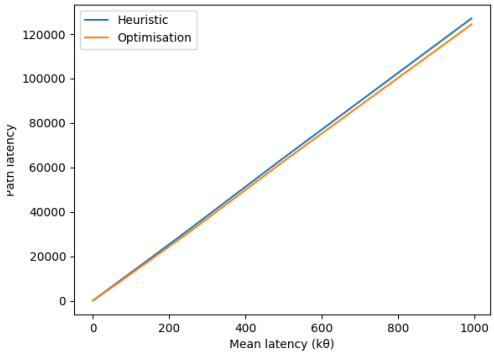

Fig. 5: Heuristic robustness

In order to maintain the characteristics of studied network traffic, according to [7], we maintained the same Gamma distribution characteristics, and scaled the resulting mean latency $k\theta$ by a constant α. We used values from $2^0 to 2^1 4$ to analyse the behaviour of our proposed approach in adverse network conditions. Each link latency is independently sampled from a Gamma distribution with the characteristics described.

In Figure 5 we compare the resulting objective functions (cumulative path latencies) of the optimisation model proposed in Section IV and the heuristic MPDA described in Section V. We average out the measurements of multiple experimental runs with the same mean latency distribution to obtain an indicator for the performance of the proposed solutions. Although faced with adverse network conditions, the heuristic placement of VSS in the network provides near-optimal results, with no significant deviation from the optimisation model used.

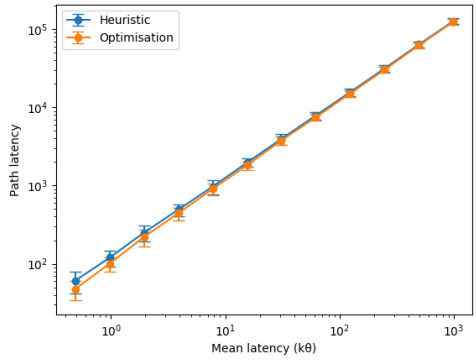

Fig. 6: Mean placement latencies and errors

Furthermore, we look at the mean placement latencies and variation in Figure 6. Using this technique, the mean and average values for the objective functions present close values, with no significant outliers detected in placement strategies. We can also see the variation in placement (error) is not directly linked to the increases in the mean latency scaling.

VII. Conclusions

In this paper, we have argued that in order to sustain emerging applications in next-generation networks while ensuring operators' need for infrastructure resilience, an inline service placement strategy has to be adopted for NFV deployment. We have defined an optimal placement model that aims to provide minimal end-to-end latency for multiple users in a network. Our placement model is defined as an Integer Linear Programming solution that places virtualised services onto physical servers within an edge network topology, taking into account resource consumption and availability, and improving the QoS the network.

We acknowledge the challenges of using optimisation models in real-world environments and put forward a Minimal Path Deviation Allocation algorithm to place services within the infrastructure. We argue our choice of a heuristic algorithm based on the behaviour of modern networks, where highly dynamic clients requires real-time allocation of services.

We have evaluated the proposed system using a simulated nation-wide network topology with real-world latency characteristics and users running latency-sensitive applications. Our results show that our placement algorithm performs in an near-optimal manner, and maximises the network infrastructure resources.

We consider the minimisation of the network path a valuable approach in orchestration of services due to elements that are susceptible to temopral influences within the network (e.g., congestion, link failures, jitter). We model the overall latency fluctuations using real-world data and integrate the findings into our experimental setup, in order to provide an enhanced simulation of network behaviour over time. By applying such a placement strategy with dynamic VNF migration strategies, operators can minimise service interruption while meeting service guarantees to subscribed users.

Although primarily intended for network edge scenarios, evaluation into the robustness of both the optimisation model and our proposed MPDA solution can also be employed within other network contexts. We envision the posibility of placement of VSS in core infrastructure architectures where operators maintain multiple points of presence (PoPs) to require minimal alterations to the placement model.

Further research into the topic includes development of generic, composable lightweight VSS that can be used in enhanced context-based security systems. Applying the proposed placement system over emerging networks that further virtualise the infrastructure (e.g., with network slicing) can enable TSPs to provide value-added services targeting specific behaviour while maintaining infrastructure resilience. Finally, the work could be expanded to integrate programmable data-plane solutions.

Acknowledgments

This work is sponsored in part by the UK Engineering and Physical Sciences Research Council (EPSRC) grants EP/N509668/1 and EP/N033957/1.

REFERENCES

[1] Yehuda Afek, Anat Bremler-Barr, Yotam Harchol, David Hay, and Yaron Koral. Making DPI Engines Resilient to Algorithmic Complexity Attacks. *IEEE/ACM Transactions on Networking*, 24(6):3262–3275, 2016.

