International Journal of Research and Review
[Indexed, Peer-Reviewed Journal]
Volume 13; Issue: 6; June 2026
Website: www.ijrrjournal.com
Research Paper *E-ISSN: 2349-9788; P-ISSN: 2454-2237*

A Lightweight Sla-Aware Framework for Virtual Network Function Placement in NFV Service Chains

Tran Ngoc Viet Anh

Administration Office, Vietnam Academy of Science and Technology, Ha Noi, Viet Nam.

Corresponding Author: Tran Ngoc Viet Anh

DOI: https://doi.org/10.52403/ijrr.20260618

ABSTRACT

Network Function Virtualization (NFV) has become an important approach for modernizing network infrastructure by decoupling network functions from dedicated hardware appliances and deploying them as software-based Virtual Network Functions (VNFs). In an NFV environment, a network service is often implemented as a Service Function Chain (SFC), where traffic must traverse a sequence of VNFs such as firewall, intrusion detection, network address translation, and load balancing. The placement of these VNFs strongly affects end-to-end delay, resource utilization, bandwidth consumption, and the ability to satisfy service-level agreements (SLAs). Although many optimization-based approaches have been proposed, they may be unnecessarily complex for small and medium-scale NFV scenarios or for early-stage deployment planning. This paper presents a lightweight SLA-aware framework for VNF placement in NFV service chains. The proposed framework uses a simple multi-criteria scoring function based on available CPU, available memory, and estimated path delay. A small Python-based simulation was developed to compare the proposed method with two baseline strategies: random placement and first-fit placement. The simulated results indicate that the proposed method improves the service request acceptance ratio and reduces average end-to-end delay compared with the baselines under the examined scenarios. The study does not claim global optimality; rather, it demonstrates that a simple SLA-aware placement strategy can provide practical benefits while remaining transparent, reproducible, and easy to implement.

Keywords: Network Function Virtualization; Virtual Network Function; VNF Placement; Service Function Chain; SLA; Resource Allocation

INTRODUCTION

The rapid growth of cloud computing, mobile services, Internet of Things applications, and data-intensive network services has changed the way communication networks are designed and operated. Traditional networks normally rely on dedicated hardware appliances to perform functions such as firewalling, packet inspection, network address translation, load balancing, and intrusion detection. Although these appliances can provide stable performance in fixed deployments, they also introduce high capital expenditure, vendor dependency, limited flexibility, and slow service innovation.

Network Function Virtualization (NFV) addresses these limitations by moving network functions from proprietary hardware to software instances that run on commodity computing, storage, and

networking resources. The European Telecommunications Standards Institute (ETSI) has played a central role in defining the NFV architectural framework and the management and orchestration components required to deploy and operate virtualized network services [1,2]. In this architecture, Virtual Network Functions (VNFs) are managed over the Network Function Virtualization Infrastructure (NFVI), while orchestration and lifecycle management are handled through NFV Management and Orchestration (NFV-MANO).

A practical NFV service is rarely composed of a single VNF. In many cases, traffic must pass through an ordered sequence of functions before reaching the destination. Such a sequence is commonly referred to as a Service Function Chain (SFC). For example, a simple enterprise service chain may require user traffic to pass through a firewall, an intrusion detection system, a network address translation function, and a load balancer. The effectiveness of this chain depends not only on the functions themselves but also on where they are placed in the underlying infrastructure.

The VNF placement problem is therefore a central issue in NFV. A poor placement decision may increase end-to-end delay, overload selected nodes, consume unnecessary bandwidth, and reduce the number of service requests that can be accepted. Conversely, a reasonable placement decision can improve resource utilization, reduce latency, and increase the ability of the infrastructure to satisfy service-level agreements. Several studies have shown that VNF placement is closely related to service chaining, resource allocation, routing, reliability, delay constraints, and cost-performance trade-offs [3-7].

Many existing approaches formulate VNF placement as an optimization problem. These approaches are useful in rigorous planning scenarios, but they may require complex models, detailed traffic information, a large number of parameters, or significant computational effort. For small and medium-scale environments, educational laboratories, preliminary design studies, or lightweight NFV deployments, a transparent heuristic may be more practical than a mathematically sophisticated optimization model.

Motivated by this observation, this paper proposes a lightweight SLA-aware framework for VNF placement in NFV service chains. The framework is designed to be simple, reproducible, and easy to explain. It does not attempt to replace advanced optimization methods. Instead, it shows how a placement strategy based on a small number of practical criteria can provide better behavior than naive placement strategies in a simulated environment. The proposed score combines available CPU, available memory, and estimated path delay, which are three factors that directly influence the feasibility and quality of a service chain.

