A Petri Net-based model to study the impact of traffic changes on 5G network resilience

Rui Li, Bertrand Decocq, Yiping Fang, Zhiguo Zeng, Anne Barros

A Petri Net-based model to study the impact of traffic changes on 5G network resilience

Rui Li, Bertrand Decocq

Orange Innovation, Orange Labs, France. E-mail: {rui.li;decocq.bertrand}@orange.com

Yiping Fang, Zhiguo Zeng, Anne Barros

Chair on Risk and Resilience of Complex Systems, Laboratoire Génie Industriel, Centralesupélec, Université Paris-Saclay, France. E-mail: {yiping.fang;zhiguo.zeng;anne.barros}@centralesupelec.fr

The advent of 5G has enabled a wide variety of devices to access the network. With the digitization of industry, more and more vertical services, such as smart cities, remote health, and autonomous driving, rely on 5G networks for communication. These verticals bring new challenges to the telecommunication network resilience. Among them, sudden traffic change seems to be a critical challenge that impacts the resilience performance of 5G networks. This paper presents a network model for future 5G infrastructures based on Petri nets by taking into consideration the particularities of network virtualization and softwarization. This work also seeks to analyze the effectiveness of microservice-level autoscaling and network isolation by using discrete event simulation. The results suggest that both autoscaling and network isolation could increase network resilience when network traffic changes abruptly.

Keywords: 5G, Resilience, Quality of Service, telecommunication network, Kubernetes, Petri Net, discrete event simulation, complex system.

Acronyms

CN	Core Network
CU	Centralized Unit
DU	Distributed Unit
MS	Micro Service
NFV	Network Function Virtualization
RAN	Radio Access Network
SDN	Software-Defined Networking
SFC	Service Function Chaining
UPF	User Plane Function
VNF	Virtual Network Function

1. Introduction

Telecommunication systems and infrastructures keep continuously evolving in order to meet the growing needs of private and business users. By adopting technologies such as NFV (Network Function Virtualization), and SDN (Software-Defined Networking), 5G can meet the various requirements from end-users. However, 5G networks are under numerous challenges as well. As defined by Sterbenz et al. (2010), network resilience is the ability of the network to provide and maintain an acceptable level of service in the face of various faults and challenges to normal operation. It has drawn a lot of attention in the field of 5G networks. One of the most common challenges that a 5G network may encounter would be traffic variation. Traffic variation may be due to user equipment's behaviors, malicious attacks, or other issues. In the event of sudden traffic increases, large amounts of packets sent to the network will congest the network functions and saturate the telecommunication system.

With more and more objects connecting to the internet, dealing with the variant traffic to better adjust the network to the load becomes a critical issue for 5G. The good news is that by adopting NFV and SDN, the 5G system can change its scalability. Scaling can help 5G networks tackle this traffic variation issue. When traffic increases, some overloaded parts will be scaled out by creating more instances to share the load to avoid congestion. When the traffic decreases, the unnecessary instances are scaled in to free up unnecessary resources for other usages.

Some research has started the study of the scaling impact on 5G performance. Alawe et al. (2018), Rahman et al. (2018) and Subramanya and Riggio (2021) modeled the scaling problem

as a time series forecasting problem that predicts the future number of VNF instances in response to dynamic traffic changes. Rotter and Van Do (2021) proposed a queuing model for a scenario where a threshold-based algorithm controls the number of UPF (User Plane Function) instances depending on users' traffic. However, none of these works analyzes the impacts on resilience. The second drawback of these works is that the impact of the traffic variation over a short time interval is neglected. 5G is believed to deliver services to almost every vertical industry, including the services sensitive to latency or services that require very high reliability (3GPP (2020)). Even a second-level performance loss will significantly violate the service level agreement. Another limitation is the lack of consideration of risk propagation. The congestion brought by traffic change can easily propagate from one part of the network to another if not well isolated, degrading 5G network resilience.

