frontiers | Frontiers in Communications and Networks

TYPE Original Research
PUBLISHED 04 July 2024
DOI 10.3389/frcmn.2024.1385656

OPEN ACCESS

EDITED BY
Junaid Shuja,
University of Technology Petronas, Malaysia

REVIEWED BY
Ehzaz Mustafa,
COMSATS University Islamabad, Pakistan
Adeel Iqbal,
Yeungnam University, Republic of Korea

*CORRESPONDENCE
Mahfuzulhoq Chowdhury,
mahfuzulhoq.cse05@gmail.com

RECEIVED 13 February 2024
ACCEPTED 10 May 2024
PUBLISHED 04 July 2024

CITATION
Chowdhury M (2024), Accelerator: an intent-based intelligent resource-slicing scheme for SFC-based 6G application execution over SDN- and NFV-empowered zero-touch network. *Front. Comms. Net* 5:1385656. doi: 10.3389/frcmn.2024.1385656

Accelerator: an intent-based intelligent resource-slicing scheme for SFC-based 6G application execution over SDN- and NFV-empowered zero-touch network

Mahfuzulhoq Chowdhury*

Chittagong University of Engineering and Technology, Chittagong, Bangladesh

Zero-touch networks (ZTNs) can provide autonomous network solutions by integrating software-based solutions for various emerging 5G and 6G applications. The current literature does not provide any suitable end-to-end network management and resource-slicing solutions for service function chaining (SFC) and user intent–based (time and cost preference) 6G/non-6G application execution over ZTNs enabled by mobile edge computing, network function virtualization, and software-defined networking. To tackle these challenges, this work initiates an end-to-end network management and user intent–aware intelligent network resource–slicing scheme for SFC-based 6G/non-6G application execution over ZTNs, taking into account various virtual and physical resources, task workloads, service requirements, and task numbers. The results depicted that at least 25.27% average task implementation delay gain, 6.15% energy gain, and 11.52% service monetary gain are realized in the proposed scheme over the compared schemes.

KEYWORDS

network slicing, zero-touch network, resource slicing, mobile edge computing, service function chaining, network function virtualization, blockchain, SDN

1 Introduction

With the growth of mobile devices, virtualized networks receive great attention from researchers due to their ability to provide flexibility and service agility for next-generation applications while incorporating a massive number of IoT devices (Ashraf et al., 2022). According to current statistics, by 2025, there will be over 27 billion IoT devices (Multiple Authors et al., 2022). One important point to note is that managing and orchestrating such a

Abbreviation: SFC, service function chaining; MEC, mobile edge computing; SDN, software-defined networking; NFV, network function virtualization; VNF, virtual network function; THz, terahertz communication; ZTN, zero-touch networks; FeMBB, further enhanced mobile broadband; LDHMC, long-distance and high-mobility communications; umMTC, ultra-massive machine-type communication; URLLC, ultra-reliable low-latency communication; ELPC, extremely low-power communications; XR, extended reality; FW: firewall; DPI, digital packet inspection; AT, network address translation; IDS, intrusion detection; EV, electric vehicle; LB, load balancing.

large number of IoT devices using traditional manual processes is impractical. Zero-touch networks (ZTNs) can address this issue by using software-based solutions instead of hardware-based platforms (Multiple Authors et al., 2022). A ZTN can be defined as a network that provides autonomous network and management operations, as well as end-to-end network programmability for various information and communication technology services, without requiring human intervention (Coronado et al., 2022). The primary goal of ZTNs is to enable autonomous services for current (5G) and future (6G) generation applications, cutting-edge technology infrastructure, computing, caching, routing, resource allocation, and self-healing facilities based on customer demands and available resources.

Currently, ZTNs face several challenges such as proper security measures for network management and application services, automated end-to-end solutions, network resource–slicing facilities for heterogeneous applications by taking into account diverse customer demands (e.g., time-saving and cost-saving demands), proper service coordination for different applications that require services from different technologies such as mobile edge computing (MEC), software-defined networking (SDN), network function virtualization (NFV), service function chaining (SFC), blockchain, federated learning (FL), and efficient resource and work node allocation, among others. To address the high communication latency and bandwidth shortage limitations of traditional centralized cloud computing technology, MEC is viewed as an edge cloud computing technology that offers cloud and storage services at the user network's edge (Tseng et al., 2021). SDN is a networking approach that uses centralized software-based controllers or applications to configure all of the underlying hardware or network elements (VMware, 2024). Instead of using proprietary hardware elements for network services [e.g., firewall (FW) and network address translation (NAT)], NFV technology provides virtualized network services through the use of virtual machines (Red Hat, 2024). NFV enables service providers to run multiple virtual network functions (VNFs) on different servers rather than rely on a dedicated server. SDN and NFV technology both support SFC (a connected chain of network services within an application), which allows connected service functions to be completed sequentially (Chen et al., 2022).

There is currently some work being done in the area of ZTNs. Theodorou et al. (2021) used blockchain technology to automate ZTN service assurance in multi-domain network slicing. However, they only looked at bandwidth prediction accuracy results, not different types of 5G and 6G application execution performance results. Coronado et al. (2022) presented a survey article on various enabling technologies for ZTN-based automated network management solutions, such as SDN, NFV, and artificial intelligence techniques. The authors also discussed some research challenges for ZTN, such as SFC for 5G/6G applications, network slicing and resource allocation, proper work node selection, security and privacy, and appropriate computing and caching solutions for various applications, among others. Xu et al. (2022) developed a Markov game and a reinforcement learning–based optimization solution for wireless power control and spectrum selection in industrial applications. Angui et al. (2022) discussed the challenges for automated cloud radio access networks (RANs) in 6G ZTNs, which included resource discovery, antenna capability, network coverage issues, and computation resource availability. Grasso et al. (2021) proposed a ZTN management technique based on deep reinforcement learning (DRL) for load balancing and computation offloading in an unmanned aerial vehicle (UAV)–aided edge network. Yoshino et al. (2021) developed a multi-service provisioning test bed for zero-touch optical access networks, utilizing access network virtualization technologies and pluggable module-type optical line terminals (OLTs). They did not, however, look into the network slicing–based resource orchestration problem or conduct a performance analysis for 5G and 6G application execution.

Ksentini (2021) investigated the resource management and orchestration operation of ZTN with heterogeneous network slices. However, their work faces significant challenges in terms of quality of service (QoS) guarantee, scalability, and sustainability due to the presence of multiple cross-platforms and domains in B5G/6G systems, which include the RAN network, core network, edge cloud, and remote cloud. Dalgkitsis et al. (2020) proposed a DRL-based VNF placement solution for zero-touch–based 5G networks that incorporate both SDN and NFV technologies. Demchenko et al. (2015) explored automated network services for zero-touch cloud computing applications, such as network slicing and resource management. Mohammadpour et al. (2022) used a ZTN to automate monitoring and traffic generation for virtualized network services. Niboucha et al. (2023) created a zero-touch security management framework for massive machine-type communications (mMTC) network slices in 5G, which include DDoS attack detection. Basu et al. (2022) used a machine learning–based ZTN management framework with dynamic VNF allocation and SFC embedding for 5G applications. Luque-Schempp et al. (2022) developed an automata learning–based smart controller with suitable configuration to predict and satisfy the requirements of time sensitive networking traffic in ZTNs. El Houda et al. (2022) examined the performance of an ensemble learning–based intrusion detection model in SDN-based zero-touch smart grid systems. Wang et al. (2022) describes a framework for optimizing UAV formation and tracking to capture 360-degree views of moving targets in ZTN-based VR applications. Martini et al. (2022) created an intent-based service chain layer for dynamically deploying service chain paths over SDN-based edge cloud networks. Intent-based networking refers to a dynamic or intended approach (i.e., digitized, automated) for network configuration and problem solving rather than a manual process. To reduce network and computation latency, Sebrechts et al. (2022) proposed using a fog-native approach rather than a remote cloud-based approach for executing intent-based workflows. Abbas et al. (2020) used a deep-learning model (e.g., generative adversarial neural network) to manage core and RAN resources.

To that end, SDN is a critical enabling technology for executing user requirements–based tasks over a ZTN. Okwuibe et al. (2021) proposed an SDN-based resource orchestration scheme for industrial IoT application execution, facilitating collaboration among edge and remote cloud networks. In addition to SDN, Wang et al. (2021) identified NFV as a key technology for automated service execution in 5G and 6G applications. However, in order to meet the requirements for SFC in SDN/NFV networks, VNFs must be properly selected and deployed.

To get the most out of an SDN-/NFV-based network with the least amount of delay and cost, SFC (logical or virtual chain) properly connects different service functions one after the other during application execution. These authors demonstrated an SFC with the following VNF execution order: 1) FW, 2) deep packet inspection, 3) encryption, 4) data monitoring, and 5) decryption. Zhong et al. (2019) proposed an SFC solution for NFV-enabled inter-data center networks that considers service financial costs and reliability. However, they did not look into service costs or dependable performance for both 5G and 6G applications.

Lin et al. (2023) present integer linear programming (ILP)–based heuristic algorithms for energy-efficient resource allocation and SFC embedding in NFV networks. Wei et al. (2022) used a quantum genetic algorithm to solve a multi-objective optimization problem for delay-aware resource provisioning and parallel SFC orchestration in NFV networks. To predict the traffic flow rate of SFC in NFV networks, Gu et al. (2019) incorporated an online learning algorithm–based VNF scaling. Pei et al. (2020) proposed a deep learning algorithm for two-phase VNF selection and chaining activities to generate efficient routing paths in SDN/NFV networks. Saha et al. (2020) developed an ILP problem to optimize the number of NFV nodes and IoT devices in SDN/NFV networks. Chen et al. (2022) proposed a Q-learning–based SFC embedding scheme for SDN/NFV-enabled wireless networks to reduce network delay and increase SFC acceptance ratio.

The above discussion mentions network slicing and resource management as critical research challenges for intent-based ZTNs. To offer low latency and high resiliency, Thiruvasagam et al. (2021) proposed a genetic algorithm–based network resource–slicing scheme for multi-connectivity–based and MEC-enabled 5G networks. Feng et al. (2020) developed a Lyapunov optimization–based short- and long-timescale bandwidth allocation scheme for 5G ultra-reliable low-latency communications (URLLC) and enhanced mobile broadband (eMBB) application execution to provide energy and cost-efficient solutions for RANs. To reduce latency and energy consumption, Tang et al. (2021) proposed a DRL slice selection for computation offloading operations in vehicular networks. Brik et al. (2020) proposed an FL-based approach for predicting service-oriented key performance indicators (KPIs) for 5G networks. Chergui et al. (2021) presented a statistical FL method for slice-level KPI prediction in energy-efficient 6G networks.

However, previous research on ZTNs with or without cloud, SDN, and NFV technologies did not present a task execution performance analysis that considered both 6G and non-6G/5G applications. For example, Salameh et al. (2022) mentioned three main types of 5G applications: (i) eMBB (e.g., video streaming and immersive gaming applications via HoloLens), (ii) mMTC (e.g., smart video surveillance and smart agriculture via IoT and cloud computing technologies such as optimal plans for irrigation or fertilizer frequency determination based on plant health), and (iii) URLLC (e.g., industrial automation, circular manufacturing, and collaborative human–robot interaction-based applications). Alwis et al. (2021) identified and discussed several promising 6G applications with their task execution requirements. They categorized 6G applications as follows: (i) further-eMBB (FeMBB) [e.g., metaverse-based social avatar applications, holographic telepresence, and haptic feedback-based extended reality (XR) applications], (ii) long-distance high-mobility communications (LDHMC) (e.g., high-speed railway applications, space travel, and deep-sea sightseeing applications), (iii) extremely URLLC (eURLLC) application (e.g., FL-based fully automated driving applications), (iv) extremely low-power communication (ELPC)–type application (e.g., blockchain, IoT, and digital twin–based electronic healthcare), (v) ultra-massive machine-type communications (umMTC) (e.g., wireless power transfer, electronic vehicle charging, and brain–computer interface-based applications such as wheelchair control by using brain signals) application (Rico-Palomo et al., 2022).

1.1 Gaps or limitations in existing studies

Based on the previous research-work discussion, it is clear that earlier related works did not investigate a proper intent-aware and service requirement–aware intelligent network resource–slicing scheme for SFC based on both 6G and non-6G application execution over SDN-, NFV-, and MEC-driven ZTN. Without an appropriate network resource–slicing scheme and proper virtual resource node selection, SFC-based 6G and non-6G applications over the ZTN may experience significant SFC execution time latency, user energy expense latency for SFC execution, service execution monetary cost, lower QoS satisfaction ratio, and low throughput, among other things.

Furthermore, the question of how to coordinate SDN and NFV technology, as well as edge cloud technology, for SFC-based 6G and non-6G application execution in ZTNs is beyond their scope. Furthermore, their analyses excluded time-first and cost-first intent-based 6G and non-6G service provisioning for SDN- and NFV-based ZTNs. Furthermore, previous research did not present any suitable network infrastructure for time-first and cost-first service-aware resource slicing for SFC-based 6G and non-6G application execution over SDN- and NFV-based ZTNs. Existing works did not investigate latency, QoS satisfaction, and cost performance analysis by taking SFC for multiple 6G applications such as metaverse-based social avatar applications, holographic telepresence, haptic feedback-based XR applications, high-speed railway applications, FL-based fully automated driving applications, blockchain, IoT, and digital twin–based electronic healthcare, wireless power transfer, electronic vehicle charging, and brain–computer interface-based wheelchair control by using brain signals.

Similarly, the existing works did not investigate performance analysis for multiple non-6G applications such as video streaming, immersive gaming applications via holoLens, smart video surveillance, and smart agriculture via IoT and cloud computing technologies such as optimal plans for irrigation or fertilizer frequency determination based on plant health, or industrial automation work such as circular manufacturing or collaborative human–robot interaction-based applications.

The SFC-based task implementation delay analysis associated with various existing works did not take into account different delays such as network inauguration phase delay, user request and resource gathering phase delay, network slicing phase delay, and task work realization delay (all of which include computation, caching, communication, and waiting time). Existing research did not provide a suitable mathematical model that included task implementation delay, energy expense, QoS guarantee ratio, achievable throughput, service execution monetary cost for users and service providers, service provider profit, user and service

provider welfare, alive node number, and survived energy amount for users, among other things.

1.2 Motivations and contributions of our work

To tackle these existing issues, this article proposes a service requirement–aware intelligent network resource–slicing scheme (i.e., accelerator) for SFC-based multiple 6G and non-6G application execution over SDN-, NFV-, and MEC-based and intent-driven ZTNs. Previous research works did not create an end-to-end network management system with proper resource allocation procedures for 6G and non-6G application execution over SDN- and NFV-based ZTNs. SDN- and NFV-based 6G and non-6G applications require proper resource allocation and network systems to meet diverse requirements, such as low task implementation deadlines, time preferences, and cost preferences. Because of a lack of intelligent network architecture and resource-slicing schemes, the current scheme incurs significant task implementation delays, energy costs, and service execution expenses.

The aforementioned limitation motivates us to present a resource-slicing scheme that maximizes the task implementation delay gain, energy gain, and monetary cost gain for SFC-based multiple 6G and non-6G application executions over the ZTN. Our work's main innovation and contribution is that it develops a resource-slicing scheme (for both communication and computation resources) that considers both time-priority and cost-priority service requirements for various applications. Furthermore, unlike previous research, this paper investigates ZTN performance for both 6G and non-6G application execution by taking different resource and task types into consideration. For the first time, it brings together SDN, NFV, blockchain, IoT, and MEC technologies to enable ZTN-based application execution. The significant contributions of this work are mentioned below:

- This work inaugurates an intent-based (time preference first and cost preference first) network resource–slicing scheme for different SFC-based 6G and non-6G applications by considering different virtual and physical resources, digital twin, blockchain, edge computing, caching, and FL services and different SFC workloads, different task data sizes, different service execution budgets, energy values, service execution deadlines, task count, and available resource statuses.
- This work develops an intelligent virtual and physical work node (e.g., NFV, cloud server) assignment along with a network resource (bandwidth) assignment scheme for different 6G application execution (e.g., metaverse, holographic telepresence, XR applications, FL, blockchain, IoT, digital twin, and brain–computer interface-based applications) and different non-6G application execution (e.g., video streaming, smart video surveillance, and industrial automation) over MEC-, SDN-, and NFV-enabled ZTNs.
- This work provides an intelligent network model that incorporates SDN technology, NFV technology, blockchain, digital twin, FL, MEC technology, and wired and wireless networks, along with different user devices [e.g., mobile phones, XR devices, holographic telepresence screens, haptic feedback sensors or devices, brain sensors, health sensors, robots, IoT devices, video cameras, and electric vehicles).
- The primary goal of the proposed resource-slicing scheme is to maximize task implementation delay gain, energy gain, and monetary gain for various 6G and non-6G application executions over a ZTN. This work introduces an accelerator algorithm that coordinates application execution steps and appropriate resource selection (time slot, work nodes, computing, and communication link) for both time-first and cost-first SFC application (6G and non-6G) execution over ZTNs.
- This paper delivers a mathematical analysis model for 6G and non-6G application execution over ZTNs, which includes task implementation delay, energy expense, QoS guarantee ratio, achievable throughput, service execution monetary cost for users and service providers, service provider profit, and user and service provider welfare. Unlike previous works, our task implementation delay includes additional delays such as network inauguration, user request and resource gathering, network slicing, and task work realization delay (such as computation, caching, communication, and waiting delay).
- To demonstrate the suitability, the proposed accelerator scheme simulation results (for both time-first and cost-first schemes) are presented with proper analysis in the Simulation results and analysis section, along with a performance comparison with the traditional scheme.

