# Towards Latency-aware vNF Placement on Heterogeneous Hosts at the Network Edge

Haruna Umar Adoga and Dimitrios P. Pezaros
*School of Computing Science*
*University of Glasgow,* G12 8QQ, Scotland, UK
h.adoga.1@research.gla.ac.uk, dimitrios.pezaros@glasgow.ac.uk

***Abstract*—In this paper, we investigate the optimal placement of vNFs over the distributed edge network while considering the heterogeneity of packet processing elements. Unlike previous efforts in this domain, our proposed hybrid placement scheme places vNFs on user space hosts and the network data plane to efficiently minimise end-to-end path latency and achieve faster service delivery. We formulate and solve the Hybrid vNF Placement Problem as a linear integer programming (ILP) problem and propose a hybrid placement heuristic algorithm (HYPHA), which performs an incremental placement of vNFs on heterogeneous hosts to find a near-optimal solution to the problem quickly. We evaluated our proposed solution using a simulation of real-world network topology with realistic latency conditions.**

**We show that leveraging the fast packet processing speed of the network data plane in conjunction with abundant user space resources for vNF placement yields minimal overall end-to-end latency and fulfils the placement of a diverse set of vNF requests to speed-up service delivery. Also, our results show that HYPHA can obtain near-optimal vNF mapping while incurring fewer latency threshold violations set by network operators.**

***Index Terms*—Latency minimisation, heterogeneous data plane, Hybrid VNF Placement, Network Function Virtualization, Edge Network, Network Softwarization, Virtual Network Functions.**

## I. Introduction

Service provider networks have evolved over the years with the ability to virtualise and host network functions at the network's edge – closer to the end-users [1]. This paradigm is primarily made possible due to the flexibility provided by network function virtualization (NFV) and network programmability, which allows network operators to define and place fine-grained network functions at the network edge based on customers' SLA requirements [2]. There are diverse frameworks for implementing vNFs in service provider network environments; these frameworks can either implement functions in the data plane, using programmable switching technologies such as P4 [3] or offloaded onto SmartNICs, or using user space hosts [4].

Although the network data plane offers high-speed packet processing, it is limited in key ways. It has a limited memory footprint, lacks *in-situ* packet header manipulation support, has poor state management support, and does not readily support manipulating floating-point values [5] etc. The network data plane is generally resource-constrained and thus does not accommodate vNFs with high storage or scalability requirements due to the low memory footprint [6].

Due to the diverse nature of emerging applications, a user-space-only or data-plane-only deployment does not efficiently cater for use cases such as e-healthcare, autonomous vehicles, mixed reality (MR), etc. A hybrid solution can leverage the advances from the data plane and user space. To achieve end-to-end programmability at the network edge and cater for diverse use cases, we must design placement frameworks to leverage data plane heterogeneity along a given network path. Thus, frameworks that allow us to choose the right packet processing elements are desirable based on their suitability to host a particular vNF.

As a departure from how vNFs are currently being placed, i.e., the lack of frameworks to support the seamless placement of vNFs implemented on diverse platforms, we propose a placement scheme that provisions network functions in the programmable data plane and on user space hosts to reduce packet processing overhead and overall path latency, by leveraging the peculiar benefits offered by heterogeneous packet processing elements. We present a novel ILP model for the hybrid placement of network functions in the heterogeneous network data planes and on user space hosts (§III), with the objective function of minimizing end-to-end latency and leveraging the network data plane vNF placement. We also propose a hybrid placement heuristic algorithm, HYPHA (§IV), as a way of finding a quick solution to the hybrid vNF placement problem efficiently, and present an in-depth evaluation of the proposed placement model and heuristic algorithm, using real service provider network conditions (§V).

In the remainder of the paper, we present some related work in §II. In §III, we introduce the formalization of the hybrid placement of virtual network functions on heterogeneous hosts. A hybrid placement heuristic algorithm is presented in §IV; we evaluate our proposed solution in §V and conclude the work in §VI.