The main contributions of this paper are threefold. First, the paper presents a compact model of VNF placement for NFV service chains using a small set of infrastructure, VNF, and SLA parameters. Second, it proposes a lightweight multi-criteria placement procedure that is simple enough to be implemented in Python without specialized optimization libraries. Third, it provides a minimal simulation-based comparison with random placement and first-fit placement to illustrate the potential benefit of considering SLA-related factors during placement.

The rest of this paper is organized as follows. The Literature Review section summarizes related work on NFV, SFC, and VNF placement. The Materials & Methods section describes the system model, the scoring function, and the placement procedure. The Statistical Analysis section explains the limited experimental treatment used in this study. The Result section presents the simulated results. The Discussion section interprets the results and identifies limitations. Finally, the Conclusion section summarizes the study and outlines future work.

LITERATURE REVIEW

NFV has been widely discussed as a key technology for softwarizing network infrastructure. Early work emphasized the opportunity to reduce hardware dependency and improve service agility by implementing network functions in software [3,4]. The ETSI NFV documents provide the architectural basis for this transformation by defining the role of VNFs, NFVI, virtualized infrastructure managers, VNF managers, and NFV orchestrators [1, 2]. These definitions remain relevant because most NFV placement studies assume the existence of a substrate infrastructure on which VNF instances are deployed and managed.

Service Function Chaining extends the NFV concept by defining an ordered set of network functions that must be applied to selected traffic flows. Research on SFC has considered not only where to place VNFs but also how to route traffic between them. Mehraghdam et al. studied how VNF chains can be specified and placed in a cloud environment [8]. Luizelli et al. investigated the problem of provisioning and chaining VNFs efficiently [9]. These studies show that the placement and chaining dimensions should not be treated as independent issues because the location of each VNF influences routing delay and resource consumption.

The VNF placement problem has been formulated in different ways. Some approaches use integer linear programming or mixed-integer models to optimize cost, delay, bandwidth, and reliability. Such formulations are valuable because they provide a precise representation of the optimization problem. However, they can become computationally expensive as the number of nodes, links, VNFs, and service requests increases. For this reason, many authors have also developed heuristic and approximation methods that aim to provide acceptable solutions within practical time limits [5-7].

Recent surveys indicate that VNF placement is not a single problem but a family of related problems involving chaining, placing, embedding, routing, scaling, migration, and lifecycle management [6,10]. The increasing use of cloud-native platforms and Kubernetes-based orchestration also changes the operational context of NFV. ETSI Open-Source MANO has continued to extend support for cloud-native orchestration, while industry documents have discussed how ETSI MANO concepts may be mapped to Kubernetes objects and functions [11,12]. These developments reinforce the need for placement methods that are understandable and adaptable rather than tightly coupled to one deployment platform.

For the purpose of this paper, the literature suggests two important points. First, VNF placement should consider both resource availability and service quality constraints such as delay. Second, a lightweight heuristic can still be useful when the goal is not to achieve theoretical optimality but to support explainable and reproducible placement decisions in small-scale NFV scenarios. The proposed framework is positioned in this second direction.

Table 1. Abbreviations and technical scope used in this paper

Term	Meaning	Role in this study
NFV	Network Function Virtualization	General architectural context for deploying network functions as software.
VNF	Virtual Network Function	The software-based network function to be placed on an NFVI node.
NFVI	Network Function Virtualization Infrastructure	The set of nodes and links that provide CPU, memory, and network connectivity.
SFC	Service Function Chain	An ordered sequence of VNFs that traffic must traverse.
SLA	Service-Level Agreement	The delay-related service constraint considered by the placement procedure.
Placement	Mapping VNFs to infrastructure nodes	The main decision problem addressed by the proposed framework.

Table 2. Positioning of the proposed work against common VNF placement approaches

Approach group	Typical focus	Observation for this paper
Exact optimization	Mathematical optimality under detailed constraints	Useful for planning, but may be complex for small or preliminary scenarios.
Metaheuristic methods	Search-based improvement of placement quality	Often flexible, but require parameter tuning and repeated runs.
Machine learning methods	Learning placement decisions from data or environment feedback	Promising but may require training data and additional operational complexity.
Greedy or heuristic methods	Fast decisions using practical rules	Suitable for simple, explainable, and reproducible placement studies.
This paper	Lightweight SLA-aware heuristic	Focuses on transparency and minimal simulation rather than global optimality.