The objectives of this work are to study the traffic variation impact on 5G network resilience performance in a short time interval and analyze the effectiveness of network isolation on congestion propagation. However, 5G is a complex system, especially in terms of management and orchestration (Nencioni et al. (2018)). To estimate the resilience of 5G networks, we need to build a comprehensive model that comprises the processing of a network service packet, the life cycle of network elements, etc.

In this work, we developed a Petri Net-based model for 5G networks whose network functions are managed by a Kubernetes management and orchestration system. By carrying out discrete event simulation, we show that our model is capable of evaluating the network service performance and resilience under different traffic patterns. The main contributions of this work are the following:

- A Petri Net based-model considering the virtualization of 5G and the transmission of user plane packets from an end-to-end point of view
- The modeling of microservice-level autoscaling mechanism
- The investigation of network congestion and its propagation using the discrete event simulation
- The estimation of network service latency and the acceptance rate under traffic change

The paper has been organized in the following way. We briefly introduce the virtualized telecommunication networks in section 2. In section 3, we present the Petri Net-based model. Two use cases on autoscaling and the simulation results are given in section 4. Finally, section 5 concludes the work with some remarks and outlines the future works.

2. The virtualized telecommunication system

To deliver an End-to-End service, 5G networks need to steer the traffic through a set of VNFs (Virtual Network Functions) distributed in RAN (Radio Access Network) and CN (Core Network), called SFC (Service Function Chaining). In the previous generations, these functions were implemented in the form of physical boxes. With NFV and SDN, these functions are virtualized, and further softwarized. By doing so, 5G networks become more flexible and can choose where and when to implement these functions. Virtual machines and containers are the most classical ways to implement virtualization. The former contains its own OS, while the latter packs only an application and necessary files.

2.1. *NFV architecture*

As shown in Figure 1, to deliver an End-to-End service, SFCs direct the traffic to traversal through a set of network functions. With NFV, these functions become VNFs and are connected by virtual links. Containerization, which is more lightweight and flexible, is selected as the virtualization solution in this work. Then, each of these VNFs is in the form of a set of containers. These containers are thus the components of a VNF and are equivalent to microservices. We assume that these containers are instantiated on physical servers. Therefore, to deploy a container, we need to select a physical machine and allocate a certain amount of resources, such as CPUs and memories.

The main benefit of adopting NFV is that it improves the scalability of 5G networks and facilitates the management of SFCs and network

functions by changing the quantities of containers at any time and place according to the service requirements, traffic conditions, or the decision of operators. In this paper, Kubernetes is selected to manage and orchestrate the containerized VNFs.

2.2. *Kubernetes: container deployment and management platform*

Kubernetes is in charge of deploying containers and managing the life cycle of containers, such as load balancing, self-healing, etc. Inside a Kubernetes cluster, there are a set of nodes that correspond to a set of worker machines. Kubernetes will deploy pods (groups of one or more containers) on nodes. We assume in our paper that a pod is equivalent to one container and one microservice application, which is also a component of a VNF. A node is equivalent to a physical machine or a server.

Since we focus on the resilience performance of network service under traffic variation, our main interest in Kubernetes is the autoscaling mechanism. There are many ways for Kubernetes to apply autoscaling. The most commonly used method is the horizontal pod autoscaling, which automatically updates the number of pods to match the traffic demand. To be more concrete, Kubernetes intermittently observes the metrics of a microservice such as CPU utilization and memory utilization and judges if scaling should be applied or not. When the traffic load increases, Kubernetes will try to scale out by deploying more Pods. If the load decreases, Kubernetes will scale in some pods to make sure the resource utilization is at the expected level.

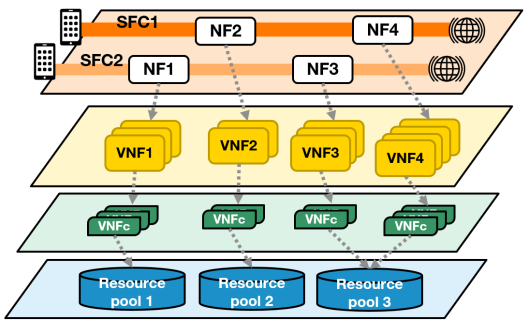

Fig. 1. 5G End-to-End service delivery model with SFC, VNF, VNF component and Resource layers.