Next, Section 2 includes the related works. The proposed accelerator scheme is depicted in Section 3 with an algorithm, working steps, and network model. Section 4 holds the mathematical analysis model that includes different performance metrics. The simulation results are investigated in Section 5. The conclusion associated with the proposed scheme is highlighted in Section 6.

2 Related works

This section discusses the existing literature on SDN-, NFV-, and MEC-enabled ZTNs. Ma et al. (2022) developed a zero-touch management scheme for IoT devices using digital twins. Brik et al. (2020) proposed a FL-based approach to predict network slice performance for 5G applications. Boškov et al. (2020) introduced a software-enabled access point and a Bluetooth-based automated zero-touch service provisioning solution for IoT devices.

To enable automated network fault management services, Sousa and Rothenberg (2021) discussed a closed loop–based ZTN management framework. Yoshino et al. (2021) discussed the feasibility of automated line opening and zero-touch provisioning–based multiple service provisioning with pluggable module-type OLT for access network virtualization. Liyanage et al. (2022) presented a detailed survey regarding the ZTN and service management concept, architectures, components, and key technical areas. Shaghaghi et al. (2021) discussed a DRL-based and age-of-information–aware failure recovery scheme for an NFV-enabled ZTN. Coronado et al. (2022) presented a survey regarding ZTN management solutions that included both automated and zero-touch management techniques for both wireless and mobile networks. To provide scalable and fast ZTN-slicing operations and service provisioning, Roy et al. (2022) presented a cloud-native and service-level agreement (SLA)–driven stochastic FL policy. To ensure proper execution of industrial IoT applications, Lin et al. (2022) presented a machine learning-based end-to-end solution for ZTN-based traffic

TABLE 1 Comparison with existing works.

Scheme	Considered both network and mathematical model for ZTN	Considered SDN, NFV, MEC, FL, BC, and DT technology	Both 6G and non-6G application performance	Minimize delay, energy cost, and monetary cost	Service requirement–aware and intent-based task and resource scheduling for ZTNs
Ma et al. (2022)	Not considered	Not considered	Not considered	Not considered	Not considered
Boškov et al. (2020)	Not considered	Not considered	Not considered	Not considered	Not considered
Sousa and Rothenberg (2021)	Not considered	Not considered	Not considered	Not considered	Not considered
Yoshino et al. (2021)	Not considered	Not considered	Not considered	Only minimize time delay	Not considered
Shaghaghi et al. (2021)	Not considered	SDN and NFV considered	Only 6G application considered	Not considered	Not considered
Roy et al. (2022)	Not considered	MEC and FL considered	Only 6G application considered	Only time delay considered	Not considered
Lin et al. (2022)	Not considered	MEC only considered	Only 6G application considered	Only time delay considered	Not considered
Martini et al. (2022)	Not considered	SDN, NFV, and MEC considered	Only 6G application considered	Not considered	Not considered
Abbas et al. (2020)	Not considered	NFV and MEC considered	Only non-6G application considered	Only time delay considered	Not considered
Theodorou et al. (2021)	Not considered	NFV, BC, and MEC considered	Only non-6G application considered	Not considered	Not considered
Xu et al. (2022)	Not considered	NFV, SDN, and MEC considered	Only non-6G application considered	Not considered	Not considered
Lin et al. (2023)	Not considered	NFV and MEC considered	Only non-6G application considered	Not considered	Not considered
Our proposed accelerator scheme	Yes, considered all	Yes, considered all	Yes, considered all	Yes, considered all	Yes, considered all

steering and fault management issues. Sebrechts et al. (2022) developed a fog native architecture for microservice application provisioning and workflow management in intent-based networking.

To design loss functions in regression tasks, Collet et al. (2022) presented a deep learning-based prediction scheme for intent-based networking. Okwuibe et al. (2021) presented constraint satisfaction problem solving based on resource orchestration scheme, in which SDN is used as an orchestrator for industrial IoT application execution in collaborative edge cloud networks. Alwis et al. (2021) discussed a detailed survey regarding 6G applications, requirements, technologies, 6G enablers, and research challenges, among others. Rico-Palomo et al. (2022) discussed several new services for the 6G ecosystem, such as FeMBB, LDHMC, eURLLC, ELPC, and umMTC. Salameh et al. (2022) discussed different challenges, technologies, and applications related to both 5G and 6G networks. Song et al. (2020) discussed constrained Markov decision process (CMDP)–based network-slicing solutions for different types of 5G applications (e.g., eMBB, URLLC, and mMTC). To maximize the network's long-term throughput, Suh et al. (2022) investigated a DRL-based network slicing solution for B5G applications. Adhikari et al. (2022) proposed a cybertwin-driven DRL scheme for dynamic resource provisioning in 6G edge computing networks. Cao et al. (2021) presented a resource availability–based SFC scheduling scheme for 6G application execution with virtualization. Alsabah et al. (2021) discussed a comprehensive survey regarding the 6G vision, key enabling technologies, key applications, and technical challenges for the 6G wireless communication networks. Thiruvasagam et al. (2021) presented a failure-resilient resource orchestration and network-slicing scheme for multi-connectivity and MEC-empowered 5G networks.

To guarantee latency and reliability, Feng et al. (2020) discussed a Lyapunov optimization–based resource scheduling scheme for 5G URLLC and eMBB services. To optimize latency and energy cost, Tang et al. (2021) presented a DRL-based slice selection and computation offloading framework for vehicular networks. By leveraging both SDN and NFV technologies, Hermosilla et al. (2020) presented a dynamic security management framework in MEC-powered UAV networks. Sun et al. (2020) developed a breadth-first search–based SFC optimization scheme. Lin et al. (2023) presented an energy-aware SFC-embedding scheme in NFV networks. Zhang et al. (2019) investigated the longest common sequence (LCS)–based flexible framework for SFC executions. To maximize the utility value for SFC embedding, a Markov chain–based optimization scheme was presented by Lin et al. (2022). Saha et al. (2023) utilized the Brown–Gibson model for

efficient cloud service provider selection for different IT-based applications. Tianran et al. (2023) presented a reputation-based collaborative intrusion detection system (IDS) using blockchain technology. Huang et al. (2023) used fuzzy C-means clustering and a bat optimization algorithm for optimizing IoT-based smart electronic services. Chowdhury (2022) highlighted an energy-harvesting and blockchain-aware healthcare task coordination policy for IoT-assisted networks. Fathalla et al. (2022) presented a preemption choice–based physical machine allocation policy for cloud computing tasks. Chen et al. (2023) discussed a non-cooperative game-based computation task offloading policy for MEC environments. A multi-objective–based evolutionary algorithm was presented by Wang et al. (2022) for joint task offloading operations, power, and resource allocation in MEC-based network. Chen et al. (2019) presented a distributed deep learning–based parameter updating and synchronization model for a video surveillance system. Hu et al. (2021) formulated a coalition game for multi-customer resource procurement in the cloud computing environment. However, most of the aforementioned related works (e.g., that of Fathalla et al., 2022; Chen et al., 2023; Wang et al., 2022; Chen et al., 2019; Hu et al., 2021) are limited only to a single type of MEC task rather than both SFC-based 6G and non-6G application execution. They also did not utilize multiple technologies such as SDN, NFV, IoT, and MEC at the same time for different intents (e.g., time-first and cost-first) based on resource-slicing policies for ZTN-based 6G and non-6G application execution.

To predict the VNF flow rate, Gu et al. (2019) proposed an online learning algorithm for SFC execution. To minimize the total SFC embedding cost, Chen et al. (2021) formulated mixed ILP (MILP)–based VNF mapping and scheduling problems in edge cloud networks. Zhou et al. (2019) presented a bidirectional offloading scheme for SFC- and NFV-enabled space–air–ground integrated networks. To minimize the latency of all SFC requests and satisfy the service level agreements, Tamim et al. (2020) utilized an MILP model for SFC placements in NFV networks. To lower the rejection rate in terms of SFC request execution, Mohamad et al. (2022) discussed a prediction-aware SFC placement and VNF-sharing scheme. To satisfy the application execution requirements, Tseng et al. (2021) utilized the MEC server for VNF placement and scheduling decisions for augmented reality application execution in NFV networks. Hantouti et al. (2020) presented a detailed survey regarding SFC execution in 5G networks that includes several use cases, key enabling technologies, and potential research problems. Zahoor et al. (2022) identified different research challenges and potential solutions associated with network slicing for 5G applications. Zahoor et al. (2023) discussed the performance evaluation of hypervisor-based virtualization technologies for NFV deployment.

Table 1 depicts the comparison between the proposed scheme and existing schemes. The existing work did not investigate both 6G and non-6G application execution for MEC-, SDN-, and NFV-empowered ZTNs. They also did not investigate proper resource-slicing schemes by taking into account both service requirements and different intents (time-first and cost-first schemes) for ZTN-based applications. In differing from the existing works, this article presents a service-aware and double intent–based (time-first and cost-first) network resource–slicing scheme for SFC-based 6G and non-6G application execution over MEC-, SDN-, and NFV-empowered ZTNs.

1: **for** network slicing manager **do**
2: sends the network's first beacon message to users.
3: gets UCR (internet connectivity request) and UMSR (service registration message) from the users and dispatches URR (internet connectivity response) and UAA (user authentication and service registration response) to the users
4: broadcasts SMTRS message (user slot assign for task request dispatch) to users and receives request messages during UTR slot from users
5: sends RURS (request for resource update) message to resource and work nodes and gets resource-update response message (URIS) from work nodes. Sends ISDR message (inter SDN/slicing controllers request message for task/resource node information) and receives ISRES message (response message) from other slicing managers
6: computes resource slicing and SFC scheduling information during CNRS slot and sends NSD message to users (selected time slot and resource information) and RSD (resource/time slot information) to selected work nodes
7: **if** SFC request = = 6G application **then**
8: executes before the non-6G application. Offers resources to the time-first task before the cost-first priority task for all 6G tasks (FeMBB, eURLLC, umMTC, LDHMC, and ELPC).
9: again sorts each time-first/cost-first priority task based on their shortest task time limit.
10: selects the best resources (physical and virtual work node with network communication link) for each sorted time-first priority task with the lowest predicted task implementation delay (min Δ^i_{tid}). Selects the best resources (physical and virtual work node with network communication link) for each sorted cost-first priority task with lowest predicted user service execution monetary value (min μ^i_{secu}) basis with min task implementation delay (min Δ^i_{tid}) among lowest cost resources
11: **else if** SFC request = = non-6G application **then**
12: executes after the 6G application. Offers resources to the time-first task request first before the cost-first priority-based task for all non-6G task types (URLLC, eMBB, and mMTC).
13: again sorts each time-first/cost-first priority task based on their shortest task execution time limit. Selects the best resources for each sorted time-first priority task with lowest predicted task implementation delay (min Δ^i_{tid}) and for each sorted cost-first priority task with the lowest predicted user service execution monetary value (min μ^i_{secu}) basis with minimum possible task implementation delay (min Δ^i_{tid})
14: **end if**
15: Go to step 1
16: **end for**

Algorithm 1. Proposed accelerator-based algorithm.

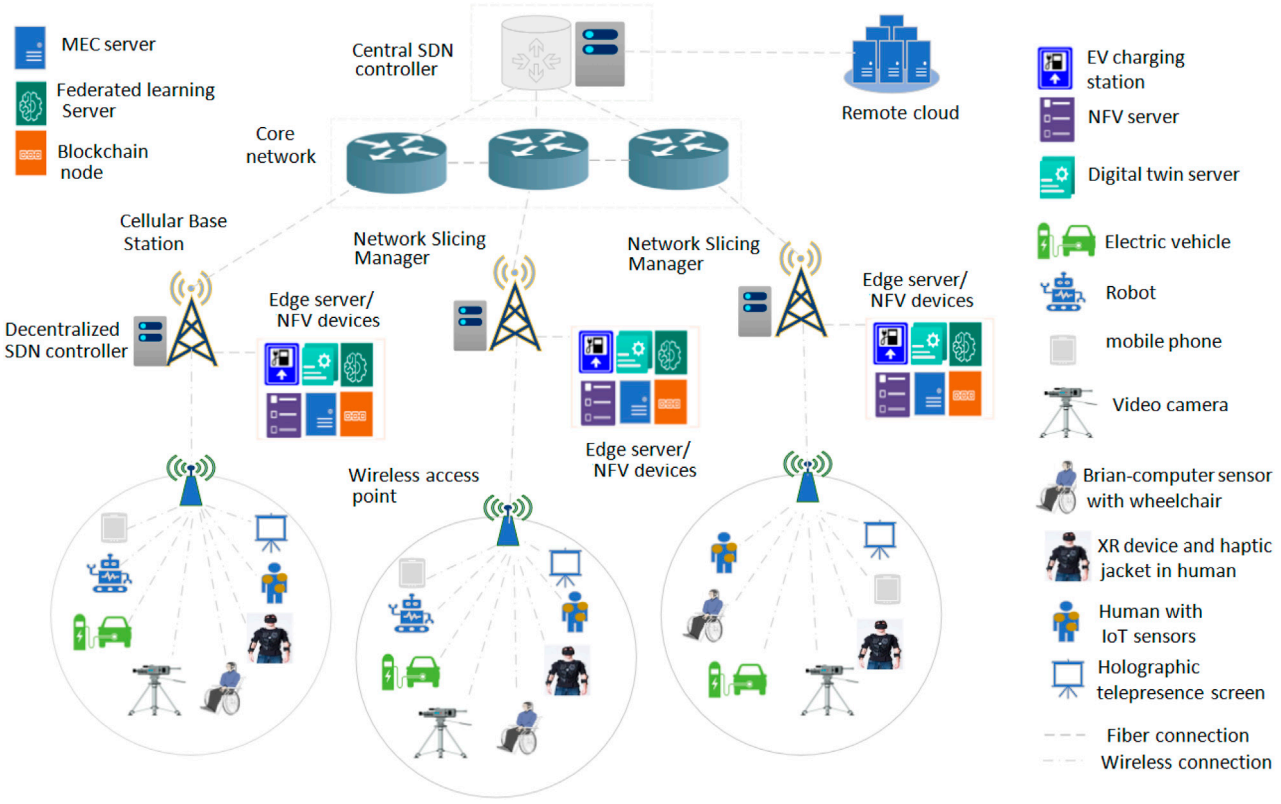

FIGURE 1
Proposed SDN-/NFV-empowered ZTN model for 6G and non-6G application execution.

3 Proposed accelerator-based network slicing for ZTN

3.1 Network model and considerations

Figure 1 represents the network model for the SFC-based 6G and non-6G application execution over the SDN-/NFV-empowered ZTN. The virtualized work nodes (e.g., MEC and caching devices, FL server, blockchain server, NFV server, and digital twin server) have resided near the cellular base station. The service requests that are from user nodes (e.g., robot, mobile phone, electric vehicles, video camera, brain–computer sensors, XR users, IoT-based electronic health users, haptic devices such as haptic jackets or glass, and holographic screens) are located within the coverage range of cellular base stations and wireless access points. The SFC task request applications are generated by the user devices and dispatched to the network slicing manager at the cellular base station. The network slicing manager decides the best work node selection (e.g., virtual and physical work nodes) for each user task implementation. The user devices can be attached to the internet via both the wireless cellular base station and WLAN access point devices. Three different types of wireless communication links are available. They are terahertz communication (IEEE 802.15.3d based, bandwidth of 5 THz, link range 1–10 m), microwave communication (IEEE 802.11b based, link range 1–300 m, bandwidth 7.2 GHz), and millimeter/millimeter wave (mmWave) link (IEEE 802.11ad based, link range 1–50 m, bandwidth 1.25 GHz). Along with the cellular link, the user devices can transfer their data by using the WLAN link (IEEE 802.11be-based data transfer, 2.4/5/6 GHz radio frequency). The best wireless communication link is selected by the network slicing manager for data transfer based on link availability and the data transfer rate.

