Microservice-based Network Digital Twins: A Slicing Approach

Lal Verda Cakir*†, Khayal Huseynov*, Kubra Duran*, Trung Q. Duong‡, Berk Canberk*
*School of Computing, Engineering and The Built Environment, Edinburgh Napier University, UK
†BTS Group, Turkey
‡Faculty of Engineering and Applied Science, Memorial University of Newfoundland, Canada
Emails: {lal.cakir, khayal.huseynov, kubra.duran, b.canberk}@napier.ac.uk,
verda.cakir@btsgrp.com, tduong@mun.ca

***Abstract*—The Network Digital Twins (NDTs) have become a frontier thanks to their real-time monitoring, analysis, prediction, and optimisation capabilities. However, their architecture has not been designed for the scale of next-generation networks that will ensure ubiquitous connectivity. At this, different NDT applications may require distinct levels of quality of service requirements to be met. With increasing size and heterogeneity in the networks, the processing load at the NDTs escalates, and these requirements may not be met. Therefore, we propose a microservice-based architecture with application-oriented slicing. Here, we present the scaling methodology, which enables scaling at both microservice and slice levels. Then, we evaluate the throughput, delay, and quality of service requirement violation rate metrics under two scenarios. Thanks to the slicing approach with microservice-based implementation, the end-to-end delay and the QoS requirement violation rate are reduced while having higher throughput performance.**

***Index Terms*—network digital twin, quality of service, slicing, microservice**

I. Introduction

Network Digital Twins (NDTs) have emerged as a key technology for network management in recent years [1]. However, they have not been designed to withstand large-scale deployments. In such a deployment, challenges emerge due to data volume, veracity, variety, and velocity. Large enterprise networks can exceed 1,000 devices [2], and each continuously streams data in different sizes and formats, requiring distinct processing steps. This puts significant data processing demands on NDT, which has to be completed under a time constraint.

With this concern, the literature has started exploring microservice-based architectures, which use a modular approach to enable effective scaling and management. However, they have neglected to consider the required infrastructural design and networking. Because as the number of data sources grows, there is a higher probability of congestion at the network and backlog at the NDT [3]. If the newly arrived data is not immediately processed, it will prevent timely actions from being applied, which may lead to degraded and fluctuating network performance. To prevent this, the following challenges have to be addressed.

A. Challenges

- **Heterogeneity**: Modern architectures involve environments where different types of topologies (enterprise, backbone, radio access networks) coexist, and various network management applications are needed [4]. These encompass various network devices, communication protocols, and access networks. This heterogeneity brings changing telemetry data with different formats, volumes and arrivals. This, coupled with the large-scale deployment, makes a monolithic NDT architecture unsuitable due to its scalability limitations. Because they require scaling the entire NDT to scale specific services, which may lead to inefficient use of resources. Additionally, in such architecture, failure in a single component can disrupt the entire NDT, preventing the ensuring resilience and reliability in large-scale environments.
- **Quality of Service Requirements**: As these heterogeneous networks rapidly change and grow in size and complexity, the NDT must be able to meet different QoS requirements. For instance, in a Distributed Denial of Service (DDoS) detection NDT, the telemetry may contain packet captures, necessitating a high bandwidth. On the other hand, a real-time health monitoring and alerting application can have low latency requirements. For this, the microservices have to be able to scale horizontally and vertically. Moreover, a focused approach to networking the microservices must be separated based on their QoS requirements.

II. Related Works

NDTs are envisioned to be a core part of real-time monitoring, optimisation, and autonomous control of the network infrastructure [5]. This is enabled by creating a virtual replica of the physical network forming the NDTs. This allows network behaviour simulation, prediction, and analysis under various conditions without disrupting the network [6]. However, the performance of these relies on the performance of the constructed NDTs. In the [7], the authors have revealed that under congestion, the NDTs cannot achieve the desired level of synchronisation, which will degrade the effectiveness of the NDT. To overcome this, different synchronisation control