## II. Background - Related Work

Some efforts in the literature considered the placement of network functions in a hybrid manner in the context of softwarised and hardware functions. Moens *et al.* [7] proposed and evaluated an ILP model using service requests that comprise hardware and software functions. The genetic algorithms proposed by Cao *et al.* [8] aim to minimize link utilization

and bandwidth consumption in hybrid NFV environments. Both works, however, did not look at the heterogeneity of hosts and the diverse nature of service requests; our approach differs in this regard – as we consider the heterogeneity of vNF hosts, which cater for emerging use cases.

Leivadeas *et al.* [9] focused on minimizing end-to-end delay and deployment cost by placing vNFs on cloud and edge servers in a hybrid context. Carpio *et al.* [10] presented a hybrid placement model to place vNFs on VMs and containers using the edge-cloud continuum. They proposed a MILP model and a heuristic solution with cost-minimization objective functions.

Shahjalal *et al.* [11] formulated a multi-objective ILP problem to handle the placement of vNFs in a hybrid infrastructure. Functions are placed on edge and cloud servers in a hybrid manner. Their approach minimises service deployment cost and maximises QoE, using user budget as a critical constraint. We place vNFs on user space hosts and leverage the speed of the data plane in distributed edge network environments. Previous work assumes a flat network topology in terms of the capabilities of hosts. We consider a hybrid scenario where vNFs are placed based on their type and complexity (e.g., delay-sensitive/delay-tolerant, stateless/stateful applications etc.) on user space and the network data plane using heterogeneous packet processing elements.

Our core objective function is minimising end-to-end latency for users requesting vNFs. We embrace the heterogeneity of packet processing elements (vNF hosts) in our approach; in particular, we leverage the processing speed of the data plane and available resources on user space hosts when placing virtual network functions. Using our solution, network operators can improve service delivery by delegating simple, time-critical operations to the data plane, leaving only more complex stateful processing for user space vNFs.

## III. Hybrid vNF placement on heterogeneous hosts - System Model

We formulate the hybrid placement model using the system parameters presented in Table I. The model outputs a mapping of vNF requests onto heterogeneous packet processing elements and the best route (in terms of lowest delay) between source and destination nodes in the hybrid topology while adhering to the user and operator-defined constraints.

We represent the capacity of data plane nodes (e.g., I/O, memory, SmartNIC CPU etc.) with the parameter $Q_j$ and vNF $n_i^o \in \mathbb{N}$ bandwidth requirements by users, with the parameter $b_k^{ij}$ (deduced from the placement requests and the network topology). The latency parameter $L_k^{ij}$ is materialized by summing all the vNF latency requirements – depending on the application class, and the latency $L_m$ of the network links $e_m \in \mathbb{E}$ along any given placement path. We capture vNF hosts requirements using the parameters $V_j$ and $R_i$ for data plane and user space packet processing elements, respectively, and use the binary decision variable $X_{ij}^k$, which has a value of 1 if a vNF $n_i^o \in \mathbb{N}$ is placed on a data plane host $d_j \in \mathbb{D}$, and a value of 0 otherwise. We also introduced a second binary decision variable $Y_{ij}^k$ which has a value of 1 if a vNF $n_i^o \in \mathbb{N}$ is placed on a user space host $h_j \in \mathbb{H}$, and uses path $p_k \in \mathbb{P}$.

