A CPN-based Model for Resource Allocation in Multi-Access Edge Computing Supporting URLLC

Caio Souza, Renata dos Reis, Maria Lima Damasceno, Marcos Falcão, Andson Balieiro
Centro de Informática (CIn), Universidade Federal de Pernambuco (UFPE), Recife, Brazil
{cbbs, rkgr, mgld, mrmf, amb4}@cin.ufpe.br

***Abstract*—Although the integration of Multi-Access Edge Computing (MEC) and Network Function Virtualization (NFV) paves the way for supporting URLCC services in 5G and beyond 5G networks, it raises some practical issues such as limited resources of edge nodes, overhead caused by virtualization, and failures during service processing, which may lead to URLLC requirement violations. By incorporating these features, this work proposes a Coloured Petri Net (CPN)-based model to address the dynamic resource allocation for URLLC services in NFV-MEC-based 5G Networks. Additionally, to mitigate the virtualization overhead, a resource pre-initialization strategy is employed in consonance with service buffering and resource scaling on demand. Results showed that resource pre-initialization has positive impacts on overall system performance without significantly affecting power consumption, achieving similar gains to those achieved by setup rate improvements.**

***Index Terms*—MEC, URLLC, Coloured Petri Nets, NFV, Dynamic Resource Allocation**

I. Introduction

Ultra Reliable and Low Latency Communications (URLLC) is a challenging 5G Network service that requires strict low latency (e.g., a few milliseconds) and ultra reliability. The integration of Multi-Access Edge Computing (MEC) and Network Function Virtualization (NFV) paves the way for supporting URLCC services in 5G and beyond 5G networks [1] as it enables hosting virtualized network functions (VNFs) and applications closer to the end users, reducing the service latency, facilitating dynamic resource allocation by aligning the network capacity with demand fluctuations, and enabling user services to be processed at the network edge [2].

Although this combination favors URLLC support, employing MEC and NFV technologies raises some practical issues that need to be considered in the resource allocation in NFV-MEC systems for supporting URLLC [3]. For instance, the likely resource limitation of edge nodes restricts their service capacity and consequently their availability, which may lead to the forwarding of URLLC service requests to neighboring NFV-MEC nodes or the central cloud [4], introducing uncertainty towards latency. The overhead incurred by virtualization may also affect URLLC services. For example, the deployment of virtualized network functions (VNFs) onto containers or virtual machines (VMs) requires time to set up the VNF, adding an extra delay to URLLC latency. Additionally, failures during service processing may impair URLLC as they lead to VNF resets, increasing the response time.

Multiple works have addressed resource allocation in NFV-MEC nodes. However, the majority overlooks the overhead caused by virtualization, assuming instantaneous provisioning times [5], consider a fault-free environment [6], or disregard the repair delay [5] [7], which may affect URLLC services. Additionally, some works are based on formalisms tied to specific probability distributions (e.g., exponential) to represent the URLLC service behavior [3] [8] [9].

This paper addresses the dynamic resource allocation for URLLC in NFV-MEC-Based 5G Networks considering the overhead caused by virtualization and failure events during service processing. Additionally, to mitigate the virtualization overhead, a resource pre-initialization strategy, which instantiates containerized VNFs in advance, is employed in consonance with service buffering and resource scaling on demand. By incorporating these features, we propose a Coloured Petri Net (CPN)-based model to analyze the dynamic resource allocation for URLLC services in NFV-MEC systems, regarding availability, response time, and power consumption. Results show that resource pre-initialization has positive impacts on overall system performance without significantly affecting power consumption, achieving similar gains to those achieved by setup rate improvements. The latter not only improves response time but also reduces power consumption. Our model offers flexibility to analyze URLLC services and system features with different behaviors and may assist the network operator in properly dimensioning and configuring NFV-MEC systems for supporting URLLC. The remainder of this paper is organized as follows. Section II describes the CPN-based model for a single node NFV-MEC, assuming a virtual environment featured with containers (VNFs) that process URLLC requests. Section III presents the result analysis obtained by extensive discrete-event simulations. Finally, Section IV concludes this paper and highlights future directions.