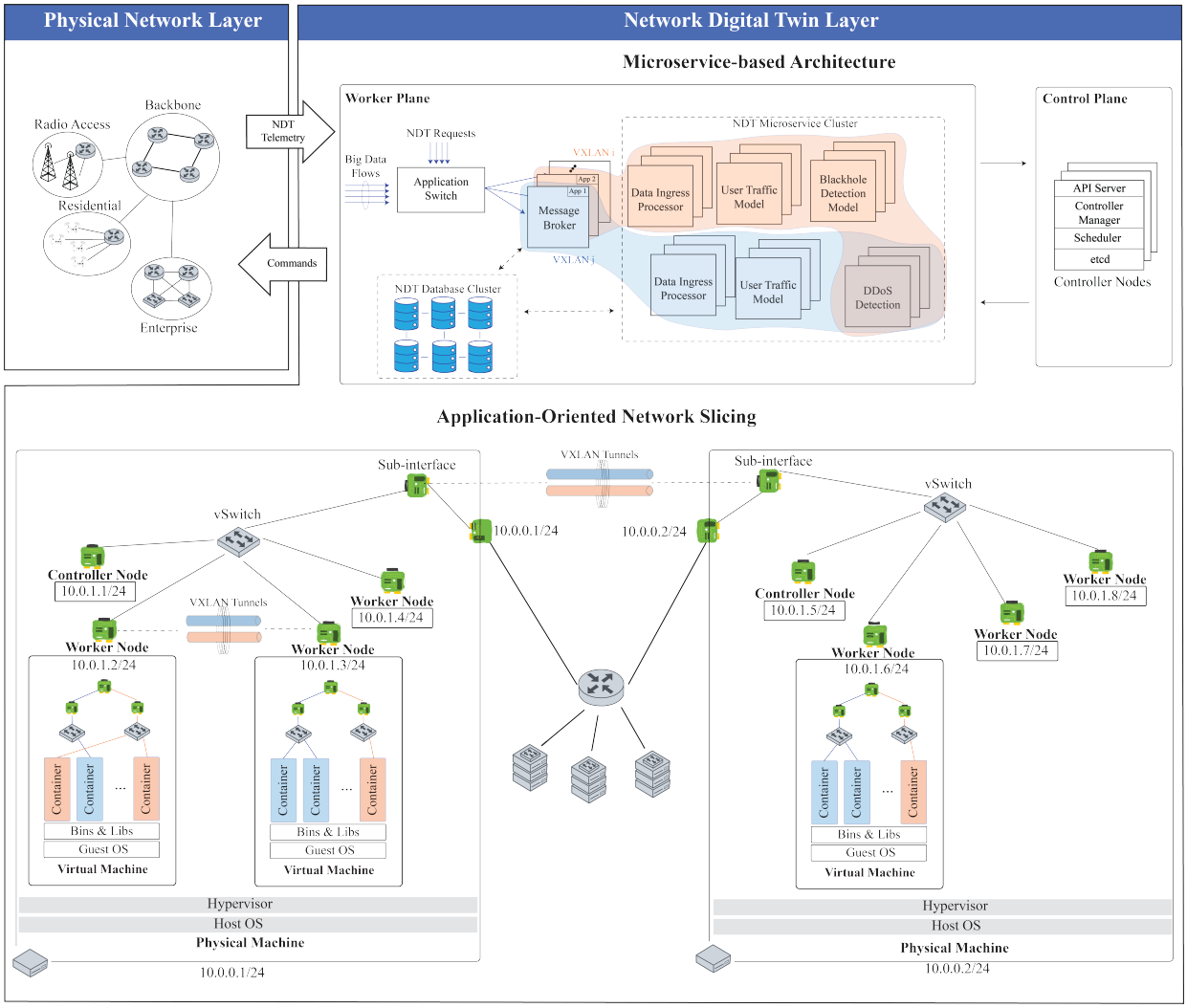

Fig. 1. System Architecture

methodologies have been explored. One example can be the work in [8], which optimises the synchronisation considering the timeliness and the energy consumption jointly. Similarly, in [9], the sampling rate of the collected data is adjusted dynamically. However, for these to yield better performance, the delay introduced by the inference mechanisms at NDT has to be lower than the communication delay. The real-time performance of the NDT does not solely rely on the communication performance. The bottlenecks and performance limitations at the NDT can result in communication delays being negligible compared to the delays caused by backlogs. At this point, designing a scalable architecture at the NDT holds trivial value.

Recent literature has explored implementing DTs using a microservice approach over a monolithic one. In [10], a distributed architecture for disaggregated DTs was proposed to ease the management. Moreover, in [11], microservices and serverless computing paradigms are adopted. Additionally, in [12] and [13], a digital twin as a service platform was designed in a microservice-based manner. However, this work has mainly discussed the services from a software perspective rather than the architecture's infrastructure and networking. Here, the use of microservices brings in the concept of distribution. Therefore, the deployment of these and the scaling has to be considered in detail [14].