The decentralized SDN controller is located near the cellular base station (network slicing manager) that offers NFV monitoring, routing path selection, automatic network device configuration, and node/link failure monitoring purposes. The central SDN controller is located three or four hops away from the decentralized SDN controller. The central SDN controller monitors and manages the network resource status and device configuration centrally. The decentralized SDN controller (at the cellular base station) can perform their work by receiving the central SDN controller's command and can contact the central SDN controller regarding any query associated with the network node, resource status, or remote services. The connectivity between the cellular base station/core network and the core network/central SDN controller is done via an IEEE 802.3cd-based dedicated fiber link. Similarly, connectivity between the cellular base station/central SDN controller and edge servers/remote cloud servers is offered via the IEEE 802.3cd-based fiber communication link. The edge server located near the cellular base station contains different types of virtualized devices, such as MEC and caching servers, FL servers, blockchain devices, digital twin servers, and NFV servers. The electronic vehicle (EV) charging station can be located within the cellular base station/WLAN access point communication range. In this paper, initially, the SFC task execution request is dispatched

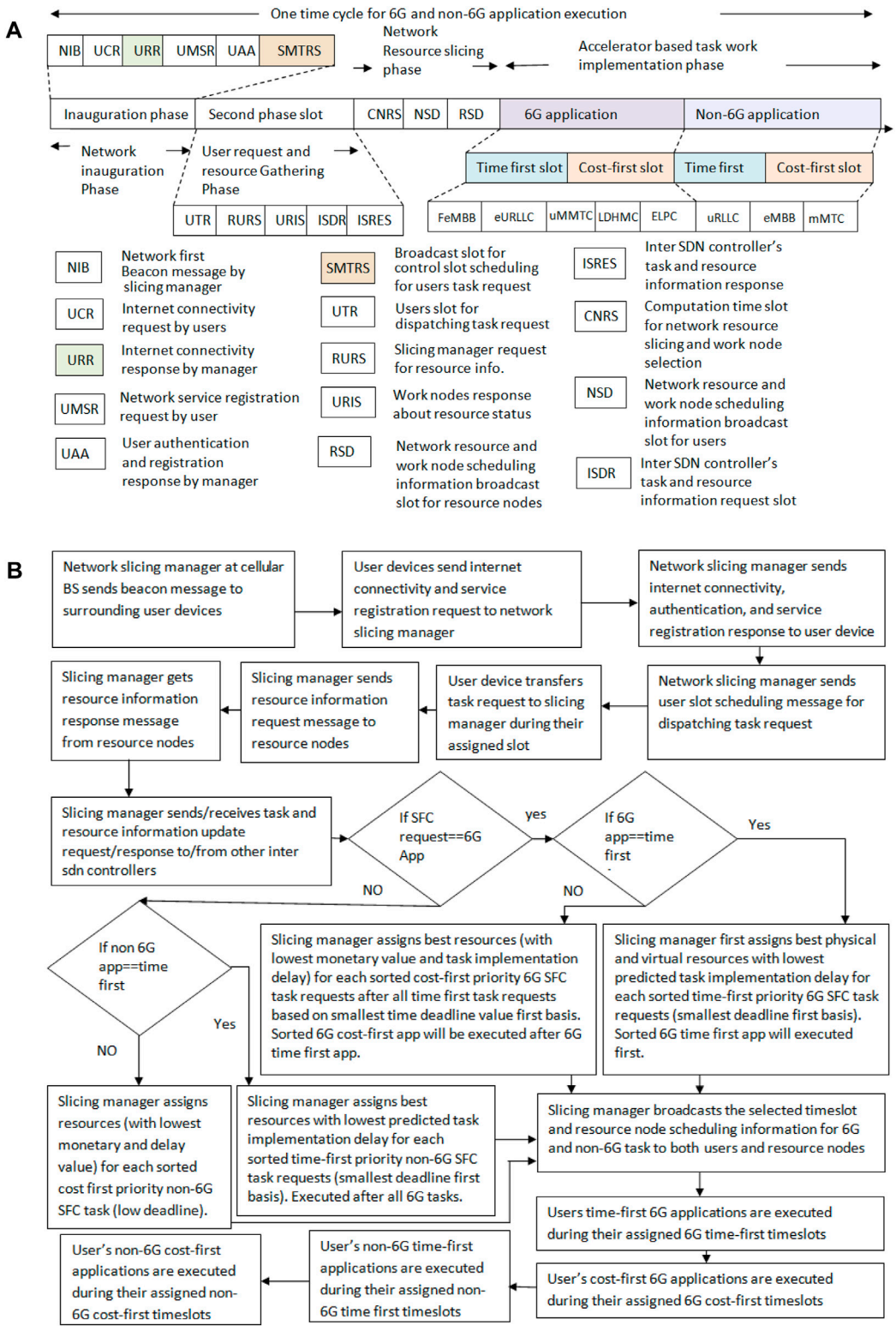

FIGURE 2
(A) Proposed timing model; **(B)** overall pipeline of the proposed model.

from the user nodes to the network slicing manager. The slicing manager collects the resource and task information from the work nodes (virtual and physical devices) and users. After that, the slicing manager selects a suitable communication time slot along with the best work nodes (virtual and physical work nodes) for users' different 6G and non-6G application executions. After the task processing or implementation, the users receive the task result from the work nodes via the wireless or wired communication links.

3.2 Accelerator-based network-slicing scheme and work node selection scheme

This section elaborately discusses all steps associated with the proposed network slicing scheme. Figure 2A highlights the timing model, and Algorithm 1 shows the proposed accelerator-based work node selection scheme. Figure 2B shows the overall pipeline of the proposed model. As shown in Figure 2A, our proposed accelerator scheme includes four phases. The first phase is the network inauguration phase. The second phase is the user request and resource information gathering phase. The third phase is the network slicing phase. The fourth phase is a different 6G- and non-6G-based task realization or implementation phase. At the first network inauguration phase, the network slicing manager (cellular base station) first transmits the beacon messages (NIB messages) to the surrounding users. The users who receive the beacon messages send or transfer an internet connectivity request message (UCR) to the network-slicing devices. The network slicing manager sends an internet connectivity response (URR) to users. Next, the users prepare and send a network service registration (UMSR) message to the slicing manager that includes registration requests for different services such as blockchain, computing, caching, FL, SFC execution, EV charging and sharing, and digital twin–based prediction services. After that, the network slicing manager dispatches user authentication and registration response messages (UAA) to the users. Moreover, the slicing manager schedules a control slot for users for their 6G and non-6G application requests and dispatches SMTRS (scheduling message task request control slot) to the users.

Next, the second phase of our proposed scheme becomes operable (i.e., the user request and resource gathering phases). After that, the user dispatches or sends the 6G and non-6G application or task execution service requests to the slicing manager during their UTR (user slot for dispatching task requests) slot. The scheduling manager next dispatches a RURS (resource information request slot) message to the resource nodes or workers. The work nodes and resource devices send a URIS message (a work node response about their resource status) to the slicing manager. It can be noted that only the URIS message within a time deadline will be accepted. After the time deadline, the remaining URIS message will not be accepted. Similarly, RURS messages are sent to multiple work nodes and resource nodes. If one RURS message is unsuccessful, the slicing manager will try to send another RURS message within the RURS message exchange time deadline (set before the time cycle begins). If all the RURS messages are unsuccessful, the slicing manager will rely on its previous resource information for scheduling (step 5 of Algorithm 1). Similarly, if one URIS message is unsuccessful, the slicing manager will rely on the other nodes' URIS response messages for resource scheduling.

Next, the host decentralized SDN controller that resides within the network slicing manager sends an ISDR message (inter SDN controller query message about information like resource link or work node and routing information, task, and resource scheduling information) to the other decentralized SDN controller and central SDN controller. The other SDN controller sends an ISRES message (other SDN controllers respond to a query like task scheduling information for a network slicing node) about their resource scheduling information and other resource link or node status. After that, the third phase starts (i.e., the network resource–slicing phase). In this phase, by using the collected information (e.g., different resource node status, other slicing managers' task scheduling information, available virtual and physical resources, and 6G and non-6G task requests), the host slicing manager at the cellular base station completes the SFC, work node, task scheduling, and resource-slice assignment process during the CNRS slot time (i.e., the computation time slot for network resource slicing and work node selection for 6G and non-6G applications). In this work, two different schemes are used for resource slicing. The first one is a time-first scheme and the second one is a cost-first scheme. The user can select one approach during their task request message dispatch process. For the time-first scheme, the suitable virtual/physical work node and wired/wireless link combination are selected for each SFC task request based on the lowest predicted task implementation delay (min Δ^i_{tid}) with the highest QoS guarantee ratio (i.e., task deadline or task execution time limit satisfaction). The cost-first scheme receives resources after the time-first scheme based on users.

For the cost-first scheme, the suitable virtual/physical work node and wired/wireless link combination are selected for each SFC task request based on the lowest predicted user service execution monetary value (min μ^i_{secu}) with minimum task implementation delay (min Δ^i_{tid}) and the highest QoS guarantee ratio (i.e., task time limit satisfaction). The readers may look into Sections 4.1 and 4.5 for the Δ^i_{tid} and μ^i_{secu} calculations, respectively. After resource slicing or scheduling completion, the network slicing node dispatches NSD (i.e., network resource and work node scheduling information for users) messages to the user devices. Next, the network slicing node dispatches an RSD (network resource and work node scheduling information for resource nodes) message to the resource nodes. The last phase of our proposed scheme is the accelerator-based task realization or implementation phase. In this phase, the work node executes the task work during the assigned computation slot, and the user device dispatches or receives task input and output data during their assigned communication slot. The user's 6G and non-6G time-first/cost-first tasks are executed during their assigned 6G application and non-6G time-first/cost-first time slot (that includes both communication and computation time slots), respectively.

It can be noted that in this work, different 6G and non-6G application execution performances are analyzed. The considered 6G applications are metaverse-based social avatar applications, holographic telepresence, haptic feedback-based XR applications, high-speed railway applications, FL-based fully automated driving applications, blockchain, IoT, and digital twin–based electronic healthcare, wireless power transfer, electronic vehicle charging, and brain–computer interface-based wheelchair control by using brain signals. The considered 6G applications are video streaming, immersive gaming applications via the holoLens, smart video surveillance, and smart agriculture via IoT and cloud computing technologies, such as optimal plans for irrigation or fertilizer frequency determination based on plant health, and industrial automation work such as circular manufacturing or collaborative human–robot interaction-based applications.

For example, during a 6G application time slot (i.e., brain–computer interaction-based wheelchair movement application), the selected work node and users have to perform several activities. Initially, the user's task request is transferred to the slicing manager for the brain–computer interaction-based

application. Before receiving the task request, the slicing manager executes different VNFs, such as FW, digital packet inspection (DPI), and NAT. Next, the slicing manager sends the task implementation instructions to the selected work nodes (users' head sensors and sensing devices, MEC server) and user devices. After receiving the task instructions, the head-sensing devices capture the users' brain signals [via electroencephalography (EEG) and functional magnetic resonance imaging (fMRI)] and offload the captured data to the MEC server for processing. Before receiving the dispatched data, the MEC server executes two VNFs, such as IDS and NAT. Next, the MEC/virtual server performs processing of brain signals from the collected data and signals, feature extraction, pattern recognition, and translation commands (from brain signals). After that, the MEC/virtual server sends a processed command to the user's wheelchair. Before receiving the processed result, the wheelchair device performs FW, NAT, and IDS operations. Then, the user's wheelchair device operates or moves based on MEC-processed commands from the brain signal. After the completion of these activities, the brain–computer interaction-based wheelchair movement work is complete. Furthermore, during a non-6G application time slot (e.g., IoT-based smart agriculture assistance), the work node and users have to perform several activities. Initially, the user's task request is transferred to the slicing manager for the IoT-based smart agriculture assistance application. Before receiving the task request, the slicing manager executes different VNFs, such as FW, DPI, and NAT. Next, the slicing manager sends the task implementation instructions to the selected work nodes (IoT sensors and sensing devices in the agriculture field, MEC server) and user devices. After receiving the task instructions, the IoT devices and sensors collect different crop and environment data (e.g., crop image, humidity, soil data, moisture, temperature) and dispatch or offload the captured data to the MEC server for processing. Before receiving the dispatched data, the MEC server executes two VNF operations, such as IDS and NAT. Next, the MEC/virtual server performs captured data processing and produces irrigation and fertilization frequency plans for farmers based on the crop data. After that, the MEC/virtual server sends processed data (task-processing results regarding irrigation/fertilization frequency plan) to the user devices. Before receiving the processed result, the user device performs FW and IDS VNF operations. Next, the MEC-processed task result data are visualized on the screen or used on a mobile device. It can be noted that in Figures 2B, if any message dispatch activity or task execution is unsuccessful, the slicing manager will consider only successful tasks or messages. The unsuccessful task will not be included in the performance evaluation process. For resource scheduling, the slicing manager will rely on its own information along with the work node information.

4 Mathematical model

This section presents the important performance metrics calculation model with a proper explanation. The considered performance metrics are task implementation delay, energy expense, service execution monetary cost for users and service providers, service providers' profit, and survived energy amount for users. First, we will discuss the average task implementation delay.

4.1 Task implementation delay

The task implementation delay calculation (Δ^i_{tid}) includes all delays associated with network inauguration phase (Δ^i_{nid}), resource information gathering phase (Δ^i_{urg}), network slicing phase (Δ^i_{nsd}), and task work realization delay (Δ^i_{twrd}). The average task implementation delay for the total number of tasks $\left(\Delta^i_{atid} = \frac{\sum_{i=1}^{y}\Delta^i_{tid}}{y}\right)$ is investigated by $\Delta^i_{atid} = \frac{\sum_{i=1}^{y}\Delta^i_{tid}}{y} = \frac{\sum_{i=1}^{y}\Delta^i_{nid}+\Delta^i_{urg}+\Delta^i_{nsd}+\Delta^i_{twrd}}{y}$, where the network inauguration phase delay is Δ^i_{nid}. The user request and resource information gathering phase delay is Δ^i_{urg}. The network slicing phase delay is Δ^i_{nsd}. y is the total number of users with 6G and non-6G task requests. Δ^i_{twrd} is the total arrived 6G and non-6G task work realization delay. The network inauguration phase delay (Δ^i_{nid}) includes an initial beacon transfer delay, a network connectivity request and response delay, a network service registration and response delay, and a control slot allocation delay. Δ^i_{nid} is computed by using Eq. 1 as follows:

$$\begin{aligned}\Delta^i_{nid} = {} & \frac{\Gamma^i_{nib}+\Gamma^i_{ucr}+\Gamma^i_{urr}+\Gamma^i_{umsr}+\Gamma^i_{uaa}+\Gamma^i_{smtrs}}{\delta_{wl}} * z_{wh} \\ & + \frac{\Gamma^i_{nib}+\Gamma^i_{ucr}+\Gamma^i_{urr}+\Gamma^i_{umsr}+\Gamma^i_{uaa}+\Gamma^i_{smtrs}}{\delta_{fl}} * z_{fh} \\ & + \frac{\gamma^i_{nmwip}}{\Omega_{cp}} + \frac{\gamma^i_{uwip}}{\Omega_{lp}} + \Theta^i_{pnd} + \Theta^i_{wgd}.\end{aligned} \tag{1}$$

$\Gamma^i_{nib}, \Gamma^i_{ucr}, \Gamma^i_{urr}, \Gamma^i_{umsr}, \Gamma^i_{uaa}$, and$\Gamma^i_{smtrs}$ are the network inauguration phase beacon message size, user-to-slicing manager internet connectivity request message size, connectivity response message size, user device–based service registration and access request, network slicing manager–based user authentication, service creation, and user approval message size, and task service implementation request message size sending slot schedule message size (from the network slicing manager to user device), respectively. Θ^i_{pnd} and Θ^i_{wgd} are the total propagation and waiting delays, respectively. In this work, the M/D/1 queuing model is incorporated to calculate both queuing and waiting delays (Amreen et al., 2017). γ^i_{uwip} and γ^i_{uwip} are network slicing manager workloads and user device workloads for the network inauguration phase, respectively, where δ_{wl}, δ_{fl}, z_{fh}, and z_{wh} are the wireless link speed, fiber link speed, hop distance per fiber link transfer, and hop distance per wireless link transfer, respectively. Ω_{cp} and Ω_{lp} are the work processing speeds for the virtual worker/cloud server and user device, respectively. The resource information gathering phase delay (Δ^i_{urg}) includes different delays, such as user SFC-based application request reception delay, resource update request delay, resource update response delay, and inter-SDN controller information exchange delay. Δ^i_{urg} is investigated by using Eq. 2 as follows:

$$\begin{aligned}\Delta^i_{urg} = {} & \frac{\Gamma^i_{utr}+\Gamma^i_{rurs}+\Gamma^i_{uris}+\Gamma^i_{isdr}+\Gamma^i_{isres}}{\delta_{wl}} * z_{wh} \\ & + \frac{\Gamma^i_{utr}+\Gamma^i_{rurs}+\Gamma^i_{uris}+\Gamma^i_{isdr}+\Gamma^i_{isres}}{\delta_{fl}} * z_{fh} \\ & + \frac{\gamma^{nsm}_{urg}}{\Omega_{cp}} + \frac{\gamma^{wd}_{urg}}{\Omega_{lp}} + \Theta^i_{pnd} + \Theta^i_{wgd}.\end{aligned} \tag{2}$$

Γ^i_{isdr} and Γ^i_{isres} are inter-SDN controller tasks and resource information requests and response messages, respectively. Γ^i_{utr}, Γ^i_{rurs}, and Γ^i_{uris} are the

message size regarding users' task service request message, resource update request message, updated resource information reply message, respectively. γ^{nsm}_{urg} and γ^{wd}_{urg} are network slicing manager workloads and worker device workloads for the resource information gathering phases, respectively. The network slicing phase delay (Δ^{i}_{nsd}) includes slice allocation computing delay, schedule transfer delay to users, and schedule transfer delay to the worker nodes. The network slicing phase delay Δ^{i}_{nsd} is estimated by using Eq. 3 as follows:

$$\Delta^{i}_{nsd} = \frac{\Gamma^{i}_{nsd} + \Gamma^{i}_{rsd}}{\delta_{wl}} * z_{wh} + \frac{\Gamma^{i}_{nsd} + \Gamma^{i}_{rsd}}{\delta_{fl}} * z_{fh} + \frac{\gamma^{j}_{nsd}}{\Omega_{cp}} + \Theta^{i}_{pnd} + \Theta^{i}_{wgd}, \tag{3}$$

where Γ^{i}_{nsd} is the broadcaster network slice, resource, and worker allocation message size for the users. Γ^{i}_{rsd} is the broadcaster network slice, resource, and worker allocation message size for the worker nodes. γ^{i}_{nsd} is the workload for the network slicing manager regarding SFC ordering, priority checking, scheduling slot for each task, and best physical/virtual work node allocation.