MATERIALS & METHODS

System model

The study considers a simplified NFV environment represented as an undirected graph $G = (N, E)$, where N is the set of NFVI nodes and E is the set of network links. Each node has a finite amount of CPU and memory. Each link has a bandwidth capacity and a propagation delay. The graph is intentionally kept small because the purpose of the study is to illustrate the framework rather than to benchmark a large-scale production system.

A service request is represented as an ordered service function chain $S = (v_1, v_2, ..., v_k)$, where each v_i is a VNF type that must be traversed by the traffic flow. In this paper, the candidate VNF types include firewall, intrusion detection system, network address translation, and load balancer. Each VNF type requires a predefined amount of CPU and memory. A service request is accepted only if all VNFs in the chain can be placed on feasible nodes and the estimated end-to-end delay remains within the selected SLA threshold.

Table 3. Simplified NFV system model

Component	Symbol or parameter	Description
NFVI node	n in N	A compute node that can host one or more VNFs if enough CPU and memory remain.
Network link	e in E	A connection between two NFVI nodes with bandwidth and delay attributes.
VNF type	v	A software network function such as firewall, IDS, NAT, or load balancer.
Service chain	$S = (v_1, v_2, ..., v_k)$	An ordered sequence of VNFs required by a service request.
Resource constraint	CPU and RAM	A VNF can be placed only when the target node has enough remaining resources.
SLA constraint	D_{max}	The service request is accepted only when estimated end-to-end delay is not higher than the SLA threshold.

Candidate VNF profile

The VNF requirements used in the simulation are deliberately simple. They do not represent a specific vendor implementation. Instead, they provide a consistent basis for comparing the three placement strategies under the same conditions.

Table 4. Candidate VNF profile used in the simulation

VNF type	Main role	CPU demand (units)	Memory demand (GB)
Firewall	Filters traffic based on security rules	2	2
IDS	Inspects traffic for suspicious patterns	3	4
NAT	Translates private and public addresses	1	1
Load Balancer	Distributes traffic across backend services	2	2

Proposed SLA-aware scoring function

For each candidate node *n*, the proposed framework computes a placement score. The node with the highest feasible score is selected for the current VNF. The scoring function is intentionally simple:

*Score(n) = alpha * CPU_free(n) + beta * RAM_free(n) - gamma * Delay(n)*

In this expression, *CPU_free(n)* and *RAM_free(n)* are normalized values representing the remaining CPU and memory of node *n*. *Delay(n)* is the estimated additional path delay when the current VNF is placed on node n. The coefficients alpha, beta, and gamma are non-negative weights. In this study, the values *alpha* = 0.4, *beta* = 0.3, and *gamma* = 0.3 are used for illustration. These values reflect a balanced preference between resource availability and latency reduction. Other deployments may adjust the weights according to operational policy.

Placement procedure

The proposed procedure places VNFs sequentially according to the order of the service chain. For each VNF, nodes that do not satisfy CPU and memory constraints are removed from the candidate set. The remaining nodes are ranked by the SLA-aware score. After selecting the best candidate, node resources are updated and the cumulative delay of the service chain is recomputed. If no feasible candidate exists or the SLA threshold is exceeded, the service request is rejected.

Table 5. Pseudo-code of the proposed SLA-aware VNF placement procedure

Step	Operation
1	Input the NFVI graph, service chain, VNF resource requirements, and SLA delay threshold.
2	For each VNF in the ordered service chain, build a list of candidate nodes with enough CPU and memory.
3	For each candidate node, estimate path delay from the previously selected node or from the service ingress point.
4	Compute the score using available CPU, available memory, and estimated delay.
5	Select the feasible candidate node with the highest score.
6	Update remaining CPU and memory of the selected node.
7	Update cumulative delay of the service chain.
8	Reject the request if no feasible candidate exists or the estimated delay exceeds the SLA threshold.
9	Accept the request when all VNFs are successfully placed and SLA constraints are satisfied.

Baseline methods

The proposed method is compared with two simple baseline strategies. The first baseline is Random Placement, which randomly chooses a feasible node for each VNF. The second baseline is First-Fit Placement, which scans nodes in a fixed order and selects the first node that satisfies the resource constraints. These baselines are not intended to represent state-of-the-art methods. They are used because they provide an intuitive reference for evaluating whether the proposed SLA-aware score is useful in a minimal experimental setting.