Since the NDTs are designed to serve heterogeneous networks, they contain data flows with different sizes, formats, and frequency characteristics. This creates a complicated landscape for ensuring QoS. While this has not yet been discussed under NDT literature, a similar challenge has been present in various networks [15], [16]. To answer this, the network slicing methodologies are used to program the allocation of resources to ensure a certain level of QoS for slices such as enhanced Mobile Broadband, ultra-reliable low-latency and massive machine-type communication.

A. *Contributions*

As summarised above, several efforts have been made to improve the NDTs. However, these cannot handle the chal-

lenges of heterogeneity and quality of service requirements in a unified approach. Therefore, in this study, we address the research question, *"How can we design the microservice networking of Network Digital Twins to meet the quality of service requirements?"*. To answer this, we design a microservice-based architecture that utilises containerization and virtualisation. Then, we implement application-oriented network slicing using VXLANs to segregate NDT resources. Accordingly, the contributions are as follows:

- We propose the microservice-based architecture with application-oriented slicing for NDTs. Here, we form the microservice and database clusters at the worker plane and allow the workload of NDT services to be shared among different instances in parallel and scale independently. Additionally, this modularity makes adding new features or extending the system easier without rebuilding the entire architecture.
- In the proposed architecture, we utilise containerization for microservices and deploy them onto the virtualised infrastructure. This is deployed as microservices that can be deployed in one virtual machine (VM) and other VMs in the same or another physical machine (PM). Then, we enable the communication between virtual machines using a VLAN infrastructure. At this, we use Virtual eXtensible Local-Area Networks (VXLANs) to aggregate and tunnel multiple Layer 3 (sub)networks through Layer 2. Here, we separate different NDT applications by application-oriented network slicing and scale using the hybrid methodology presented.
- We deploy this architecture in a data centre to create the NDT of a large-scale enterprise network. Here, we evaluate the QoS of the microservice cluster in terms of throughput and delay of DDoS NDT and Health Monitoring NDT. Here, we observed that the proposed architecture provides a higher throughput, lower end-to-end delay and lower QoS requirement violation.

III. Proposed Network Digital Twin

We consider the Network Digital Twins under two layers:

- **Physical Network Layer** comprises the network elements and/or network functions such as routers and switches, which can be of different types, such as backbone, enterprise, and radio access networks. The components collect and send data representative of their operational state to the NDT. With this, the devices send configuration data, performance metrics, and real-time operational statistics to the NDT. For example, a router might send periodic updates on its CPU utilisation, the status of its interfaces, or network traffic statistics. On the other hand, the data collected can also be in a larger form, such as packet captures. Here, while collecting packet captures continuously is infeasible, they may be used as part of services run to identify performance bottlenecks or detect intrusion attempts [17]. Therefore, the data flows from each network device can have different characteristics in terms of frequency and data size.
- **Network Digital Twin Layer** forms the virtual replica of the physical network using the telemetry coming from the Physical Network Layer. Here, although the devices may be configured to send their data according to one set twining frequency, the network's performance can introduce variability. The QoS, such as delay, jitter, and packet loss of this communication, may cause the sent data to arrive asynchronously. This asynchronous behaviour can be further exacerbated during high background traffic, leading to data bursts at the NDT side. Thus, while physical network elements are configured to deliver data in a structured manner, real-world conditions will introduce complexity to the data flow. Moreover, at this layer, different service applications are positioned to provide NDT functionalities. These may include data processors, functional models, analysis, and AI/ML-based applications.

In NDTs, big data flows come from many network entities at the physical network layer. These have to be processed, and the service applications do further processing. Therefore, the underlying architecture must be able to answer enormous computational demands [18]. To address this, we introduce the Microservice-based Architecture for NDTs and propose application-oriented network slicing.

A. Microservice-based Architecture

This paper proposes a microservice-based architecture for NDTs as shown in Figure 1. This architecture builds on a virtualised physical infrastructure. On top of these, the VMs are deployed with containers which run the microservices. Here, there are multiple instances (containers) of one microservice, meaning that the processing of different data transfers can occur in parallel. As part of this paradigm, the databases of each microservice are held individually within their containers. Then, a database cluster is formed using these databases to ensure coherence between microservices.