II. System Model

We consider an NFV-MEC node consisting of c containers (VNFs), with N pre-initialized ($n < c$), and a maximum capacity of k simultaneous URLLC services. Since containers are more agile, lightweight, and resilient than VMs, they have been standardized as the replacement of VMs to accommodate network services in NFV [10]. An arriving URLLC service is admitted to the system if the maximum capacity k has not been exceeded. An available and active container is assigned

to process the admitted service and an inactive one is turned on to ensure N is available and active in the system, i.e., pre-initialized. Setting a container (VNF) up incurs a time overhead to make it ready to process services (active). If all containers are busy processing services, the admitted services are placed in a finite buffer that supports up to q services, with $q = k-c$.

During service processing, failures may occur. In this case, the container (VNF) is restarted and the service is assigned to an available container. If no available container exists, the URLLC service is placed back in the buffer, having higher service priority than new URLLC services. In both cases, service processing is restarted. Figure 1 illustrates the NFV-MEC node dynamics, where URLLC service requests originating from UEs are processed by the RAN, passed to the MEC node, and are handled by containerized VNFs, which are scaled accordingly. Overheads due to container (VNF) instantiation and failures during processing are considered, as they may cause violations of the URLLC requirements. Additionally, a pre-initialization strategy is employed to mitigate these effects.

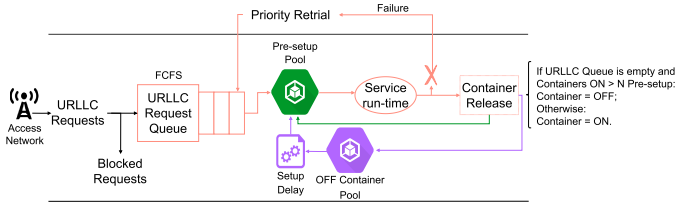


Fig. 1. Resource Allocation in a NFV-MEC node supporting URLLC

To analyze the NFV-MEC system supporting URLLC, we designed a CPN-based model that incorporates features such as admission control, service queue (buffer), failure events during service processing, repair time/setup time, dynamic resource allocation, and container (VNF) pre-initialization. Different from the Continuous Time Markov Chain (CMTC) and Stochastic Petri Nets approach, our model allows the service arrival, service processing, failure, and setup/repair events to follow different probability distributions or behaviors, not limited to exponential or Poisson ones. This flexibility enables better representation of different URLLC applications.

CPN is a formalism capable of modeling and analyzing concurrent systems. It adopts a visual representation in which places are represented by circles, transitions by rectangles, and arcs by directed arrows. An arc always connects a place to a transition or a transition to a place. The places denote the system status and may contain a set of tokens. Colors are associated with the places, denoting their data types. Thus, all tokens of a place must have the same token color, which may be simple (i.e., integer type) or complex as tuples of other colors. The transitions represent events that may cause a system state change by manipulating tokens according to their firing rules. The distribution of tokens in the places determines whether the conditions for transition triggering have been met. The arcs describe the relationship between places and transitions and determine the state change when an event occurs. Arc inscriptions work as functions that define the number of tokens transferred between states [11]. Fig. 2 shows the designed CPN-based model, which comprises 58 arcs, and 12 colors (data types). For better clarity, we divided the model into three parts: Service Arrival (Section II-A), Container Management (Section II-B), and Service Processing and Failure (Section II-C). The following text adopts a bold font for tokens, an italic for functions and variables, and a bold and italic for transitions and places in the text.

A. Service Arrival

Fig. 2 (red part) depicts the model segment that describes the arrival of URLLC services in the NFV-MEC node. The transition ***Customer Arrival*** denotes a URLLC service arrival. This transition is enabled when there is any token in the place ***Customers***. Upon triggering, it consumes a token **U** from ***Customers*** and inserts a token **U** in ***Entry Place*** and places another token back in ***Customers***. A token **U** represents a URLLC service. However, the place ***Customers*** only receives a token after a time interval that describes the time elapses between two successive URLLC arrivals. This time may be defined by a probability distribution or function that characterizes the URLLC application to be analyzed. In our model, it is denoted by the function *InterArrivalTime()*. It is worth observing that the place ***Customers*** is marked with a token **U**. To allow computing the service response time, the token **U** is associated with its arrival time in ***Entry Place***. This time is obtained via function *ModelTime()*.