Next, the total task work realization delay Δ^{i}_{twrd} is appraised by using Eq. 4 as follows:

$$\Delta^{i}_{twrd} = \sum_{i=1}^{y}\Big(\Theta^{i}_{ma} + \Theta^{i}_{ht} + \Theta^{i}_{ec} + \Theta^{i}_{bc} + \Theta^{i}_{hf} + \Theta^{i}_{ia} + \Theta^{i}_{hr} + \Theta^{i}_{ad} + \Theta^{i}_{eh} + \Theta^{i}_{vs} + \Theta^{i}_{xa} + \Theta^{i}_{sa} + \Theta^{i}_{su} + \Theta^{i}_{pnd} + \Theta^{i}_{wgd}\Big), \tag{4}$$

where $\Theta^{i}_{ma}, \Theta^{i}_{ht}, \Theta^{i}_{ec}, \Theta^{i}_{bc}, \Theta^{i}_{hf}, \Theta^{i}_{ia}, \Theta^{i}_{hr}, \Theta^{i}_{ad}, \Theta^{i}_{eh}, \Theta^{i}_{vs}, \Theta^{i}_{xa}, \Theta^{i}_{sa}$, and Θ^{i}_{su} are task work realization delay for metaverse tasks, holographic telepresence tasks, EV charging, brain–computer interaction-based tasks, haptic feedback, industrial automation, high-speed railway, FL-based automated driving, blockchain and digital twin–based electronic healthcare, video streaming, XR-based applications, smart agriculture, and video surveillance tasks, respectively. Θ^{i}_{pnd} and Θ^{i}_{wgd} are the total propagation and waiting (for resource) delays, respectively.

Our first 6G task is metaverse-based social avatar creation and avatar interaction (e.g., 6G FeMBB use case). The SFC delay (task realization delay) associated with social avatar–based metaverse task Θ^{i}_{ma} execution is given by using Eq. 5.

$$\Theta^{i}_{ma} = \sum_{i=1}^{y}\Big(\Theta^{i}_{urs} + \Theta^{i}_{vnf} + \Theta^{i}_{tis} + \Theta^{i}_{ucd} + \Theta^{i}_{ofd} + \Theta^{i}_{svnf} + \Theta^{i}_{cam} + \Theta^{i}_{am} + \Theta^{i}_{usc} + \Theta^{i}_{stm} + \Theta^{i}_{tvnf} + \Theta^{i}_{pma} + \Theta^{i}_{susc} + \Theta^{i}_{sstm} + \Theta^{i}_{fvnf} + \Theta^{i}_{psma}\Big), \tag{5}$$

where Θ^{i}_{urs} is a task request convey delay to slicing manager for metaverse-based social avatar creation and avatar interaction $\left(\Theta^{i}_{urs} = \frac{\Gamma^{i}_{tm}}{\delta_{wl}} * z_{wh} + \frac{\Gamma^{i}_{tm}}{\delta_{fl}} * z_{fh} + \Theta^{i}_{pnd} + \Theta^{i}_{wgd}\right)$.

Γ^{i}_{tm}, δ_{wl}, δ_{fl}, z_{fh}, and z_{wh} are the user's metaverse task request size, wireless link data transfer speed, fiber link data transfer speed, hop distance per fiber link–based transfer, and hop distance per wireless link–based transfer, respectively.

Θ^{i}_{vnf} is the initial VNF processing delay that includes FW, DPI, and NAT delays for metaverse-based social avatar creation and avatar interaction tasks. $\left(\Theta^{i}_{vnf} = \frac{\gamma^{i}_{fw}+\gamma^{i}_{dpi}+\gamma^{i}_{nat}}{\Omega_{cp}} + \frac{\Gamma^{i}_{tds}}{\delta_{tl}} * h_{tl}\right)$, where $\gamma^{i}_{fw}, \gamma^{i}_{dpi}, \gamma^{i}_{nat}$ are the workloads for FW, DPI, and NAT operation, respectively. Ω^{cp} is the virtual server processing speed. Γ^{i}_{tds}, δ_{tl}, and h^{tl} are the transferred data size from server to server during VNF processing, link rate, and hop distance, respectively.

Θ^{i}_{tis} is task instruction and virtual/physical work node selection-related information convey delay to the worker node from slicing manager device. $\left(\Theta^{i}_{tis} = \frac{\Gamma^{i}_{iid}}{\delta_{ud}} * z_{wh} + \frac{\Gamma^{i}_{iid}}{\delta_{fl}} * z_{fh} + \Theta^{i}_{pnd} + \Theta^{i}_{wgd}\right)$, where Γ^{i}_{iid} is the data size associated with task instruction and virtual/physical work node selection. Θ^{i}_{ucd} is the client-based avatar creation data capture delay that includes the users' image, pose, and position. $\left(\Theta^{i}_{ucd} = \frac{\gamma^{i}_{ldc}}{\Omega_{lp}}\right)$, where γ^{i}_{ldc} is the user workload for client device–based avatar creation data capture. Ω^{lp} is the client device's task work processing power. Θ^{i}_{ofd} is avatar creation data offload delay to a virtual worker at MEC server. $\Theta^{i}_{ofd} = \frac{\Gamma^{i}_{tid}}{\delta_{ul}} * z_{wh} + \frac{\Gamma^{i}_{tid}}{\delta_{fl}} * z_{fh} + \Theta^{i}_{pnd} + \Theta^{i}_{wgd}$, where Γ^{i}_{tid} is the offloaded avatar interaction task input data size.

Θ^{i}_{svnf} is the second VNF processing delay that includes IDS and NAT operation delays. $\left(\Theta^{i}_{svnf} = \frac{\gamma^{i}_{ids}+\gamma^{i}_{nat}}{\Omega_{cp}} + \frac{\gamma^{i}_{ids}+\gamma^{i}_{nat}}{\delta_{tl}} * h_{tl}\right)$, where γ^{i}_{ids} and γ^{i}_{nat} are the workloads for IDS and NAT operation processing, respectively. Ω^{cp} is the virtual MEC server processing speed. Θ^{i}_{cam} is avatar creation delay at the MEC for metaverse-based social avatar creation and avatar interaction tasks. $\left(\Theta^{i}_{cam} = \frac{\gamma^{i}_{ac}}{\Omega_{cp}}\right)$, where γ^{i}_{ac} is the workload for avatar creation by the MEC server.

Θ^{i}_{am} is the avatar movement delay at the metaverse. $\left(\Theta^{i}_{am} = \frac{\gamma^{i}_{am}}{\Omega_{cp}} + \frac{d^{i}_{am}}{\Omega_{ms}}\right)$, where γ^{i}_{am} is the MEC server workload for avatar movement in the metaverse. d^{i}_{am} is the distance from one avatar to another during movement. Ω_{ms} is the avatar movement speed.

Θ^{i}_{usc} is the first client device–based data capture delay for avatar interaction. $\left(\Theta^{i}_{usc} = \frac{\gamma^{i}_{usc}}{\Omega_{lp}}\right)$, where γ^{i}_{usc} is the user workload for conversation data capture for an avatar. Ω_{lp} is the client device's task work processing power. Θ^{i}_{stm} is the user conversation data offload delay from the user to metaverse avatar. $\Theta^{i}_{stm} = \frac{\Gamma^{i}_{acd}}{\delta_{wl}} * z_{wh} + \frac{\Gamma^{i}_{acd}}{\delta_{fl}} * z_{fh} + \Theta^{i}_{pnd} + \Theta^{i}_{wgd}$, where Γ^{i}_{acd} is the offloaded data size regarding avatar conversation.

Θ^{i}_{tvnf} is the third VNF processing delay that includes IDS, DPI, and NAT delays before offloaded task data are received at the MEC server for avatar creation and interaction tasks. $\left(\Theta^{i}_{tvnf} = \frac{\gamma^{i}_{ids}+\gamma^{i}_{dpi}+\gamma^{i}_{nat}}{\Omega_{cp}} + \frac{\gamma^{i}_{ids}+\gamma^{i}_{dpi}+\gamma^{i}_{nat}}{\delta_{tl}} * h_{tl}\right)$, where γ^{i}_{ids}, γ^{i}_{dpi}, and γ^{i}_{nat} are the workloads for IDS, DPI, and NAT operation, respectively.

Θ^{i}_{pma} is the virtual node–based avatar conversation message–playing operation. $\left(\Theta^{i}_{pma} = \frac{\gamma^{i}_{pma}}{\Omega_{cp}}\right)$, where γ^{i}_{pma} is the workload for virtual node–based avatar conversation message–playing work.

Θ^{i}_{susc} is the second user-based conversation data capture delay. $\left(\Theta^{i}_{susc} = \frac{\gamma^{i}_{susc}}{\Omega_{lp}}\right)$, where γ^{i}_{susc} is the second user workload.

Θ^{i}_{sstm} is the second user conversation data offload delay to the avatar. $\Theta^{i}_{sstm} = \frac{\Gamma^{i}_{sacd}}{\delta_{wl}} * z_{wh} + \frac{\Gamma^{i}_{sacd}}{\delta_{fl}} * z_{fh} + \Theta^{i}_{pnd} + \Theta^{i}_{wgd}$, where Γ^{i}_{sacd} is the offloaded data size regarding second user avatar conversation. Θ^{i}_{fvnf} is the fourth VNF processing delay that includes DPI, IDS, and NAT processing delays before offloaded task data are received at the MEC server. $\left(\Theta^{i}_{fvnf} = \frac{\gamma^{i}_{dpi}+\gamma^{i}_{ids}+\gamma^{i}_{nat}}{\Omega_{cp}} + \frac{\gamma^{i}_{dpi}+\gamma^{i}_{ids}+\gamma^{i}_{nat}}{\delta_{tl}} * h_{tl}\right)$, where γ^{i}_{dpi}, γ^{i}_{ids}, and γ^{i}_{nat} are the workloads for DPI, IDS, and NAT operation, respectively, at the virtual-node MEC server. Θ^{i}_{psma} is the virtual node–based second avatar message–playing operation.

$\left(\Theta^{i}_{psma} = \frac{\gamma^{i}_{psma}}{\Omega_{cp}}\right)$, where γ^{i}_{psma} is the workload for the second avatar conversation message–playing operation.

Next, this paper investigates the SFC delay (task realization delay) Θ^{i}_{ht} associated with holographic telepresence-related 6G applications using Eq. 6.

$$\Theta^{i}_{ht} = \sum_{i=1}^{y}\Big(\Theta^{i}_{urs} + \Theta^{ht}_{vnf} + \Theta^{i}_{tis} + \Theta^{ht}_{ucd} + \Theta^{ht}_{ofd} + \Theta^{ht}_{svnf} + \Theta^{ht}_{mp} + \Theta^{ht}_{str} + \Theta^{ht}_{tvnf} + \Theta^{i}_{rpr} + \Theta^{ht}_{vd}\Big), \tag{6}$$

where Θ^{i}_{urs} is a task request sending delay to the network slicing manager. Θ^{ht}_{vnf} is the initial VNF processing delay that includes FW, DPI, and NAT processing delays for holographic telepresence tasks. $\left(\Theta^{ht}_{vnf} = \frac{\gamma^{i}_{fw}+\gamma^{i}_{dpi}+\gamma^{i}_{nat}}{\Omega_{cp}} + \frac{\Gamma^{i}_{tds}}{\delta_{tl}} * h_{tl}\right)$, where $\gamma^{i}_{fw}, \gamma^{i}_{dpi}, \gamma^{i}_{nat}$ are the workloads for FW, DPI, and NAT operation processing, respectively. Θ^{i}_{tis} is task instruction and work node selection-related information convey delay to the worker node. Θ^{ht}_{ucd} is the user client device–based user data collection (e.g., user image, audio, pose, and eye position) delay for a holographic telepresence task. $\left(\Theta^{ht}_{ucd} = \frac{\gamma^{ht}_{ldc}}{\Omega_{lp}}\right)$, where γ^{ht}_{ldc} is the user workload for client device–based holographic task data capture work that includes users' image, pose, position, audio, and characteristics data input.

Θ^{ht}_{ofd} is a user-captured holographic task data offload delay to a virtual worker at MEC for a holographic telepresence task. $\Theta^{ht}_{ofd} = \frac{\Gamma^{ht}_{tid}}{\delta_{wl}} * z_{wh} + \frac{\Gamma^{i}_{tid}}{\delta_{fl}} * z_{fh} + \Theta^{i}_{pnd} + \Theta^{i}_{wgd}$, where Γ^{ht}_{tid} is the offloaded holographic telepresence task input data size. Θ^{ht}_{svnf} is the second VNF processing delay that includes IDS and NAT processing delays at the MEC server. $\left(\Theta^{ht}_{svnf} = \frac{\gamma^{ht}_{ids}+\gamma^{ht}_{nat}}{\Omega_{cp}} + \frac{\gamma^{ht}_{ids}+\gamma^{ht}_{nat}}{\delta_{tl}} * h_{tl}\right)$, where γ^{ht}_{ids} and γ^{ht}_{nat} are the workloads for IDS and NAT operation processing, respectively. Θ^{ht}_{mp} is the virtual work node–based holographic task data processing delay (3D construct, rendering, compress, and encoding) at the MEC server. $\left(\Theta^{ht}_{mp} = \frac{\gamma^{ht}_{mp}}{\Omega_{cp}}\right)$, where γ^{ht}_{mp} is the workload for holographic task data processing at the MEC server. Θ^{ht}_{str} is the MEC processing data transfer delay to the user device from the virtual server. $\Theta^{ht}_{str} = \frac{\Gamma^{ht}_{tod}}{\delta_{wl}} * z_{wh} + \frac{\Gamma^{ht}_{tod}}{\delta_{fl}} * z_{fh} + \Theta^{i}_{pnd} + \Theta^{i}_{wgd}$, where Γ^{ht}_{tod} is the MEC-processed holographic data size. Θ^{ht}_{tvnf} is the third VNF processing delay that includes IDS, FW, and NAT processing delay at user receiver device. $\left(\Theta^{ht}_{tvnf} = \frac{\gamma^{i}_{ids}+\gamma^{i}_{fw}+\gamma^{i}_{nat}}{\Omega_{lp}} + \frac{\gamma^{i}_{ids}+\gamma^{i}_{fw}+\gamma^{i}_{nat}}{\delta_{tl}} * h_{tl}\right)$, where γ^{i}_{ids}, γ^{i}_{fw}, and γ^{i}_{nat} are the workloads for IDS, FW, and NAT operation, respectively, at a receiver. Θ^{i}_{rpr} is the receiver device–based data processing delay that includes reprocessing, reconstruction, decompression, and decoding operations for holographic telepresence tasks. $\left(\Theta^{i}_{rpr} = \frac{\gamma^{ht}_{rp}}{\Omega_{lp}}\right)$, where γ^{ht}_{rp} is the workload for receiver device–based holographic data processing. Θ^{ht}_{vd} is the receiver device-based data visualization operation delay on a screen or projector with audio. $\left(\Theta^{ht}_{vd} = \frac{\gamma^{ht}_{vd}}{\Omega_{lp}}\right)$, where γ^{ht}_{vd} is the workload for receiver device-based holographic data visualization operations.

Next, this paper investigates the SFC delay (task realization delay) associated with EV charging related to 6G applications using Eq. 7.

$$\Theta^{i}_{ec} = \sum_{i=1}^{y}\Big(\Theta^{i}_{urs} + \Theta^{ec}_{vnf} + \Theta^{i}_{tis} + \Theta^{ec}_{ucd} + \Theta^{ec}_{ofd} + \Theta^{ec}_{svnf} + \Theta^{ec}_{mp} + \Theta^{ec}_{str} + \Theta^{ec}_{tvnf} + \Theta^{ec}_{um} + \Theta^{ec}_{up}\Big), \tag{7}$$

where Θ^{i}_{urs} is a task request sending delay to the network slicing manager from the user device. Θ^{ec}_{vnf} is the initial VNF processing delay that includes FW, DPI, and NAT operation for EV charging tasks. $\left(\Theta^{ec}_{vnf} = \frac{\gamma^{i}_{fw}+\gamma^{i}_{dpi}+\gamma^{i}_{nat}}{\Omega_{cp}} + \frac{\Gamma^{i}_{tds}}{\delta_{tl}} * h_{tl}\right)$, where $\gamma^{i}_{fw}, \gamma^{i}_{dpi}, \gamma^{i}_{nat}$ are the workloads for FW, DPI, and NAT operation processing, respectively. Θ^{i}_{tis} is task instruction and work node selection information convey delay to the worker node. $\Theta^{i}_{tis} = \frac{\Gamma^{i}_{iid}}{\delta_{wl}} * z_{wh} + \frac{\Gamma^{i}_{tid}}{\delta_{fl}} * z_{fh} + \Theta^{i}_{pnd} + \Theta^{i}_{wgd}$), where Γ^{i}_{iid} is the data size associated with task instruction and work node selection information.

Θ^{ec}_{ucd} is the user client device–based user data collection (e.g., user movement, vehicle charging requirement, location, endpoint, and starting point) delay for the EV charging task. $\left(\Theta^{ec}_{ucd} = \frac{\gamma^{ec}_{ldc}}{\Omega_{lp}}\right)$, where γ^{ec}_{ldc} is the user workload for the client device–based electronic vehicle task data capture work. Θ^{ec}_{ofd} is user-captured EV charging task data offload delay to a virtual worker at MEC for EV charging task. $\Theta^{ec}_{ofd} = \frac{\Gamma^{ec}_{tid}}{\delta_{wl}} * z_{wh} + \frac{\Gamma^{ec}_{tid}}{\delta_{fl}} * z_{fh} + \Theta^{i}_{pnd} + \Theta^{i}_{wgd}$, where Γ^{ec}_{tid} is the offload or EV charging task input data size. Θ^{ec}_{svnf} is the second VNF processing delay that includes IDS, FW, and NAT processing delays at the MEC server for EV charging tasks. $\left(\Theta^{ec}_{svnf} = \frac{\gamma^{i}_{ids}+\gamma^{i}_{nat}+\gamma^{i}_{fw}}{\Omega_{cp}} + \frac{\gamma^{i}_{ids}+\gamma^{i}_{nat}+\gamma^{i}_{fw}}{\delta_{tl}} * h_{tl}\right)$, where γ^{i}_{ids}, γ^{i}_{fw}, and γ^{i}_{nat} are the workloads for IDS, FW, and NAT operation.