- **Worker Plane** contains the microservice and database clusters. At this plane, the incoming data flows and the outside requests to NDT are relayed to their respective message brokers. This application switch is a proxy-based Layer 7 load balancer, which is configured in a rule-based manner. The message broker queues the incoming data to the respective microservice. A series of examples of the microservices that can be in an NDT is given within Figure 1. Moreover, a performance and flexibility trade-off exists when deciding how many worker nodes will be created within one PM. Having many smaller nodes creates flexibility in management and scheduling resources, but introduces overhead for managing resources. This would also impact efficiency because the nodes will have their own system processes, networking stack, and storage overhead. On the other hand, having a smaller number of nodes will increase bin packing efficiency, reducing the overall resource allocation needed for system overhead.

- **Control Plane** performs the management of the containers and the VMs. For this, controller nodes are deployed into a subset of PMs, which contains an Application Programming Interface (API) server, controller manager, scheduler, and etcd. Here, the API server enables communication between components and also outside of the node. Moreover, the controller manager is a daemon that executes a non-terminating loop to control the state of the VM and its containers. Furthermore, the scheduler assigns a container to a VM based on the given input parameters. Lastly, etcd is a distributed key-value store that holds the state and configuration of the containers.

B. NDT Slicing

The proposed microservice-based NDT is implemented as shown in Figure 1 with these slices for each application realised. This isolation is done through the use of overlay networks. Each slice is formed by the participating microservice instances communicating with each other. Here, microservices are run over containers deployed on multiple virtualised machines created on physical machines within the data centre. Here, an instance of one microservice may be deployed among different VMs and PMs. At this point, different microservices must be able to communicate and exchange data. Here, the communication is established using the gRPC protocol, forming bidirectional flows that carry data serialised using protocol buffers. Here, the services that depend on each other implement synchronous calls, which await the response, while others implement asynchronous calls.

For this, we use Virtual eXtensible Local-Area Networks (VXLAN) [19] within our topology. While one VXLAN can interconnect all, different applications require different QoS levels. Therefore, we employ application-oriented networking and preserve the QoS within NDT applications. Here, the incoming data flows are switched to a message broker in the respective VXLAN. Then, the microservices within each communicate over Layer 3 by tunnelling sub-networks through Layer 2 encapsulation.

C. Hybrid Scaling Methodology

Here, we consider that the processing tasks in the services can be distributed, such as parsing the incoming synchronisation data in XML format. In that case, this microservice can be scaled up to the number of data flows. However, this would cause overprovisioning at the DT. If not all of the network elements are sending at a high frequency, the instances would be idle while occupying the assigned CPU and RAM resources. Therefore, the proposed architecture slicing approach is critical. Here, this scaling can be performed in reactive and proactive manners. Here, the reactive refers to the scaling to be done if the predefined threshold is exceeded, and the proactive scaling refers to the future resource usage to decide on the scaling [20].

In this article, we form the scaling methodology described in the Algorithm 1. Here, the slicing approach monitors the QoS based on the slices, and the scaling methodology is run in a hybrid manner involving both reactive and proactive.

Algorithm 1 Hybrid Scaling Methodology

```
1:  Initialise interval, window, increment, NDTSlices, ResourceManager, LSTM
2:  while true do
3:      for all slice ∈ NDTSlices do
4:          current_load ← slice.get_current_qos()
5:          threshold ← ResourceManager.get_threshold(slice)
6:          if compare(current_load, threshold.upper) then
7:              %Mode Reactive
8:              ResourceManager.allocate(slice, increment)
9:          else %Mode Proactive
10:             history ← slice.get_historical_qos(window)
11:             predicted_load ← LSTM.predict(history)
12:             current_alloc ← ResourceManager.get(slice)
13:             if compare(predicted_load, threshold.upper) then
14:                 scale_up ← calculate_required(.)
15:                 ResourceManager.allocate(slice, scale_up)
16:             else if compare(predicted_load, threshold.lower) then
17:                 scale_down ← calculate_redundant(.)
18:                 ResourceManager.deallocate_resources(slice, scale_down)
19:             end if
20:         end if
21:     end for
22:     Sleep until next interval
23: end while
```