Table I: System model parameters

| Network entities | Description |
|---|---|
| $\mathbb{G} = (\mathbb{H}, \mathbb{E}, \mathbb{U})$ | Physical network graph. |
| $\mathbb{H} = \{h_1, h_2, h_i, ...., h_H\}$ | A set of user space hosts. |
| $\mathbb{E} = \{e_1, e_2, e_m, ...., e_E\}$ | The physical network links. |
| $\mathbb{U} = \{u_1, u_2, u_o, ...., u_U\}$ | User traffic associated with processing NFs. |
| $\mathbb{P} = \{p_1, p_2, p_k, ...., p_P\}$ | Network paths from source to destination. |
| $\mathbb{D} = \{d_1, d_2, d_j, ...., d_D\}$ | data plane elements (e.g., switches, SmartNICs) |
| $M_k$ | The last heterogeneous host in path $p_k \in \mathbb{P}$ |
| $Q_j$ | Capacity of data plane hosts (e.g., I/O, memory, CPU, etc.) $d_j \in \mathbb{D}$. |
| $W_i$ | Supported capacity of user space hosts (e.g., I/O, memory, CPU, etc.) from $h_j \in \mathbb{H}$. |
| $C_m$ | Link capacity $e_m \in \mathbb{E}$. |
| $L_m$ | Link latency of $e_m \in \mathbb{E}$. |
| **VNF entities** | **Description** |
| $\mathbb{N} = \{n_1^1, n_2^2, n_i^o, ..., n_N^U\}$ | Network functions to be placed $n_i^o \in \mathbb{N}$ linked to user $u_o \in \mathbb{U}$. |
| $\Pi_i$ | Acceptable latency threshold for vNF $n_i^o$ |
| $V_j$ | data plane vNF host requirements (storage, memory, cpu) of $n_i^o \in \mathbb{N}$. |
| $R_i$ | user-space vNF host requirements (I/O, memory, CPU.) of vNF $n_i^o \in \mathbb{N}$. |
| **Derived entities** | **Description** |
| $L_k^{ij}$ | Latency to vNF $n_i^o$, on a host using path $p_k$. |
| $b_k^{ij}$ | Required user bandwidth to vNF $n_i^o$, using path $p_k \in \mathbb{P}$. |
| **Decision variables** | **Description** |
| $X_{ij}^k$ | To denote if a particular vNF $n_i^o$ is hosted on a data plane processing element $d_j \in \mathbb{D}$, using path $p_k \in \mathbb{P}$ |
| $Y_{ij}^k$ | To denote if a particular vNF $n_i^o$ is hosted on a user space $h_j \in \mathbb{H}$ host using path $p_k \in \mathbb{P}$ |

$$X_{ij}^k = \begin{cases} 1 \text{ if a vNF } n_i^o \text{ is on data plane } d_j, \text{ path } p_k \\ 0 \text{ otherwise} \end{cases} \tag{1}$$

$$Y_{ij}^k = \begin{cases} 1 \text{ if a vNF } n_i^o \text{ is on user-space } h_j, \text{ path } p_k \\ 0 \text{ otherwise} \end{cases} \tag{2}$$

### A. Placement problem

Consider a given network topology represented by the graph $\mathbb{G}$, with a set of users $\mathbb{U}$, a set of heterogeneous packet processing elements, and the set of individual vNFs $\mathbb{N}$, and a latency matrix $l$, our hybrid placement model should find an ideal mapping of all vNFs to heterogeneous hosts, that minimizes the total expected end-to-end latency from all users to their respective network functions. Equation (3) minimizes the expected user latency when placing network functions on heterogeneous hosts.

$$\min . \sum_{p_k \in \mathbb{P}} \sum_{d_j \in \mathbb{D}} \sum_{h_j \in \mathbb{H}} \sum_{n_i^o \in \mathbb{N}} (X_{ij}^k L_k^{ij} + Y_{ij}^k L_k^{ij}) \quad (3)$$

The above objective function is subject to the constraints presented next.

*1) Capacity constraint:* The limitations of the programmable data plane in terms of capacity to host vNFs must be adhered to when placing a vNF $n_i^o \in \mathbb{N}$, on packet processing elements $d_j \in \mathbb{D}$. We enforce this constraint by using equation (4a) and ensuring that the resource requirements $V_i$ of all vNFs placed are below $Q_i$, which represents the capability of hosts at the time of placement. In eq. (4b), we enforce a similar (but relaxed) constraint for user space nodes with vNF host requirements $R_j$ and user space capacity $W_i$ for hosts $h_j \in \mathbb{H}$.