Θ^{ec}_{mp} is the virtual work node–based EV charging task data processing delay that includes users' EV charging station selection delay at the MEC server. $\left(\Theta^{ec}_{mp} = \frac{\gamma^{ec}_{mp}}{\Omega_{cp}}\right)$, where γ^{ec}_{mp} is the workload for EV charging task data processing at the MEC server. Θ^{ec}_{str} is the MEC processing data transfer delay to the user device from the virtual server. $\Theta^{ec}_{str} = \frac{\Gamma^{ec}_{tod}}{\delta_{wl}} * z_{wh} + \frac{\Gamma^{ec}_{tod}}{\delta_{fl}} * z_{fh} + \Theta^{i}_{pnd} + \Theta^{i}_{wgd}$, where Γ^{ec}_{tod} is the MEC-processed or EV charging task data output size. Θ^{ec}_{tvnf} is the third VNF processing delay that includes IDS and NAT delay processing delay at the user receiver device. $\left(\Theta^{ec}_{tvnf} = \frac{\gamma^{i}_{ids}+\gamma^{i}_{nat}}{\Omega_{lp}} + \frac{\gamma^{i}_{ids}+\gamma^{i}_{nat}}{\delta_{tl}} * h_{tl}\right)$, where γ^{i}_{ids} and γ^{i}_{nat} are the workloads for IDS and NAT operation processing at the receiver. Θ^{ec}_{um} is the user electronic vehicle movement (from the starting to charging station) delay. $\left(\Theta^{ec}_{um} = \frac{d^{ev}_{sc}}{\Omega_{msu}}\right)$, where d^{ev}_{sc} is the distance from the starting point to charging station point. Ω_{msu} is the client device's movement speed. Θ^{ec}_{up} is the EV charging delay at the selected charging station. $\left(\Theta^{ec}_{up} = \frac{a_{rq}-a_{av}+a_{bdt}*b}{\Omega_{ech}}\right)$, where a_{rq}, a_{av}, a_{bdt}, b, and Ω_{ech} are charging requirements, available charge, battery depletion threshold, battery capacity, and EV charging rate, respectively.

Next, this paper investigates the SFC delay (task realization delay) Θ^{i}_{bc} associated with brain–computer interaction-based 6G applications using Eq. 8 (e.g., umMTC task).

$$\Theta^{i}_{bc} = \sum_{i=1}^{y}\Big(\Theta^{i}_{urs} + \Theta^{bc}_{vnf} + \Theta^{i}_{tis} + \Theta^{bc}_{ucd} + \Theta^{bc}_{ofd} + \Theta^{bc}_{svnf} + \Theta^{bc}_{mp} + \Theta^{bc}_{str} + \Theta^{bc}_{tvnf} + \Theta^{bc}_{rpr}\Big), \tag{8}$$

where Θ^{i}_{urs} is a task request sending delay to the network slicing manager from the user device. Θ^{bc}_{vnf} is the initial VNF processing delay that includes FW, DPI, and NAT delays for the brain–computer interaction task. $\left(\Theta^{bc}_{vnf} = \frac{\gamma^{i}_{fw}+\gamma^{i}_{dpi}+\gamma^{i}_{nat}}{\Omega_{cp}} + \frac{\Gamma^{i}_{tds}}{\delta_{tl}} * h_{tl}\right)$.

Θ^{i}_{tis} is task instruction and work node selection information sending delay to the worker node. $\left(\Theta^{i}_{tis} = \frac{\Gamma^{i}_{iid}}{\delta_{ul}} * z_{wh} + \frac{\Gamma^{i}_{tid}}{\delta_{fl}} * z_{fh} + \Theta^{i}_{pnd} + \Theta^{i}_{wgd}\right)$. Θ^{bc}_{ucd} is the user head sensor device-based user data collection (e.g., EEG, fMRI, and MEG) delay for the brain–computer interaction task. $\left(\Theta^{bc}_{ucd} = \frac{\gamma^{bc}_{ldc}}{\Omega_{lp}}\right)$, where γ^{bc}_{ldc} is the user workload for brain data capture work. Ω_{lp} is the client device/brain sensors' task work processing power.

Θ^{bc}_{ofd} is user-captured brain–computer interaction task data offload delay to a virtual worker at MEC server. $\Theta^{bc}_{ofd} = \frac{\Gamma^{bc}_{tid}}{\delta_{ul}} * z_{wh} + \frac{\Gamma^{bc}_{tid}}{\delta_{fl}} * z_{fh} + \Theta^{i}_{pnd} + \Theta^{i}_{wgd}$, where Γ^{bc}_{tid} is the offloaded data size for the brain–computer interaction task. Θ^{bc}_{svnf} is the second VNF processing delay that includes IDS and NAT processing delays at the MEC server. $\left(\Theta^{bc}_{svnf} = \frac{\gamma^{i}_{ids}+\gamma^{i}_{nat}}{\Omega_{cp}} + \frac{\gamma^{i}_{ids}+\gamma^{i}_{nat}}{\delta_{tl}} * h_{tl}\right)$, where γ^{i}_{ids} and γ^{i}_{nat} are the workloads for IDS and NAT operation processing; Θ^{bc}_{mp} is the virtual work node–based brain–computer interaction task data processing delay that includes brain signal acquisition from collected data, feature extraction, pattern recognition, and translation command delay at the MEC server. $\left(\Theta^{bc}_{mp} = \frac{\gamma^{bc}_{mp}}{\Omega_{cp}}\right)$, where γ^{bc}_{mp} is the workload for brain–computer interaction task data processing at the MEC server. Θ^{bc}_{str} is the MEC processing brain–computer interaction task result or command data transfer delay to the user device (wheelchair). $\Theta^{bc}_{str} = \frac{\Gamma^{bc}_{tod}}{\delta_{ul}} * z_{wh} + \frac{\Gamma^{bc}_{tod}}{\delta_{fl}} * z_{fh} + \Theta^{i}_{pnd} + \Theta^{i}_{wgd}$, where Γ^{bc}_{tod} is the MEC-processed brain–computer interaction task result or command data size. Θ^{bc}_{tvnf} is the third VNF processing delay that includes IDS, FW, and NAT delay processing delay at the user receiver device. $\left(\Theta^{bc}_{tvnf} = \frac{\gamma^{i}_{ids}+\gamma^{i}_{fw}+\gamma^{i}_{nat}}{\Omega_{lp}} + \frac{\gamma^{i}_{ids}+\gamma^{i}_{fw}+\gamma^{i}_{nat}}{\delta_{tl}} * h_{tl}\right)$, where γ^{i}_{ids}, γ^{i}_{fw}, and γ^{i}_{nat} are the workloads for IDS, FW, and NAT operation processing at the receiver wheelchair device, respectively. Θ^{bc}_{rpr} is the user/receiver wheelchair device–based data processing delay that includes wheelchair movement operation based on MEC-processed commands from brain signals. $\left(\Theta^{bc}_{rpr} = \frac{\gamma^{bc}_{rp}}{\Omega_{lp}}\right)$, where γ^{bc}_{rp} is the workload for receiver device–based wheelchair movement operation based on the transferred command.

After that, this paper investigates the SFC delay (task realization delay) Θ^{i}_{hf} associated with haptic feedback-based immersive gaming 6G applications using Eq. 9 (e.g., FeMBB).

$$\Theta^{i}_{hf} = \sum_{i=1}^{y}\Big(\Theta^{i}_{urs} + \Theta^{hf}_{vnf} + \Theta^{i}_{tis} + \Theta^{hf}_{ueg} + \Theta^{hf}_{ucd} + \Theta^{hf}_{ofd} + \Theta^{hf}_{svnf} + \Theta^{hf}_{mp} + \Theta^{hf}_{str} + \Theta^{hf}_{tvnf} + \Theta^{hf}_{rpr}\Big), \tag{9}$$

where Θ^{i}_{urs} is a task request sending delay to the network slicing manager from the user device. Θ^{hf}_{vnf} is the initial VNF processing delay that includes FW, DPI, and NAT processing delays. $\left(\Theta^{bc}_{vnf} = \frac{\gamma^{i}_{fw}+\gamma^{i}_{dpi}+\gamma^{i}_{nat}}{\Omega_{cp}} + \frac{\Gamma^{i}_{tds}}{\delta_{tl}} * h_{tl}\right)$. Θ^{i}_{tis} is task instruction and work node selection-related information sending delay to the worker node from network slicing manager device for haptic feedback-based immersive gaming task $\left(\Theta^{i}_{tis} = \frac{\Gamma^{i}_{iid}}{\delta_{ul}} * z_{wh} + \frac{\Gamma^{i}_{tid}}{\delta_{fl}} * z_{fh} + \Theta^{i}_{pnd} + \Theta^{i}_{wgd}\right)$. Θ^{hf}_{ueg} is the time required for the user device entering the game and connectivity with the MEC $\left(\Theta^{hf}_{ueg} = \frac{\gamma^{hf}_{ueg}}{\Omega_{lp}}\right)$, where γ^{hf}_{ueg} is the user workload for entering the game state. Θ^{hf}_{ucd} is the user haptic input collection delay from hand gloves or haptic devices or game state input information collection. $\left(\Theta^{hf}_{ucd} = \frac{\gamma^{hf}_{ldc}}{\Omega_{lp}}\right)$, where γ^{hf}_{ldc} is the user workload for haptic feedback task input data collection (e.g., touch a ball). Θ^{hf}_{ofd} is the user-captured data offload delay to the virtual worker at MEC server. $\Theta^{hf}_{ofd} = \frac{\Gamma^{hf}_{tid}}{\delta_{ul}} * z_{wh} + \frac{\Gamma^{hf}_{tid}}{\delta_{fl}} * z_{fh} + \Theta^{i}_{pnd} + \Theta^{i}_{wgd}$, where Γ^{hf}_{tid} is the offloaded data size. Θ^{hf}_{svnf} is the second VNF processing delay that includes IDS and NAT processing delays at the MEC server. $\left(\Theta^{hf}_{svnf} = \frac{\gamma^{i}_{ids}+\gamma^{i}_{nat}}{\Omega_{cp}} + \frac{\gamma^{hf}_{ids}+\gamma^{hf}_{nat}}{\delta_{tl}} * h_{tl}\right)$, where γ^{i}_{ids} and γ^{i}_{nat} are the workloads for IDS and NAT operation, respectively. Θ^{hf}_{mp} is the virtual work node–based haptic input task data processing delay that includes rendering, game data processing, and audio/visual haptic feedback data generation delay at the MEC server. $\left(\Theta^{hf}_{mp} = \frac{\gamma^{hf}_{mp}}{\Omega_{cp}}\right)$, where γ^{hf}_{mp} is the workload for haptic input task data processing at the MEC server. Θ^{hf}_{str} is MEC-processed haptic feedback task result data transfer delay to the user device (haptic gloves or jacket) from the virtual MEC server. $\Theta^{hf}_{str} = \frac{\Gamma^{hf}_{tod}}{\delta_{ul}} * z_{wh} + \frac{\Gamma^{hf}_{tod}}{\delta_{fl}} * z_{fh} + \Theta^{i}_{pnd} + \Theta^{i}_{wgd}$, where Γ^{hf}_{tod} is the MEC-processed haptic feedback task result or data size. Θ^{hf}_{tvnf} is the third VNF processing delay that includes IDS and NAT processing delays at the user receiver device. $\left(\Theta^{hf}_{tvnf} = \frac{\gamma^{i}_{ids}+\gamma^{i}_{nat}}{\Omega_{lp}} + \frac{\gamma^{i}_{ids}+\gamma^{i}_{nat}}{\delta_{tl}} * h_{tl}\right)$, where γ^{i}_{ids} and γ^{i}_{nat} are the workloads for IDS and NAT operation at the receiver haptic device, respectively. Θ^{hf}_{rpr} is the user/receiver haptic device-based data processing delay that includes feeling processed haptic sensory feedback data or sensation via the haptic devices such as haptic jackets, gloves, and eyeglasses. $\left(\Theta^{hf}_{rpr} = \frac{\gamma^{hf}_{rp}}{\Omega_{lp}}\right)$, where γ^{hf}_{rp} is the workload for receiver device–based haptic feedback reception.

Next, this paper investigates the SFC delay (task realization delay) Θ^{i}_{ia} associated with human–robot processing-based industrial automation non-6G applications using Eq. 10 (e.g., URLLC).

$$\Theta^{i}_{ia} = \sum_{i=1}^{y}\Big(\Theta^{i}_{urs} + \Theta^{ia}_{vnf} + \Theta^{i}_{tis} + \Theta^{ia}_{sm} + \Theta^{ia}_{rm} + \Theta^{ia}_{ofd} + \Theta^{ia}_{svnf} + \Theta^{ia}_{hp} + \Theta^{ia}_{str} + \Theta^{ia}_{rpr} + \Theta^{ia}_{sofd} + \Theta^{ia}_{hfp}\Big). \tag{10}$$

Θ^{i}_{urs} is a task request sending delay to the network slicing manager from the user device. Θ^{ia}_{vnf} is the initial VNF processing delay that includes FW, DPI, and NAT operation delays. $\left(\Theta^{ia}_{vnf} = \frac{\gamma^{i}_{fw}+\gamma^{i}_{dpi}+\gamma^{i}_{nat}}{\Omega_{cp}} + \frac{\Gamma^{i}_{tds}}{\delta_{tl}} * h_{tl}\right)$. Θ^{i}_{tis} is task instruction and work node selection information convey delay to the worker node. $\Theta^{i}_{tis} = \frac{\Gamma^{i}_{iid}}{\delta_{ul}} * z_{wh} + \frac{\Gamma^{i}_{tid}}{\delta_{fl}} * z_{fh} + \Theta^{i}_{pnd} + \Theta^{i}_{wgd}$. Θ^{ia}_{sm} is the time required for supplying raw material to the robot and design supply for production $\left(\Theta^{ia}_{sm} = \frac{\gamma^{ia}_{sm}}{\Omega_{lp}} + \frac{\Gamma^{i}_{tds}}{\delta_{tl}} * h_{tl}\right)$, where γ^{ia}_{sm} is the user workload

for supplying raw material and design to the robot. Θ^{ia}_{rm} is the robot-based manufacturing product operation. $\left(\Theta^{ia}_{rm} = \frac{\gamma^{ia}_{rm}}{\Omega_{rlp}}\right)$, where γ^{ia}_{rm} is the user workload for robot-based manufacturing product operation. Ω_{rlp} is the robot's processing speed. Θ^{ia}_{ofd} is robot-processed data offload delay to the human worker. $\Theta^{ia}_{ofd} = \frac{\Gamma^{ia}_{tid}}{\delta_{wl}} * z_{wh} + \frac{\Gamma^{ia}_{tid}}{\delta_{fl}} * z_{fh} + \Theta^{i}_{pnd} + \Theta^{i}_{wgd}$, where Γ^{ia}_{tid} is the offloaded data size. Θ^{ia}_{svnf} is the second VNF processing delay that includes IDS and NAT processing at the MEC server. $\left(\Theta^{ia}_{svnf} = \frac{\gamma^{i}_{ids}+\gamma^{i}_{nat}}{\Omega_{cp}} + \frac{\gamma^{i}_{ids}+\gamma^{i}_{nat}}{\delta_{tl}} * h_{tl}\right)$, where γ^{i}_{ids} and γ^{i}_{nat} are the workloads for IDS and NAT operation at the human device, respectively. Θ^{ia}_{hp} is the human work node–based task data processing delay that includes checking robots' work and giving advice. $\left(\Theta^{ia}_{hp} = \frac{\gamma^{ia}_{hp}}{\Omega_{lp}}\right)$, where γ^{ia}_{hp} is the workload for checking robots' work and giving advice by the human device. Θ^{ia}_{str} is the human device-processed task result data transfer delay to the robot device. $\Theta^{ia}_{str} = \frac{\Gamma^{ia}_{tod}}{\delta_{wl}} * z_{wh} + \frac{\Gamma^{ia}_{tod}}{\delta_{fl}} * z_{fh} + \Theta^{i}_{pnd} + \Theta^{i}_{wgd}$, where Γ^{ia}_{tod} is the human user processed task result. Θ^{ia}_{rpr} is the robot-based rechecking and re-manufacturing of products based on humans' suggestions. $\left(\Theta^{ia}_{rpr} = \frac{\gamma^{ia}_{rpr}}{\Omega_{rlp}}\right)$, where γ^{ia}_{rpr} is the workload for robot-based rechecking and re-manufacturing product operations. Θ^{ia}_{sofd} is robots' reprocessing production data offload delay to a human worker. $\Theta^{ia}_{sofd} = \frac{\Gamma^{ia}_{stid}}{\delta_{wl}} * z_{wh} + \frac{\Gamma^{ia}_{stid}}{\delta_{fl}} * z_{fh} + \Theta^{i}_{pnd} + \Theta^{i}_{wgd}$, where Γ^{ia}_{stid} is the second offloaded data size by the robot's second inspection of the human user's device. Θ^{ia}_{hfp} is the human work node–based robot task data processing delay that includes checking robots reprocessing work and confirming work. $\left(\Theta^{ia}_{hfp} = \frac{\gamma^{ia}_{hfp}}{\Omega_{lp}}\right)$, where γ^{ia}_{hfp} is the workload for checking or confirming robots' final processed work by the human device.