This algorithm is run iteratively based on the time interval configured. Here, during each iteration, all application-oriented slices are monitored, and necessary slice scaling decisions are made. The current QoS and predefined thresholds with upper and lower bounds are retrieved (lines 4-5). Then, a reactive scaling check is performed (line 6), and if the QoS is not within the threshold, a reactive scaling is applied based on the increments configured. With this, the scaling methodology applies an immediate response to the sudden changes within the environment (line 8). On the other hand, if the current conditions lie within the thresholds, the algorithms explore proactive scaling (line 10). For this, we utilise a Long Short-Term Memory (LSTM) model to forecast the future QoS. LSTM is a recurrent neural network (RNN) that retrieves the historical QoS data. With this, the predicted load accompanied by the current resource allocation to the slice is given to the *calculate_required(.)* function, producing the scaling-up decision (lines 11-16). On the other hand, if the predicted load falls below the lower threshold, the redundant resources are identified and released (line 17-19).

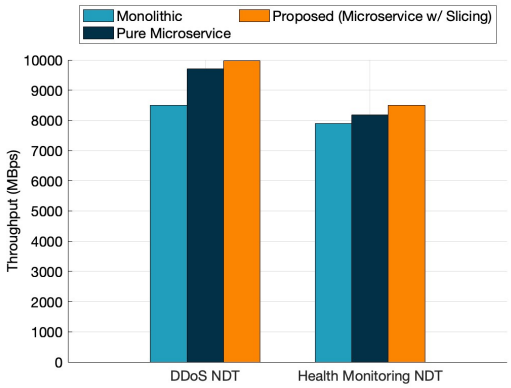

Fig. 2. Throughput Comparison

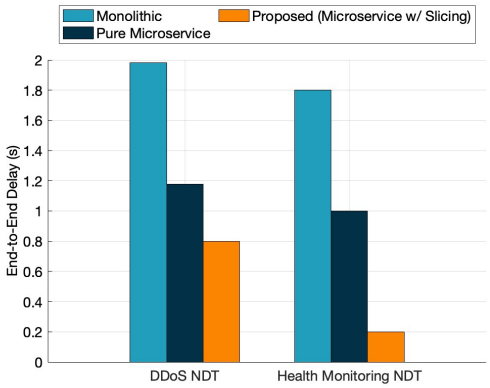

Fig. 3. Delay Comparison

IV. Performance Evaluation

Our experimental setup uses a private cloud infrastructure of HPE ProLiant DL380 Gen11 servers. Here, multiple VMs are hosted, and the microservices are deployed via containers using the runtime interface called containerd. Kubernetes handles the deployment and management of containers within VMs. Here, we deploy two worker nodes per physical machine and three control nodes, where each controller node is deployed to a different physical machine. Here, we provision the resources of both VMs and containers to achieve a certain level of QoS as outlined in Section III.

Then, we consider a large enterprise network monitored and managed by NDTs. In this, we use the network data from our in-house enterprise network. Then, using this, we consider two NDT deployments. One detects DDoS attacks and contains packet captures within the data flows. Moreover, the second deployment is supposed to be used for health monitoring and sending status updates. In both cases, the total amount of data the NDT sets to process is equal, while the payload size of each stream and the number of devices are different. We implement these by considering a monolithic, microservice-based proposed architecture and comparing the results. In the monolithic, no scaling methodology is applied; microservice-based applies reactive scaling and microservice-based with slicing applies the scaling methodology described.

A. Performance Comparison

Firstly, we compare the throughput of the NDTs as shown in Figure 2. Here, the throughput refers to the volume of data processed per second. We use the throughput terminology to measure the system's capability to ingress and process incoming data. Both scenarios show that microservice-based implementations achieve higher throughput. Here, microservice-based enables services to be scaled independently, achieving a higher throughput. On top of this, the proposed architecture isolates different NDTs via a slicing approach, which allows different NDTs to scale independently.

Here, the high volume and processing demands of PCAP data create bottlenecks that worsen the monolithic architecture's performance. As new PCAP data arrives, subsequent packets must wait. For the microservice-based implementation, the application switch can distribute load across multiple instances, reducing the impact of peak traffic. Here, the proposed slicing approach allows the common microservices among NDTs to be utilised jointly, increasing the throughput. For instance, when the new packet capture of the DDoS NDT has not arrived yet, the Health monitoring NDT can use the available resources. Furthermore, here we observe that healthcare monitoring NDT has a lower throughput than the DDoS NDT, which we attribute to the higher number of devices at the latter, causing a bottleneck and congestion.