$$\sum_{p_k \in \mathbb{P}} \sum_{n_i^o \in \mathbb{N}} X_{ij}^k (V_i) < Q_j, \forall d_j \in \mathbb{D} \quad (4a)$$

$$\sum_{p_k \in \mathbb{P}} \sum_{n_i^o \in \mathbb{N}} Y_{ij}^k (R_i) < W_i, \forall h_j \in \mathbb{H} \quad (4b)$$

*2) Placement uniqueness constraint:* In equation 5, we ensure that all possible placements of vNFs $n_i^o \in \mathbb{N}$ on nodes are unique. This constraint checks that our hybrid placement model places a network function of a particular type on exactly one host, i.e., a data plane or user space packet processing element along a given path $p_k \in \mathbb{P}$

$$\sum_{d_j \in \mathbb{D}} X_{ij}^k + \sum_{h_i \in \mathbb{H}} Y_{ij}^k = 1, \forall n_i^o \in \mathbb{N}, \forall p_k \in \mathbb{P} \quad (5)$$

*3) Link bandwidth constraint:* This constraint captures the available bandwidth of the network links $e_m \in \mathbb{E}$. It ensures that we do not overload the edge network links along a given path (Equation (6).

$$\sum_{d_j \in \mathbb{D}} X_{ij}^k b_k^{ij} + \sum_{h_j \in \mathbb{H}} Y_{ij}^k b_k^{ij} < C_m, \forall e_m \in p_k, \forall p_k \in \mathbb{P} \quad (6)$$

*4) Latency threshold:* This constraint ensures that the vNF latency threshold $\Pi_i$ is not exceeded at the time of placement for all vNFs along a path. Constraint (7) considers user latency to vNF $n_i^o \in \mathbb{N}$.

$$(\sum_{d_j \in \mathbb{D}} X_{ij}^k L_{ijk} + \sum_{h_j \in \mathbb{H}} Y_{ij}^k L_{ijk}) \sum_{p_k \in \mathbb{P}} < \Pi_i, \ \forall n_i^o \in \mathbb{N} \quad (7)$$

*5) Path validity constraint:* The network path used for placement should be valid (i.e., the path should end with the host of the network function). We enforce this requirement by using constraint (8).

$$X_{ij}^k, Y_{ij}^k = 0, n_i^o \neq M_k, \forall n_\iota^o \in \mathbb{N}, \forall p_k \in \mathbb{P}, \forall h_j \in \mathbb{H}, \forall d_j \in \mathbb{D} \quad (8)$$

**Algorithm 1** Hybrid Placement

```
1: procedure PLACEVNFS (𝔾,vNFRequests, paths)
2:     Initialise data plane capacity Q_j, W_i
3:     Initialise vNF requirements V_j, R_i
4:     for vNF ∈ vNFRequests do
5:         paths.sort(key=len)                          ▷ shortest paths
6:         placements ← [ ]
7:         while path ∈ paths do
8:             if d ∈ 𝔻 in path Λ V_j < Q_j: then
9:                 if checkvNFtype(vNF) == S then
10:                    d ∈ 𝔻 ← vNF
11:                    vNFRequests.remove(vNF)
12:                    updateResource(vNF_req, Q_j)
13:                    placements.append(vNF)        ▷ go to 25
14:                else:
15:                    h ∈ ℍ ← vNF
16:                    vNFRequests.remove(vNF)
17:                    updateResource(vNF_req, W_i)
18:                    placements.append(vNF)
19:                end if
20:            end if
21:        end while
22:        if vNFRequests then                          ▷ remaining requests
23:            go to 4
24:        end if
25:        p ← getUsedPath(placements)
26:        latency ←computeUsedPathLatency(p)
27:        return placements, latency
28:    end for
29: end procedure
```

Our placement model assumes that the network backbone is appropriately configured to provide sufficient bandwidth to handle requests from delay-sensitive applications and other application types.