The service admission control in the system is based on the amount of resources (**R** tokens) in the places ***Unavailable Resources*** and ***Available Resources***, which denote the number of resources already in use and available for assignments, respectively. For instance, *MaxRes* **R** tokens in ***Unavailable Resources*** means that all resources are being used, which implies that new arriving services are not admitted into the system. Thus, a *token* from ***Entry Place*** is moved to ***Blocked Customers*** by firing the transition ***Rejection***, representing that a new service was not accepted into the system. The variable *MaxRes* indicates the system capacity, i.e., the maximum number of simultaneous services in the system, being processed or waiting in the buffer to be processed.

On the other hand, tokens in ***Available Resources*** and ***Entry Place*** fire the transition ***Customer Admission***, representing a service admission into the NFV-MEC system. This transition consumes a token from these places and inserts a token that denotes a service request (**(U,t)**) into ***Admission Queue*** and a resource token **R** into ***Unavailable Resources*** and ***Start***. This last token may lead to a container reset, which is addressed by the ContainerManagement module (see Section II-B).

In our model, the places ***Unavailable Resources*** and ***Available Resources*** are complementary, where the sum of their tokens equals *MaxRes*. Thus, a service admission removes a resource token from ***Available Resources*** and adds one to ***Unavailable Resources***, while the opposite occurs when a service concludes. The places ***Available Resources*** and ***Unavailable Resources*** are initialized with *MaxRes* and zero tokens, respectively.