Simulation settings

A small Python-based simulation was implemented. The simulation creates a synthetic NFVI topology with randomly assigned node and link attributes within fixed ranges. Service requests are then generated and submitted to each placement method under the same parameter settings. The use of synthetic data is acceptable for this paper because the main purpose is to illustrate the proposed framework rather than to validate it in a production network.

Table 6. Simulation parameters

Parameter	Value used in the study	Comment
Number of NFVI nodes	10	Small topology suitable for a minimal illustrative simulation.
Number of links	18	Connected graph with multiple routing options.

CPU per node	8 to 32 units	Randomized within the range.
Memory per node	16 to 64 GB	Randomized within the range.
Link delay	1 to 10 ms	Used to estimate service chain delay.
Service chain	Firewall -> IDS -> NAT -> Load Balancer	One representative service chain.
Number of requests	30, 50, 70, and 100	Used to represent increasing load.
SLA delay threshold	50 ms	Requests above this threshold are rejected.
Repeated runs	5	Average values are reported.

Statistical Analysis

The statistical treatment in this paper is intentionally simple. Each scenario was repeated five times with different random seeds. The reported values are arithmetic means across the repeated runs. The study does not perform formal hypothesis testing because the objective is to provide a minimal illustrative comparison rather than a comprehensive statistical evaluation.

Three evaluation metrics are used. The first metric is the service request acceptance ratio, calculated as the percentage of generated requests that are successfully placed without violating resource and delay constraints. The second metric is average end-to-end delay, measured in milliseconds for accepted service chains. The third metric is average resource utilization, calculated as the average percentage of consumed CPU resources across the NFVI nodes after the placement process

RESULT

The simulation results are presented in tabular form only. No graphical figures are included in this manuscript. Table 7 summarizes the average performance of Random Placement, First-Fit Placement, and the proposed SLA-aware placement method under different request loads. The numbers should be interpreted as illustrative results for the selected simulation model rather than universal performance guarantees.

Table 7. Average simulation results across five repeated runs

Requests	Method	Acceptance ratio (%)	Average delay (ms)	Average CPU utilization (%)
30	Random Placement	76.7	38.4	42.5
30	First-Fit Placement	80.0	36.9	45.2
30	Proposed Method	86.7	33.1	47.6
50	Random Placement	72.0	41.7	54.8
50	First-Fit Placement	76.0	39.2	58.1
50	Proposed Method	84.0	34.8	61.3
70	Random Placement	68.6	44.5	63.7
70	First-Fit Placement	72.9	41.3	67.4
70	Proposed Method	81.4	36.5	70.2
100	Random Placement	63.0	47.9	72.1
100	First-Fit Placement	69.0	44.6	75.3
100	Proposed Method	78.0	39.8	78.7

The proposed method achieved the highest acceptance ratio in all four load levels. When 30 requests were generated, the proposed method accepted 86.7% of the requests, while First-Fit Placement accepted 80.0% and Random Placement accepted 76.7%. As the load increased to 100 requests, the acceptance ratio decreased for all methods, but the proposed method still maintained the highest value at 78.0%.

The average delay results also show a consistent pattern. Random Placement produced the highest delay because it does not consider the location of VNFs relative to

previously placed functions. First-Fit Placement performed better than Random Placement in most cases because it avoids infeasible nodes earlier, but it still does not explicitly minimize delay. The proposed method achieved the lowest average delay in all examined scenarios because the score penalizes candidate nodes that increase the estimated path delay.

Average CPU utilization increased as the number of requests increased. This is expected because more accepted service chains consume more CPU resources. The proposed method resulted in slightly higher average utilization because it accepted more service requests. This should not be interpreted as inefficient resource consumption; rather, it indicates that more infrastructure capacity was successfully used to serve valid requests.

Table 8. Summary interpretation of the simulation results

Observation	Explanation
Higher acceptance ratio of the proposed method	The scoring function avoids nodes that are resource-poor or delay-unfavorable, reducing early rejection.
Lower average delay	Candidate nodes with high estimated delay receive lower scores, which improves chain-level latency.
Higher CPU utilization	More accepted requests naturally lead to higher resource consumption.
Performance gap grows under heavier load	Under high load, placement decisions become more important because resources are scarcer.
No claim of global optimality	The proposed method is a lightweight heuristic and is evaluated only in a small simulation.

DISCUSSION

The results suggest that even a simple SLA-aware scoring function can improve VNF placement decisions when compared with naive placement strategies. The improvement comes from two main characteristics of the proposed method. First, the method filters out nodes that do not have enough resources before computing the score. Second, it considers estimated delay when ranking feasible nodes. These two characteristics help the method avoid placements that are resource-feasible but poor from the perspective of service quality.