## IV. Hybrid Placement Heuristic Algorithm (HYPHA)

To find an optimal placement solution, the placement problem introduced in §III requires a lot of computational time, depending on the topology size (e.g., a small topology with 27 edge nodes takes $\sim$ 15 minutes to place about 100 network functions optimally and compute any latency violations etc.). This time grows linearly as the size of the topology and complexity of network functions grows, thus not making it ideal for large-scale network topologies – other factors, such as available computing resources and the optimisation solver used, also contribute to the overall time. Furthermore, the optimisation model assumes the stability of network-wide information at the time of placement, which is often unrealistic due to the dynamic nature of edge network environments characterized by frequent changes in network conditions (e.g., path latency between edge nodes).

Unlike the optimal placement model in §III, the heuristic algorithm performs an incremental placement of vNFs, to find a near-optimal solution (i.e., with a relaxed optimality requirement) to the problem, which also gives network operators the flexibility to prioritise placements. The algorithm makes locally optimal decisions based on available resources and paths at each step. Requests are classified in a simple step after checking the resource requirement of each vNF (i.e., to decide if a given vNF in a set of placement requests is stateless or stateful). In Algorithm 1, `vNFRequests` consists of stateless and stateful functions $S$ and $S'$, respectively. We implement our prototype network infrastructure $\mathbb{G}$, using the Python NetworkX[1] Library, and use the `shortest_path()` function to compute all possible network paths for placement. Data plane processing elements (e.g., programmable switches or SmartNICs) along a given path are checked with the available resource capacity $Q_j$, ensuring the constraints are respected. Note that this condition is relaxed for user space hosts based on our assumption of the abundance of resources on such hosts.

We use the `checkvNFtype()` function to ascertain if a particular vNF is a stateless or stateful function before assigning it to the right packet processing element. User space hosts $h \in \mathbb{H}$ handle the requests not fulfilled by the data plane due to the lack of available resource capacity. This process repeats until all functions in `vNFRequests` are successfully assigned. The most up-to-date placements are returned, with the overall latency of the used path. Depending on the service provider's policies, our placement scheme can easily be extended to accommodate specific needs (e.g., prioritising paths with the lowest latency for premium subscribed users).

## V. Evaluation

We describe our evaluation environment in §V-A. To show the benefits of running vNFs on heterogeneous hosts, we evaluate two placement scenarios in §V-B, i.e., user-space-only and hybrid vNF placements, particularly to ascertain the average latency of users to their vNFs. We compare the performance of our heuristic algorithm and the optimisation model in §V-C (in terms of objective functions), using three main scenarios, i.e., data-plane-only, hybrid and user-space-only placements. To gain insights on the percentage of vNF placement requests fulfilled, we evaluated in §V-D. Finally, we again compared the performance of our heuristic to the optimisation model by considering a scenario with dynamic latency behaviour in §V-E, particularly in handling latency violations.

### A. Evaluation environment design

We used a real-world network topology from topology zoo[2]. We modelled the hybrid network topology by capturing the peculiar characteristics and constraints of the network data plane and introducing compute capacities on edge nodes, representing our network data plane. Table II shows some of the characteristics of representative vNF use cases [12], [13], with latency thresholds ranging between 5 to 80 ms.

Table II: Common use case latency characteristics

| Example application | Latency class | Latency bound |
|---|---|---|
| Autonomous vehicles | Real-time | 5 ms |
| Smart Grids | Low (near real-time) | 10 ms |
| Mixed Reality (AR & VR) | Low (soft real-time) | 10 - 15 ms |
| Service Orchestration | Near real-time | 20 ms |
| Monitoring applications | Non real-time | 80 ms |

The nationwide Jisc NREN UK backbone network topology was used by adjusting the bandwidth parameter and topology size to fit edge DC environments and our use case of having heterogeneous nodes. The bandwidth limit between vNF hosts is set to 100 Mbit/s or 1000 Mbit/s, depending on vNF bandwidth requirements. Compute capacity is provisioned on edge devices, and we assume the data plane to have finite computing resources (i.e., I/O, memory, storage etc.) with ample resources on user space hosts.

The values of link latency are sampled from a Gamma distribution using the parameters (k = 2.2, $\theta$ = 0.22). Thus, a time series is created by sampling from this distribution (with a value of 100 in our experiment), representing the link latency values over time. This time series reflects periods of low latency and occasional high latency spikes – a typical pattern in edge/WAN networks due to traffic rerouting, avoiding congested paths etc. We modelled these values from real-world latencies observed from the Energy Sciences Network (ESnet) PerfSonar[3] measurements.