We assume that in this paper, Kubernetes only observes the underlying resource, the CPU utilization. For each pod, it allocates several units of CPUs from the node. To process a packet, the pod will use one unit of CPU. We also assume that the autoscaling is applied at the microservice level, i.e., changing the number of pods (replicas). The metric we collect will be the average utilization rate of the pod of the same microservice. A scaling-out action will be carried out only when there are enough physical resources on a physical machine to create a new pod.

2.3. *Performance indicators of End-to-End network services*

This paper mainly focuses on two performance indicators, latency and acceptance rate. These two indicators can be directly applied to estimate the network performance under an undesirable event. We extend the definition of resilience by introducing the "resilience triangle" (Tierney and Bruneau (2007)) and the use service acceptance rate as a system performance indicator to measure network resilience.

Latency is one of the critical indicators for network service. Latency describes the time that takes to transfer a given piece of information from a source to a destination, from the moment it is transmitted by the source to the moment it is successfully received at the destination (3GPP (2021)). In our 5G system model, network service latency is composed of transmission time in RAN, processing time at each VNF(a set of microservices), and the waiting time in the queue of each microservice. Other types of latency, such as time spent on a switch, are not considered. We calculate the average latency of the packets in a time interval of 0.1 seconds. Service latency is very important for vertical industries such as remote health and autonomous driving. The 5G network resilience requires the system to meet low latency requirements despite the presence of risks.

We propose an acceptance rate indicator to estimate the 5G networks' resilience. Based on our assumptions, when a microservice queue is full,

the arriving packets may not join the queue and then be rejected. The packet losses in Transport Network and the radio transmission are not taken into consideration. In some cases, a network service can be very sensitive to packet loss since it impacts the quality of receiving data. The acceptance rate is the number of packets arrived at its SFC destination divided by the total sent packets over a time interval of 0.1 seconds. In a normal operation mode, the packet acceptance rate should be 100%. However, these indicators will not stay at a stable interval under some incidents. For example, in the case of traffic variation, congestion may occur at some microservices. As a result, packets may need to queue up for an available microservice pod and even be rejected if the queue is full. Then the latency will increase, and the acceptance rate may decrease.

In this work, both latency and acceptance rate are used to estimate the network service performance. By further presenting acceptance rate performance as a "resilience triangle", the resilience of the network can be described as the ability to adapt, maintain, and recover.

3. Petri Net-based model for 5G system description

Quite a number of different approaches have been applied to model 5G networks. In this paper, we focus on the dynamic behaviors of a 5G system. We intend to track how packets are processed in the system, creating numerous states for the system. Some approaches, such as Markov Chain and fault tree, are not practical to describe the dynamics or capture the dependencies of such a complex system. Petri Net is a widely used technique for tracking systems' states, dynamics, and constraints. We use Petri Net to model the 5G system. Readers are invited to refer to Li et al. (2022), our previous work, for more details on the Petri Net model.

3.1. *Petri Net for a packet processing*

A request packet is processed in 5G networks by a series of VNF, which can be further extended into a series of microservices. Figure 2 shows a Petri Net of one of these microservices. As explained in Table 1, a packet first arrives at the microservice at place p_1. Then the packet is inserted by t_1 to a waiting list if there is enough capacity in the queue p_2. Otherwise, this packet is rejected to the place p_3. According to the load on the microservices replicas (in the form of pods), the packet will be sent by t_2 to the less used pod p_4. The queue follows the rule of first-come, first-served. Then the packet passes a timed transition t_3 and finally successfully finishes the task in microservice p_5.

Table 1. Place and transition explanations of a microservice process Petri Net.