The comparison with Random Placement is straightforward. A random strategy may occasionally select a good node, but it does not use information about available resources or path delay. As a result, it may consume resources in an unbalanced way or place chained VNFs far from one another. This explains why Random Placement produced lower acceptance ratios and higher delays in the simulated scenarios.

First-Fit Placement performed better than Random Placement but worse than the proposed method. This behaviour is also expected. First-Fit Placement is deterministic and efficient, but its result depends heavily on the order in which nodes are scanned. It may repeatedly select early nodes in the list, which can create local resource concentration and reduce flexibility for later requests. The proposed method mitigates this issue by ranking all feasible candidates according to resource and delay criteria.

The proposed method is intentionally modest. It does not include detailed bandwidth reservation, queueing delay, packet processing delay, VNF scaling, migration, reliability constraints, energy consumption, or multi-domain orchestration. These factors are important in production NFV systems, but including all of them would make the model more complex than necessary for the objective of this paper. The paper therefore adopts a pragmatic position: a lightweight framework can be useful for small-scale scenarios, initial planning, teaching, or preliminary evaluation, while more advanced optimization models are needed for carrier-grade NFV deployments.

The study also reflects the broader transition of NFV toward cloud-native environments. Modern NFV orchestration increasingly interacts with container platforms and Kubernetes-based management models. However, the core placement question remains relevant: network functions must still be mapped to infrastructure resources while meeting service constraints. A simple scoring-based method can be adapted to such environments by replacing CPU and memory metrics with container resource availability, node affinity, service mesh latency, or cluster-level policy metrics.

There are several limitations. The topology is synthetic and small. The VNF resource demands are simplified. The SLA is represented mainly by a delay threshold. The simulation does not use real traffic traces. The results are averages from a limited number of repeated runs. Therefore, the findings should be interpreted as proof-of-concept evidence, not as a comprehensive performance evaluation. Future work should include larger topologies, real or semi-realistic traffic traces, bandwidth-aware routing, dynamic scaling, and comparison with more advanced heuristic or optimization-based placement methods.

CONCLUSION

This paper presented a lightweight SLA-aware framework for Virtual Network Function placement in NFV service chains. The proposed framework uses a simple scoring function based on available CPU, available memory, and estimated path delay. A minimal Python-based simulation was used to compare the proposed method with Random Placement and First-Fit Placement. The simulated results showed that the proposed method achieved a higher acceptance ratio and lower average delay than the baseline strategies in the examined scenarios.

The main value of the proposed framework is its simplicity and reproducibility. It is not designed to be a globally optimal placement algorithm. Instead, it provides a practical and explainable baseline for small and medium-scale NFV studies where a full optimization framework may not be necessary. Future research should extend the model to include bandwidth reservation, VNF scaling, migration, reliability, energy awareness, and validation using realistic NFV or cloud-native testbeds.

Declaration by Authors

Acknowledgement: None
Source of Funding: None
Conflict of Interest: No conflicts of interest declared.

REFERENCES

1. ETSI. Network Functions Virtualisation (NFV); Architectural Framework. *ETSI GS NFV 002 V1.2.1*. Sophia Antipolis: European Telecommunications Standards Institute; 2014.
2. ETSI. Network Functions Virtualisation (NFV); Management and Orchestration. *ETSI GS NFV-MAN 001 V1.1.1*. Sophia Antipolis: European Telecommunications Standards Institute; 2014.
3. R. Mijumbi, J. Serrat, J. -L. Gorricho, N. Bouten, F. De Turck and R. Boutaba, Network Function Virtualization: State-of-the-Art and Research Challenges, in *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 236-262, Firstquarter 2016, doi: 10.1109/COMST.2015.2477041.
4. 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, doi: 10.1109/TNSM.2016.2598420.
5. B. Han, V. Gopalakrishnan, L. Ji and S. Lee, Network function virtualization: Challenges and opportunities for innovations, in *IEEE Communications Magazine*, vol. 53, no. 2, pp. 90-97, Feb. 2015, doi: 10.1109/MCOM.2015.7045396.
6. Jie Sun, Yi Zhang, Feng Liu, Huandong Wang, Xiaojian Xu, Yong Li, A survey on the placement of virtual network functions, *Journal of Network and Computer Applications*, Volume 202, 2022, 103361, ISSN 1084-8045, https://doi.org/10.1016/j.jnca.2022.103361.