### B. User-space-only and hybrid placement latency

In this evaluation, we show the user latencies of placing network functions in two main scenarios, i.e., user-space-only and hybrid. We implemented the placement model presented in §III using a commercial optimization solver, Gurobi[4]. Edge users were assigned a typical last hop latency of 2 ms to their respective vNFs. We allocated vNFs to each user with different latency requirements, a typical scenario for vNFs with user space and hybrid requirements. We allocate computing resources to our topology's data plane and user space nodes and generate 500 vNF placement requests.

As depicted in Figure 1, placing the requested functions using the hybrid topology yields the lowest latency, which deteriorates as the number of vNFs grows ($\sim$ 300 vNFs). We attribute this phenomenon to the low resource capacity of the available data plane nodes closer to the users along a chosen network path, which results in the use of user space hosts. The user-space-only placement results in much higher latency, although this comes with the advantage of fulfilling more diverse vNF placement requests (demonstrated in §V-D). Note that the hybrid placement scenario presents a *sweet spot* in terms of reduced latency, as it

[1]https://networkx.org
[2]http://www.topology-zoo.org/
[3]http://ps-dashboard.es.net/maddash-webui/
[4]https://www.gurobi.com/

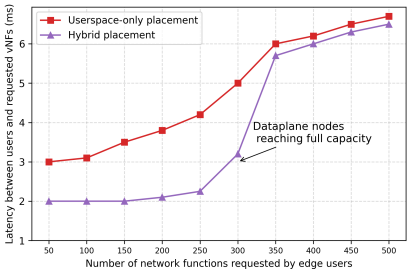


Figure 1: Average user to vNF latency in user-space-only and hybrid optimisation placement scenarios.

presents a trade-off between the resource-constrained nature of the data plane and the high latency of user space nodes.

### C. Optimisation and heuristic objective functions

In this evaluation, we take note of the values of the objective functions for all placement scenarios, for the optimisation and heuristic placements, focusing on analysing how the overall objective function (end-to-end latency) varies in each scenario. We maintained the same number of requests as with previous experiments (500 vNFs) and focused on evaluating the effect of capacity constraints. As depicted in Figure 2, we varied the resource capacity of the edge nodes in our topology – specifically, network functions were assigned using data-plane-only, hybrid and user-space-only scenarios. This allowed us to assess the model's performance and heuristic solutions under different resource constraints.

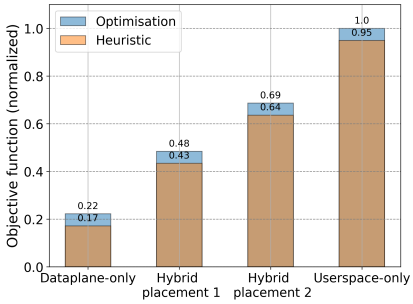


Figure 2: Heuristic and optimisation objective functions for different placement scenarios.

In the first hybrid placement scenario, data plane nodes can host 50% of the total vNF placement requests; we reduced this number in the second hybrid placement scenario to 30% to observe how it affects the objective function and finally to 0 (in the user-space-only scenario).

We noticed that the data-plane-only scenario yields the lowest end-to-end overall latency, about 78% better than the user-space-only placement scenario. The best-case scenario in hybrid placements is the first hybrid placement, which is 48% better than the user-space-only placement. We attribute this phenomenon to the preference for data plane edge nodes over user space nodes, particularly in hybrid placement scenarios. The hybrid scenario also provides the core advantage of encompassing edge nodes with heterogeneous capabilities – we can thus push more delay-sensitive applications down to the network data plane to be processed quickly. The heuristic placement yields near-optimal objective function values in all scenarios, which is also about 20X faster.