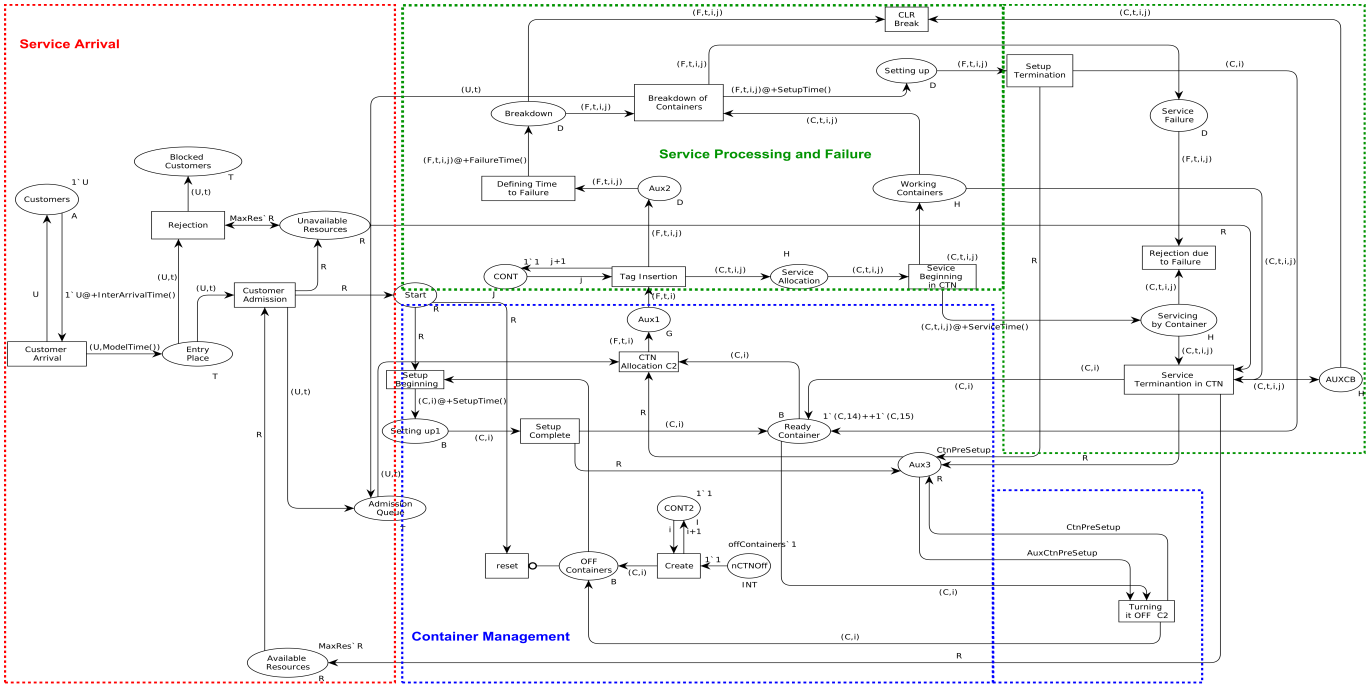


Fig. 2. CPN-Based Model for NFV-MEC Supporting URLLC

B. Container Management

Fig. 2 highlights in green color the segment model for Container Management. The admission of a service (user), represented by the triggering of the transition ***Customer Admission***, places a resource token in ***Start***, which initializes a new container to maintain the number of pre-initialized containers according to the system configuration since one container will be assigned to process the new service. Thus, since there are turned-off containers, i.e, tokens in ***OFF Containers***, a container initialization is modeled by firing the transition ***Setup Beginning***, which inserts a token **(C, i)** in ***Setting up1*** after the container setup time. This time represents the overhead of all operations required to make the container ready to process the service and it may be modeled by using probability distributions or functions, for example. In our model, this time is defined by the function *SetupTime()*. To avoid model inconsistencies when there are no OFF containers, i.e, tokens in ***OFF Containers***, the transition ***Reset*** is fired and consumes tokens from ***Start***.

The off-container generation is conducted by the transition ***Create***, with tokens in ***CONT2*** and ***nCTNOff***. These places are initialized with one and *offContainers* tokens, respectively, all assuming the value **1**, where *offContainers* is the total number of containers in the system regardless of those that must be kept turned on in advance. The firing of ***Create*** consumes a token from ***nCTNOff*** and ***CONT2***, associates an identifier **i** to the *token* **(C,i)**, and inserts it in the place ***OFFContainers***. This last place models the pool of containers available for initialization. The transition ***Create*** also puts a new token with value **i+1** in ***CONT2***. Thus, ***CONT2*** and ***Create*** implement a counter.

A token (container) in ***Setting up1*** enables the transition ***Setup Complete***. Upon triggering, it places a container token **(C,i)** in ***Ready Container***. This place represents the set of containers ready and available for service processing. It is initialized with the number of pre-initialized containers configured for the system. As ***Ready Container*** is a pool of ready containers, containers that end their service processing may be positioned in this place. Thus, ***Ready Container*** also receives tokens generated by the transitions ***Service Termination in CTN*** and ***Setup Termination***. These transitions release a container after a service conclusion and reset it when a failure occurs, respectively.

Containers are turned off only when the number of ready and available containers for service processing exceeds the number of containers configured in the system to stay on in advance (pre-initialized), denoted as *n*. This policy is incorporated into our model by using the place ***Aux3***. It receives tokens consumed by the same input transitions as the place ***Ready Container***, except for the transition ***Turning it OFF***. Thus, the places ***Aux3*** e ***Ready Container*** present the same amount of tokens. The transition ***Turning it OFF*** consumes a container token **(C,i)** from ***Ready Container*** and *n+1* tokens from ***Aux3***, denoted by variable *AuxCtnPreSetup*. This transition inserts a container token **(C, i)** in ***offContainers*** and places *n* tokens back to the place ***Aux3***, turning the exceeding containers off.

The service requests that are waiting in the buffer, represented by tokens in ***Admission Queue*** are assigned to ready containers for service processing, which are modeled by tokens in ***Ready Container***, where each container is identified by its

identifier (ID). This allocation is represented by the transition ***CTN Allocation***, which consumes a token from these two places and also from ***Aux3*** and inserts an allocation token **(F,t,i)** in the place ***Aux1***.

C. Service Processing and Failure

Fig. 2 (green part) highlights the segment that models the service processing and eventual failure occurrences during the processing. When an allocation token **(F,t,i)** is placed in ***Aux1***, the transition ***Tag Insertion*** is enabled. It adds a failure marking **j** to the allocation token, which assumes the value of the token consumed from ***CONT***, used as a counter. Since a container may process different services throughout the NFV-MEC operation, this failure marking allows identifying during which service attendance the failure happened. The token with the failure marking ***(F,t,i,j)*** is placed in ***Aux2*** and an allocation token with the same failure identifier **(C,t,i,j)** is inserted into ***Service Allocation***. The token in ***Aux2*** fires the transition ***Defining Time to Failure***, consuming a token from ***Aux2*** and adding a temporized token in ***Breakdown***. The time associated to the last token denotes the time between two successive failures and it may be modeled via probability distributions or functions that characterize the possibility of failure occurrences during the service processing. In our model, we express it via function *FailureTime()*.