Element	Explanation
p_1	Packet(s) arriving at microservice
p_2	Packet waiting list for microservice
p_3	Packet rejected due to queue capacity
p_4	Pod replicas of microservice
p_5	Treated Packet(s)
t_1	Packet(s) inserting to waiting list
t_2	Pod selection based on workload
t_3	Packet processing

3.2. *Petri Net for microservice-level autoscaling*

Scalability is one of the most important features of a 5G system. Dynamically changing the system scale according to the load will vastly improve the performance, particularly the resilience performance. In this paper, we focus on microservice-level autoscaling. Kubernetes, as an automatic deployment and management platform for microser-

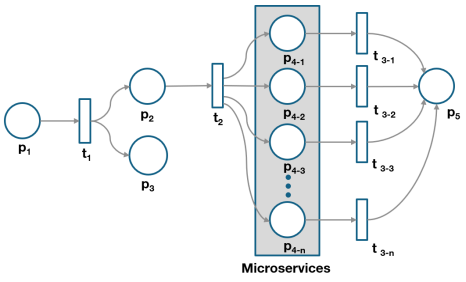

Fig. 2. Petri Net of a microservice process.

vice pods, takes charge of changing the number of pod replicas according to a certain algorithm as presented in Algorithm 1. In this paper, the autoscaling mechanism is to observe the average pod CPU usage metrics intermittently. This time interval is called the sync period. The CPU usage depends on the number of packets that a pod processes. When the resource utilization rate is above the upper threshold, Kubernetes will send a scaling-out decision to increase the number of microservice replicas (i.e., pods) and vice versa.

Algorithm 1 Autoscaling algorithm

Input:
CPU metric values: $I = [I_1, I_2, ..., I_n]$,
desired CPU metric value V,
upper bound:B_U, lower bound:B_L
Output: new replica number: N

desired number of pod replicas: $N \leftarrow n$
sum of indicator values: $s \leftarrow 0$
for $i = 1$ to n **do**
    $s \leftarrow s + I_i$
end for
average of indicator values: $a \leftarrow \frac{s}{n}$
desired replica number: $d \leftarrow$ ceil $(\frac{a}{V})$
if $a > B_U$ or $a < B_L$ **then**
    $N \leftarrow d$ ▷ new replica number
end if
return N

The Petri Net representation is depicted in Figure 3 and explained in Table 2. Kubernetes at place p_1 collects the metrics, which runs intermittently (15 seconds by default). The algorithm will tell Kubernetes to take different decisions depending on the metric value. If the value is higher than the threshold, the transition t_1 will activate and send an increasing pod number order at place p_2. Free resources at place p_4 and the order from p_2 work together to activate t_4 to create new replicas of the microservice at p_5. If the value is lower than the threshold, transition t_2 will activate and send a decreasing replica number order at place p_3. Then the running replicas at p_5 and p_3 activate the transition t_5 to terminate replicas and recycle the resources allocated by them to p_4. If the metric value is inside the threshold, only transition t_2 will activate, and autoscaling will not be triggered.

Table 2. Petri Net of Kubernetes autoscaling.

Element	Explanation
p_1	Kubernetes autoscaling probe
p_2	Increase pod number
p_3	Decrease pod number
p_4	Free resources
p_5	Running Pod replicas
t_1	Scaling-out decision
t_2	Scaling-in decision
t_3	No scaling decision
t_4	Create new replica(s)
t_5	Terminate replica(s)

4. Simulation and results

In order to test the performance and resilience of 5G networks, we modeled the system in a Python program based on the Petri Net representation. Then we run discrete event simulation using the SimPy framework based on Python. The microservice processes in the Petri Net model are coded as resource allocation events in the program. We apply the program to two use cases to evaluate network resilience under different situations. In the first use case, we consider a 5G system with one type of user equipment. We inject a traffic variation by increasing the number of packets sent by the user equipment to test the network resilience. In the second use case, the 5G system consists of four local RAN and one centralized CN. We inject the same traffic variation only to the

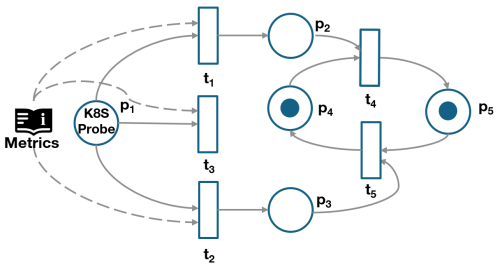

Fig. 3. Petri Net of autoscaling process.

user equipment in one zone and test the network resilience performance in the whole system.