Moreover, we compare the delay performance in Figure 3; the high volume and processing demands of PCAP data create bottlenecks that worsen the traditional architecture's performance. As the size of the PCAP data is 10 times higher than that of telemetry data, the arrival of PCAP data forces subsequent telemetry packets to wait in a queue. This backlogging increases Health Monitoring NDT's end-to-end delay. Here, the proposed network-slicing allows us to separate the scenario deployments into their own virtual networks, each with its own set of QoS parameters. Here, we implement the load balancer to prioritise the traffic of Health Monitoring NDT. This setup allows PCAP data to be routed to its dedicated microservice, which is configured to handle high bandwidth requirements. Meanwhile, a lightweight, specialised service designed for high-frequency and low-latency data can handle the Health Monitoring NDT. The high volume of PCAP data does not burden this independent microservice; it allows it to reach a lower end-to-end delay to meet the QoS requirements. Here, the slicing approach's benefit is mainly seen in the difference between different types of NDTs. In the microservice-based approach, the services are jointly utilised without isolation between NDTs. This causes the low processing need payloads of Health Monitoring NDT to wait for the DDoS NDT.

Furthermore, we evaluate how the different architectures can meet the QoS requirements. For this, we compare each scenario's QoS requirement violation percentage in Figure 4.

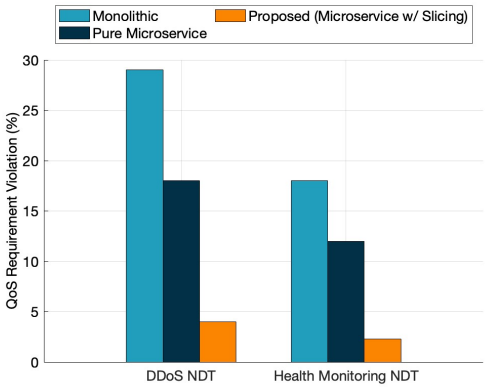

Fig. 4. Violation Comparison

Here, we observe the highest violation rate due to the DDoS NDT processing load. On the other hand, the microservice-based architecture presents a lower rate thanks to the reactive scaling that has been applied. However, the sharing of the underlying resources in this architecture prevents the scaling decisions from being made efficiently. Because the characteristics of the data flows of the different NDTs are different, the decisions fluctuate. On the other hand, the proposed approach offers a level of isolation between NDTs and allows for scaling to be done on a slice basis. Thanks to this, the violation rate is kept to a minimum.

V. Conclusion and Future Works

In this article, we propose a microservice-based architecture with a slicing approach for Network Digital Twins (NDTs) to address the challenges of heterogeneity and quality of service (QoS) requirements. Here, we serve the different processing loads of NDT applications by slicing the network in an application-oriented manner. Then, we deploy this in a private cloud environment for a large-scale enterprise network. Correspondingly, we evaluate the throughput, delay, and QoS requirement violation rates comparatively. The results show that the proposed architecture outperforms the monolithic and pure microservice-based approaches. In future work, we aim to explore implementations over infrastructure that resources are distributed edge-to-cloud continue.

Acknowledgment

This work was partially supported by The Scientific and Technological Research Council of Turkey (TUBITAK) 1515 Frontier R&D Laboratories Support Program for BTS Advanced AI Hub: BTS Autonomous Networks and Data Innovation Lab. Project 5239903.

References

[1] C. Zhou, H. Yang, X. Duan, D. Lopez, A. Pastor, Q. Wu, M. Boucadair, and C. Jacquenet, "Network Digital Twin: Concepts and Reference Architecture," Internet Engineering Task Force, Internet-Draft draft-irtf-nmrg-network-digital-twin-arch-06, Jul. 2024, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/draft-irtf-nmrg-network-digital-twin-arch/06/

[2] "Cisco networking academy connecting networks companion guide: Hierarchical network design." [Online]. Available: https://www.ciscopress.com/articles/article.asp?p=2202410seqNum=4

[3] L. U. Khan, Z. Han, W. Saad, E. Hossain, M. Guizani, and C. S. Hong, "Digital twin of wireless systems: Overview, taxonomy, challenges, and opportunities," *IEEE Communications Surveys Tutorials*, vol. 24, no. 4, pp. 2230–2254, 2022.

[4] Z. Arslan, M. Erel, Y. Özcevik, and B. Canberk, "Sdoff: A software-defined offloading controller for heterogeneous networks," in *2014 IEEE Wireless Communications and Networking Conference (WCNC)*, 2014, pp. 2827–2832.