Simultaneously with the token placed in **Aux2**, a token **(C,t,i,j)** is inserted into ***Service Allocation***, which enables the transition ***Service Beginning***. Upon triggering, this transition denotes that the service attendance has started, positioning a token **(C,t,i,j)** in ***Working Containers*** and a temporized token in ***Servicing by Container***, which models the service processing time and may be described by distribution probabilities or functions. In our model, the function *ServiceTime()* defines this time. The place ***Working Containers*** models the pool of containers that are processing services and, consequently, susceptible to failures. Thus, tokens in ***Working Containers*** enable the transition that represents the failure occurrence (***Breakdown of Containers***) as well as the transition that indicates the successful service conclusion (***Service Termination in CTN***), but only one is fired and it depends on when the tokens will be available in ***Breakdown*** and ***Servicing by Container***.

If the token is available in ***Servicing by Container*** first, the transition ***Service Termination in CTN*** is fired, denoting a service conclusion. It consumes tokens **(C,t,i,j)** from ***Servicing by Container*** and ***Working Containers*** and places resource tokens **R** in ***Aux3***, ***Available Resources***, and ***Ready Containers***. Otherwise, the transition ***Breakdown of Containers*** is triggered, consuming allocation **(C,t,i,j)** and failure tokens **(F,t,i,j)** from ***Working Containers*** and ***Breakdown***, respectively, and placing the service request back to the service queue, by inserting a token into ***Admission Queue***. In addition, ***Breakdown of Containers*** generates an immediate failure token **(F,t,i,j)** in ***Service Failure*** and a temporized one in ***Setting up*** to denote the container reset (recovery from failure). The reset time may be similar to the container setup time. So, in Fig. 2, this time is also denoted by *SetupTime()*. In this respect, the place ***Setting up*** models the pool of containers that face failures during service processing and are being restarted. Moreover, a token **(F,t,i,j)** available in ***Setting up*** fires the transition ***Setup Termination***, representing that the container reset process concluded and it is ready for service processing. This transition places a token **(C,i)** in ***Ready Container***.

D. Performance Metrics

Placing services closer to the users can reduce latency and enable URLLC. However, the likely resource limitation of edge nodes restricts their service capacity and consequently its availability.Thus, analyzing the MEC-NFV node availability is imperative. In our model, availability (A) is the system's ability to accept a new URLLC service request, which depends on the amount of available resources. It is the ratio between the number of admitted services (triggering of ***Customer Admission***) and the number of arrived service requests (triggering of ***CustomerArrival***).

Response time assumes a crucial role in URLLC applications as these services have strict latency requirements. It is defined as the interval between the service admission at the MEC-NFV node and its conclusion, including any configuration/restarting overhead. The response time of the service i is computed by subtracting the timestamps of the occurrences of ***Customer Admission*** and ***Service Termination***. The average response time (T) is given considering the number of admitted services ($\#CustomerAdmission$).

The power consumption (P) is an important component of the operational costs. It is given by summing the power consumption of the containers in idle, setup, and busy states, denoted as P_{idle},P_{setup}, P_{setup}, and P_{busy}, respectively. These components are computed using Eqs. 1, 2, 3 from [10], where x represents the number of containers in the respective state. The numbers of containers in idle and busy states are denoted by the number of tokens in ***Ready Container*** and ***working Containers***, respectively. The sum of tokens present in ***Setting Up 1*** and ***Setting Up*** expresses the number of containers in the setup state.

$$P_{idle} = 110 + 0.0501x \tag{1}$$

$$P_{setup} = 112 + 0.455x - 0.00224x^2 \tag{2}$$

$$P_{busy} = 110 + 7.23x \tag{3}$$

III. Result Analysis

Results from our CPN-based model (markers) were validated against our previous CTMC-based model (lines) [9]. Thus, the functions *InterArrivalTime(), SetupTime(), FailureTime, and ServiceTime()* were set as exponential distributions. Additionally, results considering Gaussian distributions are also presented to show the model's flexibility. We consider two scenarios, each evaluating the influence of a pair of parameters: container setup rate (α) and resource pre-initialization (N) in Section III-A. These parameters may denote hardware and software improvements for reducing the time to make network functions ready for processing services, and the number of containers (VNFs) maintained in an idle state; service (μ)

and failure rates (γ) in Section III-B, which may represent a system with different reliability levels and enhancements in service request process speed, respectively.