Next, this paper investigates the SFC delay (task realization delay) Θ^{i}_{hr} associated with high-speed railway-based user data transfer applications using Eq. 11 (e.g., LDHMC).

$$\Theta^{i}_{hr} = \sum_{i=1}^{y}\Big(\Theta^{i}_{urs} + \Theta^{hr}_{vnf} + \Theta^{i}_{tis} + \Theta^{hr}_{svnf} + \Theta^{hr}_{mp} + \Theta^{hr}_{str} + \Theta^{hr}_{uds} + \Theta^{hr}_{tvnf} + \Theta^{hr}_{ofd} + \Theta^{hr}_{fvnf} + \Theta^{hr}_{udr}\Big), \tag{11}$$

where Θ^{i}_{urs} is a task request sending delay to the network slicing manager from the user device. Θ^{hr}_{vnf} is the initial VNF processing delay that includes FW, DPI, and NAT operation. $\left(\Theta^{hr}_{vnf} = \frac{\gamma^{i}_{fw}+\gamma^{i}_{dpi}+\gamma^{i}_{nat}}{\Omega_{cp}} + \frac{\Gamma^{i}_{tds}}{\delta_{tl}} * h_{tl}\right)$. Θ^{i}_{tis} is task instruction and work node selection-related information sending delay to the worker node $\left(\Theta^{i}_{tis} = \frac{\Gamma^{i}_{iid}}{\delta_{wl}} * z_{wh} + \frac{\Gamma^{i}_{tid}}{\delta_{fl}} * z_{fh} + \Theta^{i}_{pnd} + \Theta^{i}_{wgd}\right)$. Θ^{hr}_{svnf} is the second VNF processing delay that includes IDS and NAT processing delays at the MEC server. $\left(\Theta^{hr}_{svnf} = \frac{\gamma^{i}_{ids}+\gamma^{i}_{nat}}{\Omega_{cp}} + \frac{\gamma^{i}_{ids}+\gamma^{i}_{nat}}{\delta_{tl}} * h_{tl}\right)$, where γ^{i}_{ids} and γ^{i}_{nat} are the workloads for IDS and NAT operation, respectively. Θ^{hr}_{mp} is the virtual work node–based task data processing delay that includes selecting a suitable base station and time slot for data transfer for the user. $\left(\Theta^{hr}_{mp} = \frac{\gamma^{hr}_{mp}}{\Omega_{cp}}\right)$, where γ^{hr}_{mp} is the workload for BS selection at the MEC server.

Θ^{hr}_{str} is MEC-processed BS selection data transfer delay to the user device from virtual MEC server. $\Theta^{hr}_{str} = \frac{\Gamma^{hr}_{tod}}{\delta_{wl}} * z_{wh} + \frac{\Gamma^{hr}_{tod}}{\delta_{fl}} * z_{fh} + \Theta^{i}_{pnd} + \Theta^{i}_{wgd}$, where Γ^{hr}_{tod} is the MEC-processed task result data size for high-speed railway users. Θ^{hr}_{uds} is the user data offload delay to the receiver base station. $\Theta^{hr}_{uds} = \frac{\Gamma^{hr}_{uds}}{\delta_{wl}} * z_{wh} + \frac{\Gamma^{hr}_{uds}}{\delta_{fl}} * z_{fh} + \Theta^{i}_{pnd} + \Theta^{i}_{wgd}$, where Γ^{hr}_{uds} is the offloaded data size. Θ^{hr}_{tvnf} is the third VNF processing delay that includes FW, LB, and encryption processing delay at the user receiver base station device. $\left(\Theta^{hr}_{tvnf} = \frac{\gamma^{i}_{fw}+\gamma^{i}_{lb}+\gamma^{i}_{en}}{\Omega_{lp}} + \frac{\gamma^{i}_{fw}+\gamma^{i}_{lb}+\gamma^{i}_{en}}{\delta_{tl}} * h_{tl}\right)$, where γ^{i}_{lb} and γ^{i}_{en} are the workloads for LB and encryption operation processing at the receiver base station device.

Θ^{hr}_{ofd} is the user data transfer delay from the receiver base station to the receiver device. $\Theta^{hr}_{ofd} = \frac{\Gamma^{hr}_{ofd}}{\delta_{wl}} * z_{wh} + \frac{\Gamma^{hr}_{ofd}}{\delta_{fl}} * z_{fh} + \Theta^{i}_{pnd} + \Theta^{i}_{wgd}$, where Γ^{hr}_{ofd} is the user's transferred data size to the receiver. Θ^{hr}_{fvnf} is the fourth VNF processing delay that includes IDS and decryption processing delay at the receiver device. $\left(\Theta^{hr}_{fvnf} = \frac{\gamma^{i}_{ids}+\gamma^{i}_{de}}{\Omega_{lp}} + \frac{\gamma^{i}_{ids}+\gamma^{i}_{de}}{\delta_{tl}} * h_{tl}\right)$, where γ^{i}_{ids} and γ^{i}_{de} are the workloads for IDS and decryption operation at the receiver, respectively. Θ^{hr}_{udr} is the receiver device–based data display operation delay for high-speed railway-based data transfer tasks. $\left(\Theta^{hr}_{udr} = \frac{\gamma^{hr}_{ldc}}{\Omega_{lp}}\right)$, where γ^{hr}_{ldc} is the workload for output data display at the receiver device.

Next, this paper investigates the SFC delay (task realization delay) Θ^{i}_{ad} for FL-based autonomous driving 6G applications using Eq. 12 (e.g., eURLLC).

$$\Theta^{i}_{ad} = \sum_{i=1}^{y}\Big(\Theta^{i}_{urs} + \Theta^{ad}_{vnf} + \Theta^{i}_{tis} + \Theta^{ad}_{svnf} + \Theta^{ad}_{gd} + \Theta^{ad}_{ucd} + \Theta^{ad}_{ult} + \Theta^{ad}_{ofd} + \Theta^{ad}_{tvnf} + \Theta^{ad}_{mp} + \Theta^{ad}_{fvnf} + \Theta^{ad}_{fud}\Big) \tag{12}$$

Θ^{i}_{urs} is a task request sending delay to the network slicing manager from the user device. Θ^{ad}_{vnf} is the initial VNF processing delay that includes FW, DPI, and NAT operation. $\left(\Theta^{ad}_{vnf} = \frac{\gamma^{i}_{fw}+\gamma^{i}_{dpi}+\gamma^{i}_{nat}}{\Omega_{cp}} + \frac{\Gamma^{i}_{tds}}{\delta_{tl}} * h_{tl}\right)$, where $\gamma^{i}_{fw}, \gamma^{i}_{dpi}, \gamma^{i}_{nat}$ are the workloads for FW, DPI, and NAT operation processing. Θ^{i}_{tis} is task instruction and work node selection information convey delay to the worker node. $\Theta^{i}_{tis} = \frac{\Gamma^{i}_{iid}}{\delta_{wl}} * z_{wh} + \frac{\Gamma^{i}_{tid}}{\delta_{fl}} * z_{fh} + \Theta^{i}_{pnd} + \Theta^{i}_{wgd}$, where Γ^{i}_{iid} is the data size associated with task instruction and work node selection. Θ^{ad}_{svnf} is the second VNF processing delay at the cloud server that includes FW, DPI, and IDS delay. $\left(\Theta^{ad}_{svnf} = \frac{\gamma^{i}_{fw}+\gamma^{i}_{dpi}+\gamma^{i}_{ids}}{\Omega_{cp}} + \frac{\Gamma^{i}_{tds}}{\delta_{tl}} * h_{tl}\right)$, where $\gamma^{i}_{fw}, \gamma^{i}_{dpi}, \gamma^{i}_{ids}$ are the workloads for FW, DPI, and IDS operation processing.

Θ^{ad}_{gd} is a global deep learning model download delay from the global edge server. $\Theta^{ad}_{gd} = \frac{\Gamma^{i}_{gd}}{\delta_{wl}} * z_{wh} + \frac{\Gamma^{i}_{gd}}{\delta_{fl}} * z_{fh} + \Theta^{i}_{pnd} + \Theta^{i}_{wgd}$), where Γ^{i}_{gd} is the data size associated with the global deep learning model (e.g., image processing or object detection model). Θ^{ad}_{ucd} is the user client device–based user data collection (e.g., roadside image) delay. $\left(\Theta^{ad}_{ucd} = \frac{\gamma^{ad}_{ldc}}{\Omega_{vp}}\right)$, where γ^{ad}_{ldc} is the workload for client device–based roadside data capture work. Ω_{vp} is the client device's task-work processing power. Θ^{ad}_{ult} is the user client device (vehicle)–based local data training delay. $\left(\Theta^{ad}_{ult} = \frac{\gamma^{ad}_{ldt}}{\Omega_{vp}}\right)$, where γ^{ad}_{ldt} is the workload for client device–based local model data training delay and updated local parameter generation delay. Θ^{ad}_{ofd} is user-generated updated model

parameters for task data offload delay to a virtual worker at MEC. $\Theta^{ad}_{ofd} = \frac{\Gamma^{ad}_{tid}}{\delta_{ul}} * z_{wh} + \frac{\Gamma^{ad}_{tid}}{\delta_{fl}} * z_{fh} + \Theta^{i}_{pnd} + \Theta^{i}_{wgd}$, where Γ^{ad}_{tid} is the offload data size or updated local model task input data size. Θ^{ad}_{tvnf} is the third VNF processing delay that includes IDS and NAT processing delays at the MEC server. $\left(\Theta^{ad}_{tvnf} = \frac{\gamma^{i}_{ids}+\gamma^{i}_{nat}}{\Omega_{cp}} + \frac{\gamma^{i}_{ids}+\gamma^{i}_{nat}}{\delta_{tl}} * h_{tl}\right)$, where γ^{i}_{ids} and γ^{i}_{nat} are the workloads for IDS and NAT operation processing, respectively. Θ^{ad}_{mp} is the virtual work node–based global deep learning model update delay by locally updating data aggregation at the MEC server. $\left(\Theta^{ad}_{mp} = \frac{\gamma^{ad}_{mp}}{\Omega_{cp}}\right)$, where γ^{ad}_{mp} is the workload for global model update and local data aggregation at the MEC server. Θ^{ad}_{fvnf} is the fourth VNF processing delay that includes IDS and NAT processing delays at the user receiver device. $\left(\Theta^{ad}_{fvnf} = \frac{\gamma^{i}_{ids}+\gamma^{i}_{nat}}{\Omega_{lp}} + \frac{\gamma^{i}_{ids}+\gamma^{i}_{nat}}{\delta_{tl}} * h_{tl}\right)$, where γ^{i}_{ids} and γ^{i}_{nat} are the workloads for IDS and NAT operation at a receiver. Θ^{ad}_{fud} is an updated global deep learning model download delay at the receiver. $\Theta^{ad}_{fud} = \frac{\Gamma^{ad}_{tod}}{\delta_{ul}} * z_{wh} + \frac{\Gamma^{ad}_{tod}}{\delta_{fl}} * z_{fh} + \Theta^{i}_{pnd} + \Theta^{i}_{wgd}$, where Γ^{ad}_{tod} is the updated global deep learning model data size.

Next, the article investigates the SFC delay (task realization delay) Θ^{i}_{eh} associated with blockchain and digital twin–based e-healthcare 6G applications using Eq. 13 (e.g., ELPC).

$$\begin{aligned}\Theta^{i}_{eh} = \sum_{i=1}^{y}\Big(&\Theta^{i}_{urs} + \Theta^{eh}_{vnf} + \Theta^{i}_{tis} + \Theta^{eh}_{bco} + \Theta^{eh}_{ucd} + \Theta^{eh}_{ofd} \\ &+\Theta^{eh}_{svnf} + \Theta^{eh}_{mp} + \Theta^{eh}_{std} + \Theta^{eh}_{dt} + \Theta^{eh}_{stm} + \Theta^{eh}_{bc} \\ &+\Theta^{eh}_{stv} + \Theta^{eh}_{bv} + \Theta^{eh}_{ulb} + \Theta^{eh}_{stu} + \Theta^{eh}_{tvnf} + \Theta^{eh}_{rdd}\Big).\end{aligned} \tag{13}$$

Θ^{i}_{urs} is a task request sending delay to the network slicing manager from the user. Θ^{eh}_{vnf} is the initial VNF processing delay that includes FW, DPI, and NAT operation. $\left(\Theta^{eh}_{vnf} = \frac{\gamma^{i}_{fw}+\gamma^{i}_{dpi}+\gamma^{i}_{nat}}{\Omega_{cp}} + \frac{\Gamma^{i}_{tds}}{\delta_{tl}} * h_{tl}\right)$.

Θ^{i}_{tis} is task instruction and work node selection information sending delay to the worker node. $\Theta^{i}_{tis} = \frac{\Gamma^{i}_{iid}}{\delta_{ul}} * z_{wh} + \frac{\Gamma^{i}_{iid}}{\delta_{fl}} * z_{fh} + \Theta^{i}_{pnd} + \Theta^{i}_{wgd}$. Θ^{eh}_{bco} is the delay associated with initial key exchange, smart contract, and blockchain operation registration. $\Theta^{eh}_{bco} = frac\Gamma^{i}_{bco}\delta_{ul} * z_{wh} + \frac{\Gamma^{i}_{bco}}{\delta_{fl}} * z_{fh} + \frac{\gamma^{cp}_{bci}}{\Omega_{cp}} + \frac{\gamma^{ld}_{bci}}{\Omega_{lp}} + \Theta^{i}_{pnd} + \Theta^{i}_{wgd}$, where Γ^{i}_{bco} is the exchanged data size for initial key exchange, smart contract, and blockchain registration operations. γ^{cp}_{bci} and γ^{ld}_{bci} are the MEC blockchain server workload and local client workload for initial blockchain work, respectively. Θ^{eh}_{ucd} is the IoT device or sensor data collection regarding human health (blood pressure and temperature) status. $\left(\Theta^{eh}_{ucd} = \frac{\gamma^{eh}_{ldc}}{\Omega_{lp}}\right)$, where γ^{eh}_{ldc} is the user workload for e-healthcare data collection. Θ^{eh}_{ofd} is the user-captured data offload delay to the virtual worker at MEC. $\Theta^{eh}_{ofd} = \frac{\Gamma^{eh}_{tid}}{\delta_{ul}} * z_{wh} + \frac{\Gamma^{eh}_{tid}}{\delta_{fl}} * z_{fh} + \Theta^{i}_{pnd} + \Theta^{i}_{wgd}$, where Γ^{eh}_{tid} is the offloaded data/input task data size.

Θ^{eh}_{svnf} is the second VNF processing delay that includes IDS and NAT processing delays at the MEC server. $\left(\Theta^{eh}_{svnf} = \frac{\gamma^{i}_{ids}+\gamma^{i}_{nat}}{\Omega_{cp}} + \frac{\gamma^{i}_{ids}+\gamma^{i}_{nat}}{\delta_{tl}} * h_{tl}\right)$, where γ^{i}_{ids} and γ^{i}_{nat} are the workloads for IDS and NAT operation, respectively. Θ^{eh}_{mp} is the virtual work node–based e-healthcare task data processing delay that includes digital twin–based healthcare disease prediction $\left(\Theta^{eh}_{mp} = \frac{\gamma^{eh}_{mp}}{\Omega_{cp}}\right)$, where γ^{eh}_{mp} is the workload for e-healthcare task data processing at the digital twin server. Θ^{eh}_{std} is a digital twin server–processed task result data transfer delay to the doctor device. $\Theta^{eh}_{std} = \frac{\Gamma^{eh}_{std}}{\delta_{ul}} * z_{wh} + \frac{\Gamma^{eh}_{std}}{\delta_{fl}} * z_{fh} + \Theta^{i}_{pnd} + \Theta^{i}_{wgd}$, where Γ^{eh}_{std} is the digital twin–based processed data size. Θ^{eh}_{dt} is the doctor device–based e-healthcare task data processing delay that includes prescription generation. $\left(\Theta^{eh}_{dt} = \frac{\gamma^{eh}_{dt}}{\Omega_{lp}}\right)$, where γ^{eh}_{dt} is the workload for e-healthcare task data processing at a doctor's device. Ω_{lp} is the processing speed for doctors' devices. Θ^{eh}_{stm} is the doctor device–processed task result data transfer delay to blockchain device. $\Theta^{eh}_{stm} = \frac{\Gamma^{eh}_{stm}}{\delta_{ul}} * z_{wh} + \frac{\Gamma^{eh}_{stm}}{\delta_{fl}} * z_{fh} + \Theta^{i}_{pnd} + \Theta^{i}_{wgd}$, where Γ^{eh}_{stm} is the doctor's device-based processed data size. Θ^{eh}_{bc} is the virtual blockchain work node-based block or medical transaction task data creation delay $\left(\Theta^{eh}_{bc} = \frac{\gamma^{eh}_{bc}}{\Omega_{cp}}\right)$, where γ^{eh}_{bc} is the workload for block creation and hashing at the blockchain MEC server. Θ^{eh}_{stv} is block data transfer delay to blockchain verifier device. $\Theta^{eh}_{stv} = \frac{\Gamma^{eh}_{stv}}{\delta_{ul}} * z_{wh} + \frac{\Gamma^{eh}_{stv}}{\delta_{fl}} * z_{fh} + \Theta^{i}_{pnd} + \Theta^{i}_{wgd}$, where Γ^{eh}_{stv} is the transferred block data size. Θ^{eh}_{bv} is the virtual blockchain verifier work node–based block verification delay. $\left(\Theta^{eh}_{bv} = \frac{\gamma^{eh}_{bv}}{\Omega_{cp}}\right)$, where γ^{eh}_{bv} is the workload for block verification at the blockchain verifier device. Θ^{eh}_{ulb} is the block update delay on the blockchain ledger. $\left(\Theta^{eh}_{ulb} = \frac{\gamma^{eh}_{ulb}}{\Omega_{cp}} + \frac{\Gamma^{eh}_{vd}}{\delta_{ul}} * z_{wh} + \frac{\Gamma^{eh}_{vd}}{\delta_{fl}} * z_{fh} + \Theta^{i}_{pnd} + \Theta^{i}_{wgd}\right)$, where γ^{eh}_{ulb} is the workload for ledger update at the primary blockchain node. Γ^{eh}_{vd} is the transferred verified block data size. Θ^{eh}_{stu} is a verified e-healthcare result or updated block data transfer delay to the user device. $\Theta^{eh}_{stu} = \frac{\Gamma^{eh}_{stu}}{\delta_{ul}} * z_{wh} + \frac{\Gamma^{eh}_{stu}}{\delta_{fl}} * z_{fh} + \Theta^{i}_{pnd} + \Theta^{i}_{wgd}$, where Γ^{eh}_{stu} is the user-accessed e-healthcare data size. Θ^{eh}_{tvnf} is the third VNF processing delay that includes FW, IDS, and NAT processing delays at the user receiver device. $\left(\Theta^{eh}_{tvnf} = \frac{\gamma^{i}_{ids}+\gamma^{i}_{fw}+\gamma^{i}_{nat}}{\Omega_{lp}} + \frac{\gamma^{i}_{ids}+\gamma^{i}_{fw}+\gamma^{i}_{nat}}{\delta_{tl}} * h_{tl}\right)$, where γ^{i}_{ids}, γ^{i}_{fw}, and γ^{i}_{nat} are the workloads for IDS, FW, and NAT operation. Θ^{eh}_{rdd} is the receiver device–based e-healthcare result data display delay. $\left(\Theta^{eh}_{rdd} = \frac{\gamma^{eh}_{rd}}{\Omega_{lp}}\right)$, where γ^{eh}_{rd} is the workload for receiver device–based data display.