For the first use case, the End-to-End service follows an SFC composed of 3 VNF, as depicted in Figure 4 (in the user plane, they could be DU(Distributed Unit), CU(Centralized Unit)) in RAN, and UPF in CN). The transmission time and packet processing time follow an exponential distribution. However, the packet arrival rate follows a Poisson distribution with a variant parameter. Other parameters are given in Table 3.

Table 3. 5G network parameters.

Number of VNF MSs (microservices)	DU:1 MS, CU:2 MS, UPF:1 MS
MS processing time	8 ms for DU and CU MSs, 10 ms for UPF MS
MS resource (in CPU units) allocation	6 for DU and CU MSs, 12 for UPF MS
Packet processing resource	1 CPU unit for each MS
Initial container/pod replicas	3 pods for each MS
Node capacity (in CPU units)	18 in RAN and 36 in CN
Number of nodes	4 in RAN and 8 in CN
Desired CPU utilization rate	50%
Autoscaling threshold	$\pm$ 30%
Simulation run	1000 iterations

The network has been initially well scaled to meet the traffic of 1000 requests per second. We inject a traffic variation into the system. The request arrival rate linearly increases which lasts 25 seconds from 1200 to 4200 requests per second, beginning at 10s and ending at 35 seconds. Then the traffic goes back to its normal state.

In the normal state, the packet average delay is around 0.035 s, including 34 ms processing delay, 1 ms transmission delay, and negligible waiting delay. However, the load on pods grows with the traffic, and they are soon congested. When a packet demands a microservice, there are no more available pods to serve it. The waiting delay increases, and when the waiting list is complete, the coming packets will be rejected. We compare the packet latency and acceptance rate results under different autoscaling strategies.

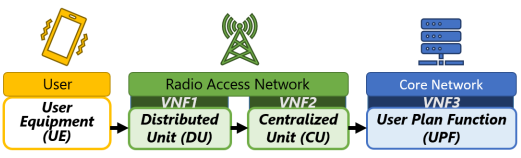

Fig. 4. Service function chain of use case 1.

The service delay result is given Figure 5. If there is no autoscaling, the waiting delay increases up to 80 ms, and the overall delay will not decrease unless the traffic comes back to normal. When we adopt a 15 seconds autoscaling sync period, we find that few pods are scaled out at time 15 s, and more pods are scaled out at 30 s. These two scaling operations are not enough to immediately handle the congestion. In the 10 seconds sync period situation, the scaling-out decisions are taken at 20 and 30 s. The network service delay is shorter than the 15 seconds sync period case after 30 s. Finally, in the 5 seconds sync period autoscaling case, scaling decisions are taken more frequently, and the congestion time and service delay are significantly reduced.

The service acceptance rate result is given in Figure 6. Without autoscaling, the acceptance rate may reduce up to almost 50%. With 5-second autoscaling, both duration and packet rejection is largely reduced. The resilience is improved by shortening the time to adapt to the reverse event and better maintaining the performance. While for 10-second or 15-second autoscaling, the disturbance interval is not significantly reduced, the maximum packet acceptance degradation is about 40%. The acceptance rate is improved only after 30 s. However, the system is not fully recovered. It keeps suffering from the disturbance since the autoscaling at 30 s is insufficient to cope with the continuously growing traffic.