For both scenarios, we considered different URLLC loads, varying the arrival rate (λ) from 5 to 30 requests/ms. The main parameters were set following a subset of the 3GPP Release 16 [12] and unless stated otherwise, the baseline values for γ and α were set to 0.001 and 1 unit/ms, respectively, in accordance with [13]. Thus, the rates for exponential distributions or mean (inverse of the rate) for Gaussian ones were set according to Table I, with the Gaussian ones adopting variance equals 1. The following results represent the average of each metric, considering 10 simulation instances, with 2700000 steps and a confidence interval of 95%.

TABLE I
SCENARIO SETS

Sec.	λ	α	γ	μ	N	C	k
III-A	5-30	0.5, 1	10^{-3}	1	2,4,6	15	25
III-B	5-30	1	10^{-2}, 10^{-3}	1,2,4	2	15	25

A. Effects of the resource pre-initialization and setup rate

Figs. 3a and 3b show that availability decreases with the increment of the URLCC load. However, it is only slightly affected by variations in α (setup rate) and N (number of pre-initialized resources), with all configurations exhibiting similar behaviors and the curves predominantly overlapping. Notably, in Fig. 3a, configurations with higher N and α values show a small superiority when λ equals 15, the total number of containers, when exponential distributions are considered. These results are somewhat expected since the amount of resources in the NFV-MEC node is kept the same for all configurations. We can also notice a similar performance between the configurations (α = 0.5, N = 4) and (α = 1, N = 2), as well as (α = 0.5, N = 6) and (α = 1, N = 4), which indicates that a lower setup rate can be mitigated by container pre-initialization. Increasing N by 2 provides availability similar to that obtained by incrementing the setup rate by 0.5 for points where the container states (VNFs) vary more often (α from 10 to 20).

N and α present more impact on the response time, as shown in Figs. 3c and 3d, because their improvements reduce the time that a service waits for a container (VNF) to be ready for processing. As a result, URLLC services may experience shorter response times. This effect is noticeable for λ up to 20, when container initialization occurs more frequently. In Fig.3c, we can observe similar performance between the configurations (α = 0.5, N = 4) and (α = 1, N = 2), as well as (α = 0.5, N = 6) and (α = 1, N = 4). For higher URLLC loads, the response time is not affected by α, because the containers stay active, processing services continuously for longer periods. For Gaussian distributions (Fig. 3d), variations on the response time are more evident when λ exceeds the system's processing capacity, where configurations with higher N values achieve lower response times and the setup rate shows less influence.

For power consumption, Figs. 5a and 5b show that there is no significant difference between the configurations, even those with higher N values. This suggests that maintaining many pre-initialized containers is a good strategy, as it lowers the response time while only slightly increasing the power consumption, as indicated by the low increase rate shown in Eq. 1. However, this approach may cause excessive memory consumption, which could impose restrictions on the number of containers kept in an idle state, in addition to increasing the power spent without processing services.

B. Effects of the URLLC service rate and failure rate

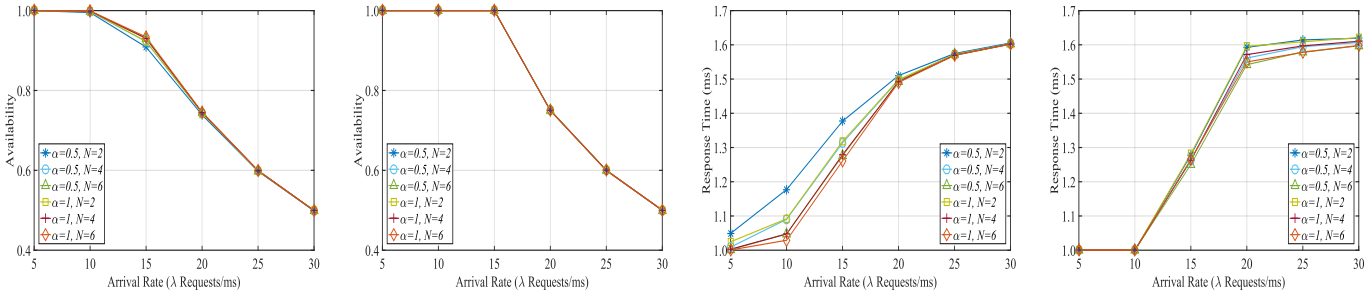
Figs. 4a and 4b illustrate the significant impact of the service rate on availability, whereas the failure rate (γ) has a relatively minor impact. For instance, when λ equals 20, improving μ from 1 to 2 boosts availability by around 61%, from 0.59 to 0.96. Conversely, for the same load and considering μ equals 1, increasing the container (VNF) reliability by a factor of 10 (from 0.01 to 0.001) results in a gain of only 0.86%, with availability increasing from 0.5939 to 0.599. This suggests that investing in service processing improvements (e.g., advanced processing techniques) may be more beneficial than focusing solely on system reliability.