### D. Number of placement requests fulfilled

In this evaluation, we focus on the heterogeneity of the placed vNFs, while keeping other components of our evaluation the same for consistency. We generated 500 user vNF placement requests using a Gamma distribution with k=1.6 and $\theta$ = 0.5 for stateless functions and k=2.5 and $\theta$ = 1 for stateful functions. The Gamma probability distribution captures the nature of commonly used network functions, where many are simple stateless functions deployable in the data plane [14]. Thus, the heavy-tailed distribution values for stateless functions in our evaluations represent this phenomenon (i.e., a high occurrence of such functions).

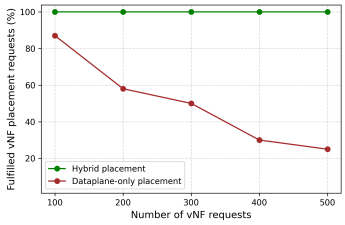


Figure 3: Comparison of the number of fulfilled vNF placement requests with different numbers of user placement requests.

As depicted in Figure 3, the hybrid scenario maintains many fulfilled requests, attributed to the support for *all* types of network functions. Although the number of fulfilled placement requests is lower in the data-plane-only scenario, it offers reduced latency (as demonstrated in §V-B). More than 60% of received requests are processed in the data plane, particularly for $<$ 300 requests; this number drops as the number of placement requests grows, and fewer stateless network functions are found in the requests.

### E. Dynamic latency violations

The dynamic nature of mobile edge users often results in changes in the temporal latency values over time, i.e., due to the mobility of users. Note that vNFs have a latency threshold value $\Pi_i > 0$ (defined in eq. (7)), which is set by network operators (e.g., based on latency SLA between providers and customers) – violating this set latency threshold could result in application performance degradation.

We analyze the number of latency violations over a time series (introduced in §V-A); a time instance in the series could be, e.g., 3 minutes in real-world edge network environments. We set the value of $\Pi_i$ at 30 ms for this evaluation (note that network operators can define this threshold based on the agreed SLA). By

incorporating a time series component, we employed a hybrid placement scenario that expands upon evaluating the heuristic and optimal objective functions presented in §V-C. Specifically, we track latency violations at each time instance.

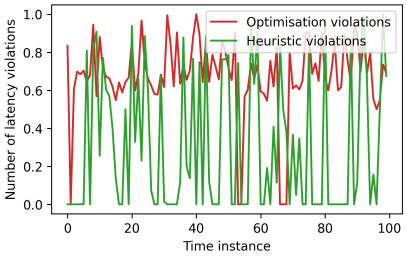


Figure 4: Heuristic and optimisation latency violations performance.

In Figure 4, we show the latency violations over the time series to evaluate the ILP model and heuristic performance and provide service providers with valuable insights into application latency violations. Note that the hybrid placement heuristic presents fewer latency violations – we attribute this to the incremental placement mechanism employed by the heuristic, which prioritises the available shortest network paths containing data plane processing elements at the time of placement, i.e., unlike the ILP model, which tries to compute the end-to-end latency by considering the *global* network parameters, thus leading to a higher number of latency violations at several time instances.

## VI. Conclusions

In this work, we argue that to achieve faster service delivery and cater for diverse vNF requirements in emerging edge network environments, in addition to edge vNF placement, network operators can leverage the network data plane for deploying virtual network functions. We investigated the diverse nature of user vNF placement requests based on emerging use cases such as e-healthcare, autonomous vehicles etc. We proposed a hybrid vNF placement optimisation model and heuristic that leverages the low-latency, high-speed feature of the network data plane and the abundant resources on user space hosts.

The heuristic solution is designed to cater to real-world networks' dynamic nature, which makes using optimisation models challenging; thus, HYPHA can quickly obtain near-optimal vNF mapping while incurring fewer latency threshold violations set by network operators (§V-C and §V-E). Through extensive evaluations, we demonstrate that employing a hybrid deployment scheme that leverages the processing capability of the network data plane yields minimal user-to-vNF latency and overall end-to-end path latency, thus fulfilling the placement of a diverse set of vNF requests from emerging use cases. Network operators can leverage the high-speed, low-latency feature of data plane packet processing elements for hosting delay-sensitive applications and improving service delivery for subscribed users. Also, in addition to emerging edge use cases, our placement solution can be adapted to efficiently place network functions in core network infrastructure while leveraging the heterogeneity of servers. Further research includes supporting live vNF placement rescheduling in a hybrid edge network topology to leverage the heterogeneous network data plane capabilities.