Next, this paper discusses the SFC delay (task realization delay) Θ^{i}_{vs} for video download–based non-6G applications using Eq. 14 (e.g., eMBB).

$$\begin{aligned}\Theta^{i}_{vs} = \sum_{i=1}^{y}\Big(&\Theta^{i}_{urs} + \Theta^{vs}_{vnf} + \Theta^{i}_{tis} + \Theta^{vs}_{svnf} + \\ &\Theta^{vs}_{mp} + \Theta^{vs}_{tvnf} + \Theta^{vs}_{fud}\Big).\end{aligned} \tag{14}$$

Θ^{i}_{urs} is a task request forwarding delay to the network slicing manager from the user device. Θ^{vs}_{vnf} is the initial VNF processing delay that includes FW, DPI, and NAT operation. $\left(\Theta^{vs}_{vnf} = \frac{\gamma^{i}_{fw}+\gamma^{i}_{dpi}+\gamma^{i}_{nat}}{\Omega_{cp}} + \frac{\Gamma^{i}_{tds}}{\delta_{tl}} * h_{tl}\right)$, where $\gamma^{i}_{fw}, \gamma^{i}_{dpi}, \gamma^{i}_{nat}$ are the workloads for FW, DPI, and NAT operation. Θ^{i}_{tis} is task instruction and work node selection information sending delay to the worker node $\left(\Theta^{i}_{tis} = \frac{\Gamma^{i}_{iid}}{\delta_{ul}} * z_{wh} + \frac{\Gamma^{i}_{iid}}{\delta_{fl}} * z_{fh} + \Theta^{i}_{pnd} + \Theta^{i}_{wgd}\right)$, where Γ^{i}_{iid} is the data size associated with task instruction and work node selection. Θ^{vs}_{svnf} is the second VNF processing delay at the cloud server that includes LB and IDS processing. $\left(\Theta^{vs}_{svnf} = \frac{\gamma^{i}_{lb}+\gamma^{i}_{ids}}{\Omega_{cp}} + \frac{\Gamma^{i}_{tds}}{\delta_{tl}} * h_{tl}\right)$, where γ^{i}_{lb} and γ^{i}_{ids} are the workloads for LB and IDS operation processing at the MEC server, respectively. Θ^{vs}_{mp} is the virtual work node–based cache look-up and video file preparation for download at the MEC server for video download-based tasks. $\Theta^{vs}_{mp} = \frac{\gamma^{vs}_{mp}}{\Omega_{cp}} + u * (\Theta^{vs}_{ecd} + \Theta^{vs}_{rcd})$, where γ^{vs}_{mp} is the workload for cache lookup and video file access at the MEC server. u is the cache miss ratio (u = 0 or 1). Θ^{vs}_{ecd} and Θ^{vs}_{rcd} are

communication delays associated with edge cloud-based and remote cloud-based video access, respectively (Drolia et al., 2017). Θ^{vs}_{tvnf} is the third VNF processing delay that includes IDS and NAT processing delays at the user receiver device. $\left(\Theta^{vs}_{tvnf} = \frac{\gamma^i_{ids}+\gamma^i_{nat}}{\Omega_{lp}} + \frac{\gamma^i_{ids}+\gamma^i_{nat}}{\delta_{tl}} * h_{tl}\right)$, where γ^i_{ids} and γ^i_{nat} are the workloads for IDS and NAT operation. Θ^{vs}_{fud} is the video file download delay from the virtual work node by the receiver device. $\Theta^{vs}_{fud} = \frac{\Gamma^{vs}_{tod}}{\delta_{wl}} * z_{wh} + \frac{\Gamma^{vs}_{tod}}{\delta_{fl}} * z_{fh} + \Theta^i_{pnd} + \Theta^i_{wgd}$, where Γ^{vs}_{tod} is the downloaded video file data size.

Now, this paper calculates the SFC delay Θ^i_{xa} (task realization delay) associated with XR-based education learning non-6G applications using Eq. 15 (e.g., eMBB).

$$\Theta^i_{xa} = \sum_{i=1}^{y}\Big(\Theta^i_{urs} + \Theta^{xa}_{vnf} + \Theta^i_{tis} + \Theta^{xa}_{ucd} + \Theta^{xa}_{ofd} + \Theta^{xa}_{svnf} + \Theta^{xa}_{mp} + \Theta^{xa}_{str} + \Theta^{xa}_{tvnf} + \Theta^{xa}_{rpr}\Big). \quad (15)$$

Θ^i_{urs} is a task request sending or dispatch delay to the network slicing manager from the user. Θ^{xa}_{vnf} is the initial VNF processing delay that includes FW, DPI, and NAT operation. $\left(\Theta^{xa}_{vnf} = \frac{\gamma^i_{fw}+\gamma^i_{dpi}+\gamma^i_{nat}}{\Omega_{cp}} + \frac{\Gamma^i_{ids}}{\delta_{tl}} * h_{tl}\right)$.

Θ^i_{tis} is the task instruction and work node selection information convey delay to the worker node. $\Theta^i_{tis} = \frac{\Gamma^i_{iid}}{\delta_{wl}} * z_{wh} + \frac{\Gamma^i_{tid}}{\delta_{fl}} * z_{fh} + \Theta^i_{pnd} + \Theta^i_{wgd}$). Θ^{xa}_{ucd} is the user XR device (e.g., VR glass or headset)–based data collection (e.g., airplane image) delay from the environment. $\left(\Theta^{xa}_{ucd} = \frac{\gamma^{xa}_{ldc}}{\Omega_{lp}}\right)$, where γ^{xa}_{ldc} is the user workload for task input data collection. Θ^{xa}_{ofd} is the user-captured data offload delay to a virtual worker at the MEC server. $\Theta^{xa}_{ofd} = \frac{\Gamma^{xa}_{iid}}{\delta_{wl}} * z_{wh} + \frac{\Gamma^{xa}_{tid}}{\delta_{fl}} * z_{fh} + \Theta^i_{pnd} + \Theta^i_{wgd}$, where Γ^{xa}_{tid} is the offloaded data size. Θ^{xa}_{svnf} is the second VNF processing delay that includes IDS and NAT processing delays at the MEC server. $\left(\Theta^{xa}_{svnf} = \frac{\gamma^i_{ids}+\gamma^i_{nat}}{\Omega_{cp}} + \frac{\gamma^i_{ids}+\gamma^i_{nat}}{\delta_{tl}} * h_{tl}\right)$, where γ^i_{ids} and γ^i_{nat} are the workloads for IDS and NAT operation, respectively. Θ^{xa}_{mp} is the virtual work node-based XR task data processing delay that includes rendering, object detection from the image, and audio/visual virtual data adding delay. $\left(\Theta^{xa}_{mp} = \frac{\gamma^{xa}_{mp}}{\Omega_{cp}}\right)$, where γ^{xa}_{mp} is the workload for XR task data processing at the MEC server. Θ^{xa}_{str} is MEC-processed XR task resulting data transfer delay to the user device. $\Theta^{xa}_{str} = \frac{\Gamma^{xa}_{tod}}{\delta_{wl}} * z_{wh} + \frac{\Gamma^{xa}_{tod}}{\delta_{fl}} * z_{fh} + \Theta^i_{pnd} + \Theta^i_{wgd}$, where Γ^{xa}_{tod} is the MEC-processed XR task result data size (e.g., airplane detection result). Θ^{xa}_{tvnf} is the third VNF processing delay that includes IDS, FW, and NAT processing at the receiver device. $\left(\Theta^{xa}_{tvnf} = \frac{\gamma^i_{ids}+\gamma^i_{fw}+\gamma^i_{nat}}{\Omega_{lp}} + \frac{\gamma^i_{ids}+\gamma^i_{fw}+\gamma^i_{nat}}{\delta_{tl}} * h_{tl}\right)$, where γ^i_{ids}, γ^i_{fw}, and γ^i_{nat} are the workloads for IDS, FW, and NAT operation at the receiver XR device, respectively. Θ^{xa}_{rpr} is the user/receiver XR device-based data processing delay that includes data visualization. $\left(\Theta^{xa}_{rpr} = \frac{\gamma^{xa}_{rp}}{\Omega_{lp}}\right)$, where γ^{xa}_{rp} is the workload for receiver device-based XR data reception.

Next, we compute the SFC delay Θ^i_{sa} (task realization delay) associated with smart agriculture-based non-6G applications (e.g., mMTC). In this application, IoT sensors can upload agriculture data to the MEC server. The MEC server transfers the agriculture-related suggestion to the farmer's or user's device after processing. Θ^i_{sa} is given by using Eq. 16.

$$\Theta^i_{sa} = \sum_{i=1}^{y}\Big(\Theta^i_{urs} + \Theta^{sa}_{vnf} + \Theta^i_{tis} + \Theta^{sa}_{ucd} + \Theta^{sa}_{ofd} + \Theta^{sa}_{svnf} + \Theta^{sa}_{mp} + \Theta^{sa}_{str} + \Theta^{sa}_{tvnf} + \Theta^{sa}_{rpr}\Big). \quad (16)$$

Θ^i_{urs} is a smart agriculture task request sending or dispatch delay to the network slicing manager from the user device. Θ^{sa}_{vnf} is the initial VNF processing delay that includes FW, DPI, and NAT operation. $\left(\Theta^{sa}_{vnf} = \frac{\gamma^i_{fw}+\gamma^i_{dpi}+\gamma^i_{nat}}{\Omega_{cp}} + \frac{\Gamma^i_{tds}}{\delta_{tl}} * h_{tl}\right)$. Θ^i_{tis} is task instruction and work node selection information convey delay to the selected worker node $\left(\Theta^i_{tis} = \frac{\Gamma^i_{iid}}{\delta_{wl}} * z_{wh} + \frac{\Gamma^i_{tid}}{\delta_{fl}} * z_{fh} + \Theta^i_{pnd} + \Theta^i_{wgd}\right)$. Θ^{sa}_{ucd} is the IoT sensor-based agriculture data collection regarding crops (e.g., humidity, crop image, temperature, moisture). $\Theta^{sa}_{ucd} = \frac{\gamma^{sa}_{ldc}}{\Omega_{lp}}$), where γ^{sa}_{ldc} is the IoT device workload for task input data collection (e.g., humidity, crop image, temperature, and moisture). Θ^{sa}_{ofd} is an IoT device that captured data offload delay for a virtual worker at MEC. $\Theta^{sa}_{ofd} = \frac{\Gamma^{sa}_{iid}}{\delta_{wl}} * z_{wh} + \frac{\Gamma^{sa}_{tid}}{\delta_{fl}} * z_{fh} + \Theta^i_{pnd} + \Theta^i_{wgd}$, where Γ^{sa}_{tid} is the offloaded data size. Θ^{sa}_{svnf} is the second VNF processing delay that includes IDS and NAT processing at the MEC server. $\left(\Theta^{sa}_{svnf} = \frac{\gamma^i_{ids}+\gamma^i_{nat}}{\Omega_{cp}} + \frac{\gamma^i_{ids}+\gamma^i_{nat}}{\delta_{tl}} * h_{tl}\right)$, where γ^i_{ids} and γ^i_{nat} are the workloads for IDS and NAT operation processing, respectively. Θ^{sa}_{mp} is the virtual work node–based smart agriculture task data processing delay that includes IoT data processing and suggestion generation regarding agriculture problems such as irrigation plans and fertilizer plans. $\Theta^{sa}_{mp} = \frac{\gamma^{sa}_{mp}}{\Omega_{cp}}$), where γ^{sa}_{mp} is the workload for data processing at the MEC server. Θ^{sa}_{str} is MEC-processed task result data transfer delay to the user device. $\Theta^{sa}_{str} = \frac{\Gamma^{sa}_{tod}}{\delta_{wl}} * z_{wh} + \frac{\Gamma^{sa}_{tod}}{\delta_{fl}} * z_{fh} + \Theta^i_{pnd} + \Theta^i_{wgd}$, where Γ^{sa}_{tod} is the MEC-processed smart agriculture task result data size (e.g., crop disease). Θ^{sa}_{tvnf} is the third VNF processing delay that includes IDS and FW processing delay at the user receiver device. $\left(\Theta^{sa}_{tvnf} = \frac{\gamma^i_{ids}+\gamma^i_{fw}}{\Omega_{lp}} + \frac{\gamma^i_{ids}+\gamma^i_{fw}}{\delta_{tl}} * h_{tl}\right)$, where γ^i_{ids} and γ^i_{fw} are the workloads for IDS and FW operation at the receiver device, respectively. Θ^{sa}_{rpr} is the receiver device–based task result data visualization delay $\left(\Theta^{sa}_{rpr} = \frac{\gamma^{sa}_{rp}}{\Omega_{lp}}\right)$, where γ^{sa}_{rp} is the workload for receiver device–based data reception.

Last, we present the SFC delay Θ^i_{su} (task realization delay) associated with video surveillance-based non-6G applications (e.g., mMTC). In this application, video cameras capture and upload video data to the MEC server. The MEC server processes and transfers the security threat information to the user's device. Θ^i_{su} is given by using Eq. 17.

$$\Theta^i_{su} = \sum_{i=1}^{y}\Big(\Theta^i_{urs} + \Theta^{su}_{vnf} + \Theta^i_{tis} + \Theta^{su}_{ucd} + \Theta^{su}_{ofd} + \Theta^{su}_{svnf} + \Theta^{su}_{mp} + \Theta^{su}_{str} + \Theta^{su}_{tvnf} + \Theta^{su}_{rpr}\Big). \quad (17)$$

Θ^i_{urs} is a video surveillance–based task request sending/dispatch delay to the slicing manager from the user device. Θ^{su}_{vnf} is the initial VNF processing delay that includes FW, IDS, and NAT operation. $\left(\Theta^{su}_{vnf} = \frac{\gamma^i_{fw}+\gamma^i_{ids}+\gamma^i_{nat}}{\Omega_{cp}} + \frac{\Gamma^i_{tds}}{\delta_{tl}} * h_{tl}\right)$. Θ^i_{tis} is task instruction and work node selection information convey delay to the selected worker node $\left(\Theta^i_{tis} = \frac{\Gamma^i_{iid}}{\delta_{wl}} * z_{wh} + \frac{\Gamma^i_{tid}}{\delta_{fl}} * z_{fh} + \Theta^i_{pnd} + \Theta^i_{wgd}\right)$. Θ^{su}_{ucd} is the video camera–based task location data collection regarding security threat detection. $\Theta^{su}_{ucd} = \frac{\gamma^{su}_{ldc}}{\Omega_{lp}}\right)$, where γ^{su}_{ldc} is the workload for video surveillance task input data collection by a video camera. Θ^{su}_{ofd} is a video camera device based on captured data offload delay to a virtual worker at MEC. $\Theta^{su}_{ofd} = \frac{\Gamma^{su}_{iid}}{\delta_{wl}} * z_{wh} + \frac{\Gamma^{su}_{tid}}{\delta_{fl}} * z_{fh} + \Theta^i_{pnd} + \Theta^i_{wgd}$, where Γ^{su}_{tid} is the offloaded data size. Θ^{su}_{svnf} is the second VNF processing delay that includes IDS and NAT processing delays at the MEC server. $\left(\Theta^{su}_{svnf} = \frac{\gamma^i_{ids}+\gamma^i_{nat}}{\Omega_{cp}} + \frac{\gamma^i_{ids}+\gamma^i_{nat}}{\delta_{tl}} * h_{tl}\right)$, where γ^i_{ids} and γ^i_{nat} are the workloads for IDS and NAT operation, respectively. Θ^{su}_{mp} is the MEC-based video surveillance task data processing delay that includes object detection, scenario analysis, and security threat detection. $\Theta^{su}_{mp} = \frac{\gamma^{su}_{mp}}{\Omega_{cp}}\right)$, where γ^{su}_{mp} is the workload for MEC server.