Enhancing processing efficiency also significantly impacts response times, as shown in Figs. 4c and 4d. For instance, doubling the service rate from 1 to 2 enables the NFV-MEC system to support URLLC applications such as robotics and telepresence, which demand a maximum response time of only 1 ms [1], even at higher arrival rates. Naturally, a higher failure rate increases the number of services redirected to the service queue, thus increasing the response time. Despite this, the differences in response times are small because resource pre-initialization can mitigate this effect by ensuring that services redirected to the queue are served more quickly.

Figs. 6a and 6b show that a higher service rate leads to lower power consumption, especially under low URLLC loads. This occurs because, at a higher URLLC service rate, fewer containers remain in the processing state, which is the state with the greatest impact on the system's consumption. However, the curves tend to converge as the URLLC load increases, reaching a point where the containers are continuously active and no longer turned off or idle. Additionally, there is a small variation in power consumption concerning the failure rate, where higher failure rates result in lower P. This decrease is not necessarily beneficial, as it stems from the container setup process (which consumes less energy) and leads to power being consumed without carrying out services.

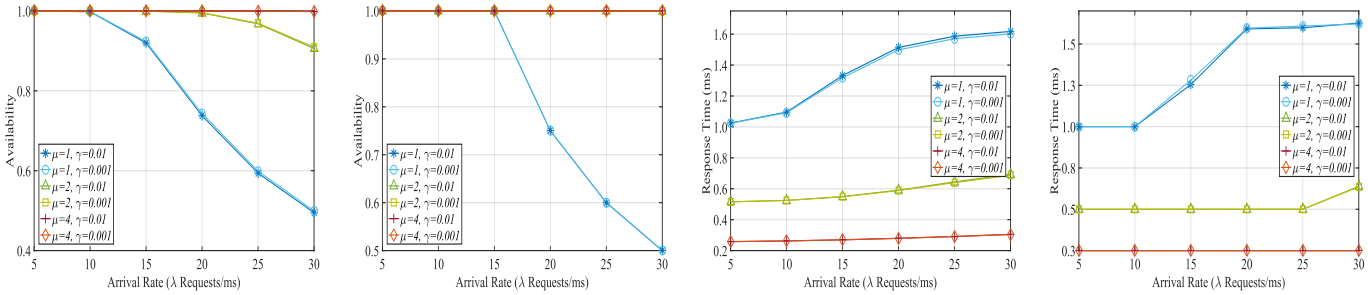
IV. CONCLUSIONS AND FUTURE DIRECTIONS

This work proposed a CPN-based model for virtual resource allocation in MEC-NFV systems. The model incorporated practical factors such as failures during processing and setup (repair) times, since they can incur significant impacts on the URLLC, and employed resource pre-initialization to mitigate their effects. Results showed that resource pre-initialization has positive impacts on overall system performance without significantly affecting power consumption, achieving similar gains to those achieved by setup rate improvements. Results showed that resource pre-initialization has positive impacts



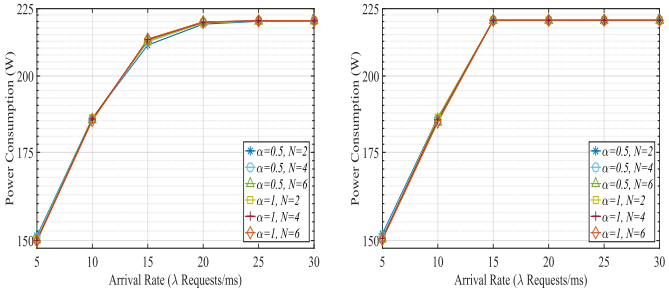
(a) Availability (Exponential) (b) Availability (Gaussian) (c) Response Time (Exponential) (d) Response Time (Gaussian)

Fig. 3. Effects of the container setup rate (α) and resource pre-initialization (N)