[2] Abeer Ali, Christos Anagnostopoulos, and DImitrios P. Pezaros. Resource-aware placement of softwarised security services in cloud data centers. *2017 13th International Conference on Network and Service Management, CNSM 2017*, 2018-January(i):1–5, 2017.

[3] Cataldo Basile, Christian Pitscheider, Fulvio Risso, Fulvio Valenza, and Marco Vallini. Towards the Dynamic Provision of Virtualized Security Services. In *Communications in Computer and Information Science*, volume 470, pages 65–76. 2015.

[4] Andreas Baumgartner, Varun S. Reddy, and Thomas Bauschert. Mobile core network virtualization: A model for combined virtual core network function placement and topology optimization. *1st IEEE Conference on Network Softwarization: Software-Defined Infrastructures for Networks, Clouds, IoT and Services, NETSOFT 2015*, 2015.

[5] Mark Claypool and Kajal Claypool. Latency and player actions in online games. *Communications of the ACM*, 49(11):40, nov 2006.

[6] Richard Cziva, Christos Anagnostopoulos, and Dimitrios P. Pezaros. Dynamic, Latency-Optimal vNF Placement at the Network Edge. *Proceedings - IEEE INFOCOM*, 2018-April:693–701, 2018.

[7] Richard Cziva, Christopher Lorier, and Dimitrios P. Pezaros. Ruru: High-speed, flow-level latency measurement and visualization of live internet traffic. *SIGCOMM Posters and Demos 2017 - Proceedings of the 2017 SIGCOMM Posters and Demos, Part of SIGCOMM 2017*, pages 46–47, 2017.

[8] Roberto Doriguzzi-Corin, Sandra Scott-Hayward, Domenico Siracusa, Marco Savi, and Elio Salvadori. Dynamic and Application-Aware Provisioning of Chained Virtual Security Network Functions. *IEEE Transactions on Network and Service Management*, 17(1):294–307, 2020.

[9] ETSI. Network Functions Virtualisation,An Introduction, Benefits, Enablers, Challenges & Call for Action. Technical report, 2012.

[10] Abhishek Gupta, M. Farhan Habib, Pulak Chowdhury, Massimo Tornatore, and Biswanath Mukherjee. On service chaining using Virtual Network Functions in Network-enabled Cloud systems. *International Symposium on Advanced Networks and Telecommunication Systems, ANTS*, 2016-February:1–3, 2016.

[11] LLC Gurobi Optimization. Gurobi optimizer reference manual, 2020.

[12] Xin He, Tian Guo, Erich M. Nahum, and Prashant Shenoy. Placement strategies for virtualized network functions in a NFaaS cloud. *Proceedings - 4th IEEE Workshop on Hot Topics in Web Systems and Technologies, HotWeb 2016*, pages 48–53, 2016.

[13] Rhys Hill, Christopher Madden, Anton Van Den Hengel, Henry Detmold, and Anthony Dick. Measuring latency for video surveillance systems. *DICTA 2009 - Digital Image Computing: Techniques and Applications*, pages 89–95, 2009.

[14] Yun Chao Hu, Milan Patel, Dario Sabella, Nurit Sprecher, and Valerie Young. Mobile edge computing—A key technology towards 5G. ETSI White Paper. *ETSI White Paper*, 11(11):1—-16., 2015.

[15] Insun Jang, Sukjin Choo, Myeongsu Kim, Sangheon Pack, and Myung Ki Shin. Optimal network resource utilization in service function chaining. *IEEE NETSOFT 2016 - 2016 IEEE NetSoft Conference and Workshops: Software-Defined Infrastructure for Networks, Clouds, IoT and Services*, pages 11–14, 2016.

[16] Jaydip Sen. Visions of 5G Communications. *The ENVISON Journal of Tata Consultancy Services Ltd.*, 1(1):4 – 15, 2009.

[17] Mohammad M. Tajiki, Stefano Salsano, Luca Chiaraviglio, Mohammad Shojafar, and Behzad Akbari. Joint Energy Efficient and QoS-Aware Path Allocation and VNF Placement for Service Function Chaining. *IEEE Transactions on Network and Service Management*, 16(1):374–388, 2019.

[18] Petra Vizarreta, Massimo Condoluci, Carmen Mas Machuca, Toktam Mahmoodi, and Wolfgang Kellerer. QoS-driven function placement reducing expenditures in NFV deployments. *IEEE International Conference on Communications*, 2017.

[19] Tianlong Yu, Seyed K Fayaz, Michael Collins, Vyas Sekar, and Srinivasan Seshan. PSI : Precise Security Instrumentation for Enterprise Networks.