By comparing the acceptance rate performance, the 5-second performs the best in terms of service latency, system suffering time, performance degradation, and restoration time. However, frequently adjusting the scale of 5G network may not be a wise choice. When doing scaling-in, it

takes some time to terminate pods gracefully. The pod resources will not be released immediately. During this time, some of the resources become unavailable, and the system may not be able to scale out when the traffic immediately increases due to a lack of resources. Therefore, some more complicated algorithms can be further applied to set up scaling rules to better adjust the system to the traffic load.

In the second example, we move closer to reality. The DU and CU are located in the local RAN, and UPF is located in a centralized CN as depicted in Figure 7. We consider a 5G system composed of 4 local RAN, which treats the local end users' packets, and a centralized UPF, which treat all end users' packets. Unlike the previous example where we only consider one VNF instance, in this use case, different DU and CU instances are assigned to the user equipment in different zone according to geographical locations. Therefore, these user equipment's packets are isolated in DU and CU but not for UPF. A traffic variation in one zone will firstly congest DU and CU and probably UPF, which is initially set up with more redundancy than local VNFs. Then the packets from other zones will also be delayed due to the congestion happening in shared UPF. In this example, the autoscaling sync period is set to 10 seconds. The abnormal traffic in zone 1 is the same pattern as in the first use case. The latency and acceptance rate for packets starting from different zones are presented in Figure 8 and Figure 9. The parameters are similar to use case 1.

The traffic change from zone 1 congests not only the local VNFs DU and CU but also UPF. Since the UPF is initially scaled for four radio network zones, the traffic congestion on UPF is

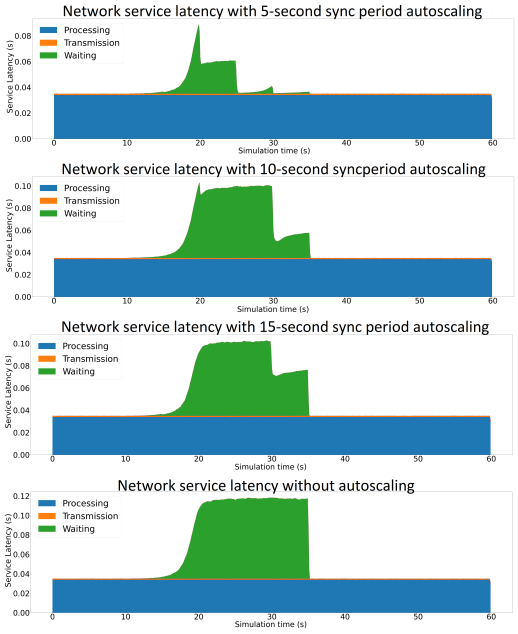

Fig. 5. Network service latency with and without autoscaling. Blue for processing delay, yellow for transmission delay and green for waiting delay.

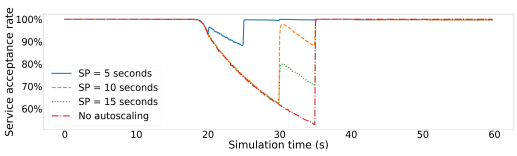

Fig. 6. Network service acceptance rate with 15, 10, 5 seconds sync period autoscaling and no autoscaling.

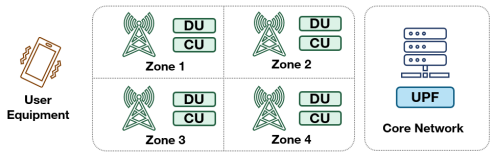

Fig. 7. Network installation of use case 2.

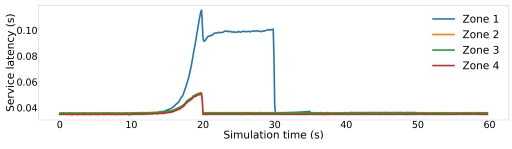

Fig. 8. Network service latency in different zones.