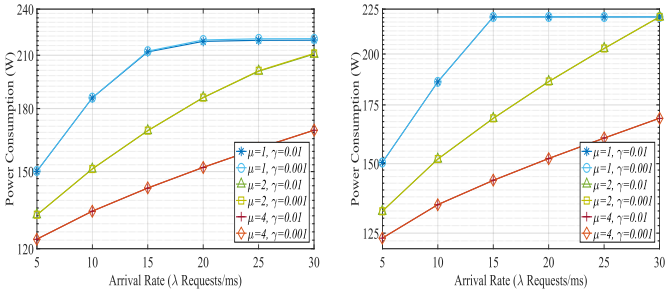
(a) Availability (Exponential) (b) Availability (Gaussian) (c) Response Time (Exponential) (d) Response Time (Gaussian)

Fig. 4. Effects of the service rate (μ) and failure rate (γ)



(a) Exponential (b) Gaussian

Fig. 5. Effects of α and N values on energy consumption



(a) Exponential (b) Gaussian

Fig. 6. Effetcs of μ and γ values on energy consumption

on overall system performance without significantly affecting power consumption, achieving similar gains to those achieved by setup rate improvements. The latter not only improves response time but also reduces power consumption. In brief, the proposed model serves as a valuable tool for comprehending the operational dynamics of MEC-NFV nodes, thereby assisting network operators in properly dimensioning and configuring these systems.

REFERENCES

[1] M. U. A. Siddiqui et.al., "URLLC in Beyond 5G and 6G Networks: An Interference Management Perspective," in IEEE Access, vol. 11, pp. 54639-54663, 2023.

[2] M. Setayesh, S. Bahrami and V. W. S. Wong, "Resource Slicing for eMBB and URLLC Services in Radio Access Network Using Hierarchical Deep Learning," in IEEE Transactions on Wireless Communications, vol. 21, no. 11, pp. 8950-8966, Nov. 2022.

[3] M. Falcao, C. B, Souza, A. Balieiro, A. et al., "An analytical framework for URLLC in hybrid MEC environments", The Journal of Supercomputing, 78, 2245–2264, 2022, doi: 10.1007/s11227-021-03945-8.

[4] I. Sarrigiannis et. al., "Online VNF Lifecycle Management in an MEC-Enabled 5G IoT Architecture," in IEEE Internet of Things Journal, vol. 7, no. 5, pp. 4183-4194, May 2020, doi: 10.1109/JIOT.2019.2944695.

[5] Z. Tong, T. Zhang, Y. Zhu and R. Huang, "Communication and Computation Resource Allocation for End-to-End Slicing in Mobile Networks," 2020 IEEE/CIC International Conference on Communications in China (ICCC), Chongqing, China, 2020, pp. 1286-1291.

[6] W. Li and S. Jin, "Performance Evaluation and Optimization of a Task Offloading Strategy on the Mobile Edge Computing with Edge Heterogeneity," The Journal of Supercomputing, vol. 77, no. 11, pp. 12486-12507, Nov. 2021.

[7] L. Yala, P. A. Frangoudis and A. Ksentini, "Latency and Availability Driven VNF Placement in a MEC-NFV Environment," 2018 IEEE Global Communications Conference, 2018, pp. 1-7.

[8] M. Falcão at. al., "Dynamic Resource Allocation for URLLC in UAV-Enabled Multi-access Edge Computing," 2023 Joint European Conf. on Networks and Comm. & 6G Summit, 2023, pp. 293-298.

[9] C. Souza et.al., "Modelling and Analysis of 5G Networks Based on MEC-NFV for URLLC Services," in IEEE Latin America Transactions, vol. 19, no. 10, pp. 1745-1753, 2021.

[10] K. Nguyen, F. Simonovski, F. Loh, T. Hoßfeld and N. H. Thanh, "Investigation of Container Network Function Deployment Costs in the Edge Cloud," 2024 27th Conference on Innovation in Clouds, Internet and Networks (ICIN), Paris, France, 2024, pp. 9-16.

[11] A. Shahidinejad, M. Ghobaei-Arani, L. Esmaeili, Leila, "An elastic controller using Colored Petri Nets in cloud computing environment", Cluster Computing, pp. 1-27, 2019, doi: 10.1007/s10586-019-02972-8.

[12] 3GPP. System architecture for the 5g system (5gs). White Paper, 2020.

[13] K. Kaur et.al., "Container-as-a-Service at the Edge: Trade-off between Energy Efficiency and Service Availability at Fog Nano Data Centers," in IEEE Wireless Comm., vol. 24, no. 3, pp. 48-56, June 2017.