## Acknowledgments

This work was supported in part by the UK Engineering and Physical Sciences Research Council (EPSRC) grant EP/N033957/1, the PETRAS National Centre of Excellence for IoT Systems Cybersecurity, which the UK EPSRC has funded under grant number EP/S035362/1, and the Petroleum Technology Development Fund (PTDF) Nigeria, grant 1563/19.

## References

[1] D. Harris and D. Raz, "Dynamic vnf placement in 5g edge nodes," in *2022 IEEE 8th International Conference on Network Softwarization (NetSoft)*. IEEE, 2022, pp. 216–224.

[2] H. U. Adoga and D. P. Pezaros, "Network function virtualization and service function chaining frameworks: A comprehensive review of requirements, objectives, implementations, and open research challenges," *Future Internet*, vol. 14, no. 2, p. 59, 2022.

[3] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese, and D. Walker, "P4: Programming protocol-independent packet processors," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 3, pp. 87–95, 2014.

[4] H. U. Adoga, Y. Elkhatib, and D. P. Pezaros, "On the performance benefits of heterogeneous virtual network function execution frameworks," in *2022 IEEE 8th International Conference on Network Softwarization (NetSoft)*. IEEE, 2022, pp. 109–114.

[5] K. A. Simpson, R. Cziva, and D. P. Pezaros, "Seir: Dataplane assisted flow classification using ml," 2020.

[6] D. Kim, Y. Zhu, C. Kim, J. Lee, and S. Seshan, "Generic external memory for switch data planes," in *Proceedings of the 17th ACM Workshop on Hot Topics in Networks*, 2018, pp. 1–7.

[7] H. Moens and F. De Turck, "Vnf-p: A model for efficient placement of virtualized network functions," in *10th international conference on network and service management (CNSM) and workshop*. IEEE, 2014, pp. 418–423.

[8] J. Cao, Y. Zhang, W. An, X. Chen, Y. Han, and J. Sun, "Vnf placement in hybrid nfv environment: Modeling and genetic algorithms," in *2016 IEEE 22nd International Conference on Parallel and Distributed Systems (ICPADS)*. IEEE, 2016, pp. 769–777.

[9] A. Leivadeas, G. Kesidis, M. Ibnkahla, and I. Lambadaris, "Vnf placement optimization at the edge and cloud," *Future Internet*, vol. 11, no. 3, p. 69, 2019.

[10] F. Carpio, W. Bziuk, and A. Jukan, "On optimal placement of hybrid service function chains (sfcs) of virtual machines and containers in a generic edge-cloud continuum," *arXiv preprint arXiv:2007.04151*, 2020.

[11] M. Shahjalal, N. Farhana, P. Roy, and M. A. Razzaque, "Qoe aware optimal deployment of virtual network functions in 5g hybrid cloud," in *2021 3rd International Conference on Sustainable Technologies for Industry 4.0 (STI)*. IEEE, 2021, pp. 1–6.

[12] Y. Rafique, A. Leivadeas, and M. Ibnkahla, "An iot-aware vnf placement proof of concept in a hybrid edge-cloud smart city environment," in *2022 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, 2022, pp. 1395–1400.

[13] R. Cziva, C. Anagnostopoulos, and D. P. Pezaros, "Dynamic, latency-optimal vnf placement at the network edge," in *Ieee infocom 2018-ieee conference on computer communications*. IEEE, 2018, pp. 693–701.

[14] X. Chen, H. Liu, D. Zhang, Z. Meng, Q. Huang, H. Zhou, C. Wu, X. Liu, and Q. Yang, "Automatic performance-optimal offloading of network functions on programmable switches," *IEEE Transactions on Cloud Computing*, 2022.