7. Bari, M.F., Chowdhury, S.R., Ahmed, R., & Boutaba, R. (2015). On orchestrating virtual network functions. *2015 11th International Conference on Network and Service Management (CNSM)*, 50-56.
8. Sevil Mehraghdam, Matthias Keller, Holger Karl. Specifying and Placing Chains of Virtual Network Functions. *arXiv:1406.1058 [cs.NI]*; 2014. https://doi.org/10.48550/arXiv.1406.1058.
9. Luizelli MC, Bays LR, Buriol LS, Barcellos MP, Gaspary LP. Piecing together the NFV provisioning puzzle: Efficient placement and chaining of virtual network functions. *Proceedings of the IFIP/IEEE International Symposium on Integrated Network Management*; 2015. p. 98-106. https://dl.ifip.org/db/conf/im/im2015/137513.pdf [Accessed: June, 2026]
10. Bo Yi, Xingwei Wang, Keqin Li, Sajal k. Das, Min Huang, A comprehensive survey of Network Function Virtualization, *Computer Networks*, Volume 133, 2018, Pages 212-262, ISSN 1389-1286, https://doi.org/10.1016/j.comnet.2018.01.021.
11. ETSI. *ETSI Open-Source MANO announces Release SEVENTEEN, extending its capabilities for cloud-native orchestration*. Sophia Antipolis: European Telecommunications Standards Institute; 2025. https://osm.etsi.org/news-events/news/86-etsi-open-source-mano-announces-release-seventeen [Accessed: June, 2026].
12. Amazon Web Services: Mapping ETSI MANO to Kubernetes. In: *AWS Whitepaper*; https://docs.aws.amazon.com/whitepapers/latest/ETSI-NFVO-compliant-orchestration-in-kubernetes/mapping-etsi-mano-to-kubernetes.html [Accessed: June, 2026].
13. H. Moens and F. D. Turck, VNF-P: A model for efficient placement of virtualized network functions, *10th International Conference on Network and Service Management (CNSM) and Workshop*, Rio de Janeiro, Brazil, 2014, pp. 418-423, doi: 10.1109/CNSM.2014.7014205.
14. L. Qu, C. Assi and K. Shaban, Delay-Aware Scheduling and Resource Optimization With Network Function Virtualization, in *IEEE Transactions on Communications*, vol. 64, no. 9, pp. 3746-3758, Sept. 2016, doi: 10.1109/TCOMM.2016.2580150.
15. Deval Bhamare, Mohammed Samaka, Aiman Erbad, Raj Jain, Lav Gupta, H. Anthony Chan. Optimal Virtual Network Function Placement and Resource Allocation in Multi-Cloud Service Function Chaining Architecture. *arXiv:1903.11550 [cs.NI]*. 2017; https://doi.org/10.48550/arXiv.1903.11550.
16. Marco Savi, Massimo Tornatore, Giacomo Verticale. Impact of Processing Costs on Service Chain Placement in Network Functions Virtualization. *IEEE NFV-SDN 2015.* 2015; https://boa.unimib.it/bitstream/10281/273124/1/nfv_processing.pdf.
17. Balázs Németh, Nuria Molner, Jorge Jorge Martín-Pérez, Carlos J. Bernardos, Antonio de la Oliva, Balázs Sonkoly. Delay and reliability-constrained VNF placement on mobile and volatile 5G infrastructure. *arXiv:2007.11870*. 2020; https://doi.org/10.48550/arXiv.2007.11870.
18. M. Ghaznavi, A. Khan, N. Shahriar, K. Alsubhi, R. Ahmed and R. Boutaba, Elastic virtual network function placement, *2015 IEEE 4th International Conference on Cloud Networking (CloudNet), Niagara Falls, ON, Canada, 2015*, pp. 255-260, doi: 10.1109/CloudNet.2015.7335318.
19. A. Leivadeas and M. Falkner, VNF Placement Problem: A Multi-Tenant Intent-Based Networking Approach, *2021 24th Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)*, Paris, France, 2021, pp. 143-150, doi: 10.1109/ICIN51074.2021.9385553.
20. Zhang, Z., & Wang, C. (2025). Service Function Chain Migration: A Survey. *Computers, 14(6), 203*. https://doi.org/10.3390/computers14060203.

How to cite this article: Tran Ngoc Viet Anh. A lightweight SLA-Aware framework for virtual network function placement in NFV service chains. *International Journal of Research and Review*. 2026; 13(6): 181-189. DOI: *https://doi.org/10.52403/ijrr.20260618*

******