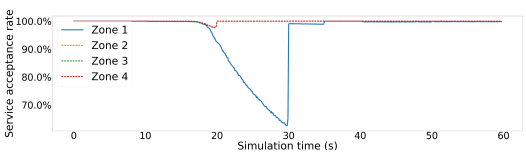

Fig. 9. Network service acceptance rate in different zones.

less severe than in use case 1. The packet waiting delay in zone 1 increases to 70 ms. The packet waiting delay in other zones is about 15 ms, only caused by UPF congestion. At 20 s, a scaling-out decision is taken by Kubernetes. It creates the maximum number of pod replicas that our scaling algorithm allows. For the UPF microservice, scaling out once is enough, the congestion is released, and the packets in zone 2-4 are no longer queuing for UPF. However, in zone 1, local DU and CU microservices are still congested. The result shows that a traffic change can cause congestion on VNF and the congestion can propagate from RAN to CN. By adopting a local RAN isolation, only zone 1 is largely impacted by the traffic change. The network services of other zones are less impacted, with only less than 3% packet loss proving that the effeteness of network isolation on improving network service resilience. Although in this use case, we are limited in physical or geographical network isolation, this result can still be meaningful since it may be extended to a virtual isolation case for 5G QoS or network slicing.

5. Conclusion

In this paper, we proposed a model based on Petri Net for 5G system, considering the dynamics of the virtualized telecommunication network. This model is applied to simulate the performance of 5G network services for two use cases. The results from two use cases show that microservice-level autoscaling can increase the resilience of 5G network in case of traffic variation and how congestion caused by traffic change can propagate from the local RAN to CN. These results are useful for network operators to implement orchestration and management systems and design network isolation according to service requirements.

This work gives an approach to estimating network performance and resilience. Although many parameters, such as processing time and pod capacity, are based on assumptions, the qualitative results on the efficiency of autoscaling and propagation of congestion are still valid. For better simulation results, they can be further refined.

The continuation of this work will focus on a 5G network comprising different network services and analyze if 5G networks are able to satisfy the resilience requirements of different verticals.

References

3GPP (2020, July). TR 38.913 V16.0.0 Study on Scenarios and Requirements for Next Generation Access Technologies.

3GPP (2021, December). TS 22.261 V16.16.0 Service requirements for the 5G system.

Alawe, I., A. Ksentini, Y. Hadjadj-Aoul, and P. Bertin (2018). Improving traffic forecasting for 5g core network scalability:a machine learning approach. *IEEE Network 32*(6), 42–49.

Li, R., B. Decocq, A. Barros, Y. Fang, and Z. Zeng (2022). Petri net-based model for 5g and beyond networks resilience evaluation. In *2022 25th Conference on Innovation in Clouds, Internet and Networks (ICIN)*, pp. 131–135.

Nencioni, G., R. G. Garroppo, A. J. Gonzalez, B. E. Helvik, and G. Procissi (2018, August). Orchestration and control in software-defined 5g networks: Research challenges. *Wireless communications and mobile computing 2018*.

Rahman, S., T. Ahmed, M. Huynh, M. Tornatore, and B. Mukherjee (2018). Auto-scaling vnfs using machine learning to improve qos and reduce cost. In *2018 IEEE International Conference on Communications (ICC)*, pp. 1–6.

Rotter, C. and T. Van Do (2021). A queueing model for threshold-based scaling of upf instances in 5g core. *IEEE Access 9*, 81443–81453.

Sterbenz, J. P., D. Hutchison, E. K. Çetinkaya, A. Jabbar, J. P. Rohrer, M. Schöller, and P. Smith (2010). Resilience and survivability in communication networks: Strategies, principles, and survey of disciplines. *Computer Networks 54*(8), 1245–1265. Resilient and Survivable networks.

Subramanya, T. and R. Riggio (2021). Centralized and federated learning for predictive vnf autoscaling in multi-domain 5g networks and beyond. *IEEE Transactions on Network and Service Management 18*(1), 63–78.

Tierney, K. and M. Bruneau (2007). Conceptualizing and measuring resilience: A key to disaster loss reduction. *TR news* (250), 14–17.