[5] P. Almasan, M. Ferriol-Galmés, J. Paillisse, J. Suárez-Varela, D. Perino, D. López, A. A. P. Perales, P. Harvey, L. Ciavaglia, L. Wong, V. Ram, S. Xiao, X. Shi, X. Cheng, A. Cabellos-Aparicio, and P. Barlet-Ros, "Network digital twin: Context, enabling technologies, and opportunities," *IEEE Communications Magazine*, vol. 60, no. 11, pp. 22–27, 2022.

[6] Y. Wu, K. Zhang, and Y. Zhang, "Digital twin networks: A survey," *IEEE Internet of Things Journal*, vol. 8, no. 18, pp. 13 789–13 804, 2021.

[7] L. V. Cakir, S. Al-Shareeda, S. F. Oktug, M. Özdem, M. Broadbent, and B. Canberk, "How to synchronize digital twins? a communication performance analysis," in *2023 IEEE 28th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*, 2023, pp. 123–127.

[8] W. Liu, Y. Fu, Y. G. F. L. Wang, W. Sun, and Y. Zhang, "Two-timescale synchronization and migration for digital twin networks: A multi-agent deep reinforcement learning approach," *IEEE Transactions on Wireless Communications*, pp. 1–1, 2024.

[9] K. Duran, L. V. Cakir, A. Fonzone, T. Q. Duong, and B. Canberk, "Digital twin-empowered green mobility management in next-gen transportation networks," *IEEE Open Journal of Vehicular Technology*, vol. 5, pp. 1650–1662, 2024.

[10] Z. Chen, W. Yi, A. Nallanathan, and J. A. Chambers, "Distributed digital twin migration in multi-tier computing systems," *IEEE Journal of Selected Topics in Signal Processing*, vol. 18, no. 1, pp. 109–123, 2024.

[11] P. Bellavista, N. Bicocchi, M. Fogli, C. Giannelli, M. Mamei, and M. Picone, "Exploiting microservices and serverless for digital twins in the cloud-to-edge continuum," *Future Generation Computer Systems*, vol. 157, pp. 275–287, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X24001249

[12] P. Talasila, C. Gomes, P. H. Mikkelsen, S. G. Arboleda, E. Kamburjan, and P. G. Larsen, "Digital twin as a service (dtaas): A platform for digital twin developers and users," in *2023 IEEE Smart World Congress (SWC)*, 2023, pp. 1–8.

[13] L. V. Cakir, M. Özdem, H. Ahmadi, T. Q. Duong, and B. Canberk, "Internet of twins approach: Digital-twin-as-a-platform architecture," *IEEE Internet Computing*, vol. 29, no. 1, pp. 65–74, 2025.

[14] M. Ghaznavi, N. Shahriar, S. Kamali, R. Ahmed, and R. Boutaba, "Distributed service function chaining," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 11, pp. 2479–2489, 2017.

[15] G. S. Uyanik, B. Canberk, and S. Oktug, "Predictive spectrum decision mechanisms in cognitive radio networks," in *2012 IEEE Globecom Workshops*, 2012, pp. 943–947.

[16] E. Ak and B. Canberk, "Fsc: Two-scale ai-driven fair sensitivity control for 802.11ax networks," in *GLOBECOM 2020 - 2020 IEEE Global Communications Conference*, 2020, pp. 1–6.

[17] M. Ariman, G. Seçinti, M. Erel, and B. Canberk, "Software defined wireless network testbed using raspberry pi of switches with routing add-on," in *2015 IEEE Conference on Network Function Virtualization and Software Defined Network (NFV-SDN)*, 2015, pp. 20–21.

[18] A. Rasheed, O. San, and T. Kvamsdal, "Digital twin: Values, challenges and enablers from a modeling perspective," *IEEE Access*, vol. 8, pp. 21 980–22 012, 2020.

[19] M. Mahalingam, D. Dutt, K. Duda, P. Agarwal, L. Kreeger, T. Sridhar, M. Bursell, and C. Wright, "Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks," RFC 7348, Aug. 2014. [Online]. Available: https://www.rfc-editor.org/info/rfc7348

[20] Y. Al-Dhuraibi, F. Paraiso, N. Djarallah, and P. Merle, "Elasticity in cloud computing: State of the art and research challenges," *IEEE Transactions on Services Computing*, vol. 11, no. 2, pp. 430–447, 2018.