Θ^{su}_{str} is MEC-processed video surveillance task result transfer delay to the user. $\Theta^{su}_{str} = \frac{\Gamma^{su}_{tod}}{\delta_{wl}} * z_{wh} + \frac{\Gamma^{sa}_{tod}}{\delta_{fl}} * z_{fh} + \Theta^{i}_{pnd} + \Theta^{i}_{wgd}$, where Γ^{su}_{tod} is the video surveillance task result or data size (e.g., object detection and security threat detection). Θ^{su}_{tvnf} is the third VNF processing delay that includes IDS and NAT processing delay at the receiver device. $\left(\Theta^{su}_{tvnf} = \frac{\gamma^{i}_{ids}+\gamma^{i}_{nat}}{\Omega_{lp}} + \frac{\gamma^{su}_{ids}+\gamma^{su}_{nat}}{\delta_{tl}} * h_{tl}\right)$, where γ^{i}_{ids} and γ^{i}_{nat} are the workloads for IDS and NAT operation, respectively. Θ^{su}_{rpr} is the user/receiver device–based task result data visualization delay for the video surveillance task. $\left(\Theta^{su}_{rpr} = \frac{\gamma^{su}_{rp}}{\Omega_{lp}}\right)$, where γ^{su}_{rp} is the workload for receiver device–based data reception.

4.2 User energy usage value for task implementation

The user device energy usage value (ξ^{i}_{euc}) for different 6G and non-6G task implementations includes the energy expense value for task data transmission, task data receiving, resource waiting, virtual work node–based task work processing, and a user client device–based or physical work node–based work processing (see Eq. 18).

$$\begin{aligned}\xi^{i}_{euc} = &\sum_{i=1}^{y_{ts}}\left(\kappa^{i}_{tm} * \Theta^{i}_{tm} + \kappa^{i}_{re} * \Theta^{i}_{re} + \kappa^{i}_{sdp} * \Theta^{i}_{sdp}\right) \\ &+ \sum_{i=1}^{y_{ts}}\left(\kappa^{i}_{vnp} * \Theta^{i}_{vnp} + \kappa^{i}_{wd} * \Theta^{i}_{rwd}\right) \\ &+ \sum_{i=1}^{y_{cs}}\left(\kappa^{i}_{tm} * \Theta^{i}_{tm} + \kappa^{i}_{re} * \Theta^{i}_{re} + \kappa^{i}_{sdp} * \Theta^{i}_{sdp}\right) \\ &+ \sum_{i=1}^{y_{cs}}\left(\kappa^{i}_{vnp} * \Theta^{i}_{vnp} + \kappa^{i}_{wd} * \Theta^{i}_{rwd}\right),\end{aligned} \tag{18}$$

where y_{ts} and y_{cs} are the total implemented users' time-saving priority tasks and cost-saving priority tasks, respectively. κ^{i}_{tm}, κ^{i}_{re}, κ^{i}_{sdp}, κ^{i}_{vnp}, and κ^{i}_{wd} are the average energy expense values (per millisecond) regarding task data transmission operation, receive operation, user device/physical work node–based workload processing, virtual work node–based workload processing, and waiting time for resource or workload access, respectively. Θ^{i}_{tm}, Θ^{i}_{re}, Θ^{i}_{sdp}, Θ^{i}_{vnp}, and Θ^{i}_{rwd} are the delays regarding task data transmission operation, reception operation, user device/physical work node–based workload processing, virtual work node–based workload processing, and resource or workload access, respectively.

4.3 QoS guarantee ratio

The QoS guarantee ratio ϕ^{i}_{qgr} is investigated by calculating the ratio of the total 6G/non-6G task that satisfies task requirements and the total 6G/non-6G task that requests service execution. In this work, the QoS ratio can be defined as the task execution time limit or deadline satisfaction ratio. In other words, QoS for any task execution is the ability of the network to satisfy the task execution deadline limit.

The QoS guarantee ratio ϕ^{i}_{qgr} is measured by using Eq. 19.

$$\phi^{i}_{qgr} = \frac{\sum_{i=1}^{y}\Lambda^{i}_{tt} - \left(\sum_{i=1}^{y_{ts}}\Lambda^{i}_{ust} + \sum_{i=1}^{y_{cs}}\Lambda^{i}_{ust}\right)}{\sum_{i=1}^{y}\Lambda^{i}_{tt}} \tag{19}$$

where Λ^{i}_{tt} is the total 6G and non-6G task number count based on arrival. Λ^{i}_{ust} is the total unsuccessful task that misses the task deadline. y_{ts} and y_{cs} are the total implemented users' time-saving priority tasks and cost-saving priority tasks, respectively. y is the total user task number ($y = y_{ts} + y_{cs}$).

4.4 Maximum achievable throughput

The maximum achievable throughput Π^{i}_{mat} can be investigated by calculating the ratio of total task data amount (both input task data and output task data) and maximum time span delay value regarding task implementation.

Π^{i}_{mat} is computed by using Eq. 20.

$$\Pi^{i}_{mat} = \frac{\sum_{i=1}^{y}\left(\Gamma^{i}_{tin} + \Gamma^{i}_{tot} + \Gamma^{i}_{odt}\right)}{\sum_{i=1}^{y}\Delta^{i}_{mpd}}, \tag{20}$$

where Γ^{i}_{tin}, Γ^{i}_{tot}, Γ^{i}_{odt}, and Δ^{i}_{mpd} are the total exchanged 6G and non-6G task input data amount, total exchanged task output or result data amount, other data amount, and time span delay for task implementation, respectively. The maximum time span delay value (Δ^{i}_{mpd}) can be defined by taking the maximum task implementation delay among all arrived task executions or the maximum task implementation delay to complete all arrived task requests (y) at the slicing manager. The maximum time span delay value for the number of tasks is given by $\Delta^{i}_{mpd} = \max\{\Delta^{1}_{tid}, \Delta^{2}_{tid}, \Delta^{y}_{tid}\}$, $i \in 1, 2, y$, where y is the total task number. Δ^{i}_{tid} is the minimum possible task implementation delay for a single task execution with the selected resource node (please see Section 4.1 for each task implementation delay calculation).

4.5 Users' service execution monetary cost

The service execution cost for users' task implementation (μ^{i}_{secu}) can be obtained by taking the monetary sum value regarding network resource use, virtual work node or cloud resource use, physical or client device use, and waiting delay during resource usage service. μ^{i}_{secu} is determined by using Eq. 21.

$$\begin{aligned}\mu^{i}_{secu} = &\sum_{i=1}^{y_{ts}}\left(\pi^{i}_{nru} * \Theta^{i}_{tm} + \pi^{i}_{nru} * \Theta^{i}_{re} + \pi^{i}_{sdp} * \Theta^{i}_{sdp}\right) \\ &+ \sum_{i=1}^{y_{ts}}\left(\pi^{i}_{vnp} * \Theta^{i}_{vnp} + \pi^{i}_{wd} * \Theta^{i}_{rwd}\right) \\ &+ \sum_{i=1}^{y_{cs}}\left(\pi^{i}_{nru} * \Theta^{i}_{tm} + \pi^{i}_{nru} * \Theta^{i}_{re} + \pi^{i}_{sdp} * \Theta^{i}_{sdp}\right) \\ &+ \sum_{i=1}^{y_{cs}}\left(\pi^{i}_{vnp} * \Theta^{i}_{vnp} + \pi^{i}_{wd} * \Theta^{i}_{rwd}\right),\end{aligned} \tag{21}$$

where y_{ts} and y_{cs} are the total implemented users' time-saving priority tasks and cost-saving priority tasks, respectively. π^{i}_{nru}, π^{i}_{sdp}, π^{i}_{vnp}, and π^{i}_{wd} are the average monetary expense values (per millisecond) regarding network bandwidth resource usage, user device/physical work node–based resource usage, virtual work node–based resource usage, and waiting time for resource or during work node access, respectively.

4.6 Internet service providers' and cloud providers' profit

The total task execution profit for internet service providers (ISPs) and that for the cloud providers can be determined by taking the difference ($\chi^i_{spp} = \sum_{i=1}^{y} \mu^i_{secu} - \chi^i_{spc}$) between the collected revenue from users (μ^i_{secu}) for task implementation services and the ISP/cloud providers' monetary cost (χ^i_{spc}) regarding service continuation (e.g., resource buy and maintenance). ISP and cloud service provider costs are determined by using Eq. 22.

$$\begin{aligned}\chi^i_{spc} = &\sum_{i=1}^{y_{ts}} \left(\tau^i_{nru} * \Theta^i_{tm} + \tau^i_{nru} * \Theta^i_{re} + \tau^i_{sdp} * \Theta^i_{sdp}\right) \\ &+ \sum_{i=1}^{y_{ts}} \left(\tau^i_{vnp} * \Theta^i_{vnp} + \tau^i_{wd} * \Theta^i_{rwd}\right) \\ &+ \sum_{i=1}^{y_{cs}} \left(\tau^i_{nru} * \Theta^i_{tm} + \tau^i_{nru} * \Theta^i_{re} + \tau^i_{sdp} * \Theta^i_{sdp}\right) \\ &+ \sum_{i=1}^{y_{cs}} \left(\tau^i_{vnp} * \Theta^i_{vnp} + \tau^i_{wd} * \Theta^i_{rwd}\right), \end{aligned} \tag{22}$$

where τ^i_{nru}, τ^i_{sdp}, τ^i_{vnp}, and τ^i_{wd} are the average monetary expense value (per ms) for service providers regarding network bandwidth resource use, user device/physical work node–based resource usage, virtual work node–based resource usage, and waiting time, respectively.

4.7 User and service providers' welfare value

User welfare for task implementation value χ^i_{uw} is obtained by taking the summation of users' task implementation time gain, energy usage gain, and service execution monetary cost gain. χ^i_{uw} is determined by using Eq. 23.

$$\begin{aligned}\chi^i_{uw} &= \sum_{i=1}^{y} \chi^i_{tw} + \chi^i_{ew} + \chi^i_{mw} \\ &= \frac{\sum_{i=1}^{y} \Delta^i_{td} - \Delta^i_{md}}{\sum_{i=1}^{y} \Delta^i_{td}} + \frac{\sum_{i=1}^{y} \xi^i_{eb} - \xi^i_{euc}}{\sum_{i=1}^{y} \xi^i_{eb}} + \frac{\sum_{i=1}^{y} \mu^i_{mb} - \mu^i_{secu}}{\sum_{i=1}^{y} \mu^i_{mb}}, \end{aligned} \tag{23}$$

where Δ^i_{td}, ξ^i_{eb}, and μ^i_{mb} are the user's task implementation deadline, energy budget, and monetary budget for task implementation, respectively. Δ^i_{md}, ξ^i_{euc}, and μ^i_{secu} are the user's required task implementation time span delay during task execution, energy expense cost, and service execution monetary cost, respectively. ISP's and cloud service providers' welfare (χ^i_{spw}) for task implementation value is obtained by taking the summation of service execution monetary cost gain. χ^i_{spw} is estimated by $\chi^i_{uw} = \sum_{i=1}^{y} \chi^i_{spp} = \frac{\sum_{i=1}^{y} \chi^i_{secu} - \chi^i_{spc}}{\sum_{i=1}^{y} \chi^i_{spc}}$, where χ^i_{secu} and χ^i_{spc} are the ISPs' and cloud providers' revenue and monetary cost, respectively.

4.8 User's survived energy

The user's average survived energy Υ^i_{use} is measured by taking the ratio of the total remaining energy to the total number of user devices. Υ^i_{use} is investigated by $\Upsilon^i_{use} = \frac{\sum_{i=1}^{y} \sigma^i_{te} - y_{at} * \xi^i_{aec} * \rho}{\sum_{i=1}^{y} \Xi^i_{tn}}$, where σ^i_{te}, y_{at}, ξ^i_{aec}, ρ, and Ξ^i_{tn} are the summation of total user energy, active task number, average energy per user in each round, simulation round, and total user node number in a network, respectively.

4.9 Total number of capable or alive user devices

The total number of capable or alive user devices φ^i_{udn} is computed by taking the ratio of the total cost of energy by all user devices and the initial energy of a user device. φ^i_{udn} is estimated by $\varphi^i_{udn} = \frac{\sum_{i=1}^{y} y_{at} * \xi^i_{aec} * \rho}{\zeta^i_{ie}}$, where ζ^i_{ie}, y_{at}, ξ^i_{aec}, ρ, and Ξ^i_{tn} are the initial energy of a user device, active task number, average energy per user in each round and the simulation round, and the total user node number in a network, respectively.

5 Simulation results and analysis

Section 5 presents the comparison results of (*i*) the proposed time-first accelerator scheme with a minimum predicted computation delay, communication delay, and waiting delay–based network resource slicing, (*ii*) the proposed cost-first accelerator scheme with a minimum predicted service execution monetary cost–based network resource slicing, (*iii*) the traditional minimum communication delay–based network resource–slicing scheme (e.g., Sun et al., 2020; Siasi et al., 2020; Marotta et al., 2017; Cai et al., 2022; Przybylski et al., 2021), and (*iv*) the traditional computational power–based work node selection with casual task scheduling scheme (e.g., Zhang et al., 2015; Ma et al., 2022; Demchenko et al., 2015; Angui et al., 2022). The detailed simulation parameters (e.g., data size, workload, monetary cost, energy cost, and deadline) and the values associated with the different SFC tasks are discussed in Table 2.

For simulation, the total value of task count is varied between 26 and 195. The task data amount associated with users' task request, server-to-server dispatched amount during VNF processing, task instruction with virtual/physical work node selection are 1,024 bits, 512 bits, and 256 bits, respectively. The data size associated with metaverse-based avatar interaction task input, users' first and second avatar conversation data, holographic telepresence task input, MEC processed holographic task output, EV charging task input, MEC processed EV charging task output are chosen random basis within 1–20 KB, 1–20 KB, 1–10 MB, 1–10 MB, and 1–5 KB range, respectively. The distance from starting to the charging station point for EV movement, client devices' own movement speed, charging requirement, and available charge value data are varied randomly within 100–500 m, 50–80 m/s, 0.15–0.6 KWh, and 0.2–0.6 KWh, respectively. The battery depletion threshold, battery capacity, and EV charging/discharging rate are 0.15, 50 KWh, and 45 KW, respectively. The data size associated with offloaded brain–computer interaction task, MEC-processed brain–computer interaction task result, offloaded haptic feedback-based gaming task, MEC-processed haptic feedback task result, offloaded human–robot interaction-based automation task, human users' processed task result for automation task, robots' second offloaded task data, and robots' second inspection data to the human users' device are chosen on random basis within 5–50 KB,

TABLE 2 Simulation notations or parameters with values.

Parameter	Values/units
Γ^i_{nib}, Γ^i_{ucr}, Γ^i_{urr}, Γ^i_{umsr}, Γ^i_{uaa}, and Γ^i_{smtrs}, Θ^i_{pnd}, and Θ^i_{wgd}, Ω_{cp}/Ω_{lp}, t_{tldd}	Beacon message size (160 bits), internet connectivity request message size (160 bits), connectivity response message size (160 bits), service registration and access request message (192 bits), user authentication and user approval message size (192 bits), and task request slot schedule message size (192 bits), total propagation delay (ms, vary) and waiting (ms, vary) delay, work processing speed for the virtual worker (32–45 GHz)/user device (1,000–2000 MHz), and time limit or deadline for task execution (3,000–73,500 ms)
$\gamma^{nsm}_{uwip}/\gamma^{ud}_{uwip}$, δ_{wl}/δ_{fl}, z_{fh}/z_{wh}, $\Gamma^i_{isdr}/\Gamma^i_{isres}$, Γ^i_{utr}, Γ^i_{rurs}, Γ^i_{uris}, γ^{nsm}_{urg}, γ^{wd}_{urg}, $\Gamma^i_{nsd}/\Gamma^i_{rsd}$, γ^i_{nsd}	Network slicing manager/user device workload for network inauguration (10 CPU cycles/bit), wireless link (10–20 Gbps for terahertz, 5–7 Gbps for mmWave, 50–500 Mbps for microwave, 10,000–40,000 Mbps for WLAN link)/fiber link data rate (20–40 Gbps), hop for per fiber (1–10, vary)/wireless link (1–5, vary), inter SDN controller request (160 bits)/response (160 bits), message size regarding users' task request message (1 KB), resource info. request (160 bits), updated resource info. reply message (192 bits), network slicing manager workload (10 CPU cycles/bit)/worker workload (5 CPU cycles/bit) for user request and resource information gathering phase, broadcasted resource/worker allocation message size for the users/worker (192 bits), and the workload for network slicing manager (10 CPU cycles/bit) for network resource slicing
κ^i_{tm}, κ^i_{re}, κ^i_{sdp}, κ^i_{vmp}, and κ^i_{wd}, π^i_{nru}, π^i_{sdp}, π^i_{vmp}, and π^i_{wd}, τ^i_{nru}, τ^i_{sdp}, τ^i_{vmp}, and τ^i_{wd}, simulation area	Average energy expense value (per millisecond) regarding task data transmission operation (0.43 W), data receive operation (0.35 W), user device–based workload processing (0.52 W), virtual work node–based workload processing (0.0008 W), and waiting time (0.0005 W), average monetary expense value (per millisecond) regarding network bandwidth resource usage (0.4/0.3 USD for time-first/cost-first policy), user device–based resource usage (0.2/0.15 USD for time-first/cost-first policy), virtual work node–based resource usage (0.4/0.3 USD for time-first/cost-first policy), and waiting time (0.001 USD), average monetary expense value (per millisecond) for service providers regarding network bandwidth resource use (0.3/0.2 USD for time-first/cost-first policy), user device–based resource usage (0.15/0.1 USD for time-first/cost-first policy), virtual work node–based resource usage (0.3/0.2 USD for time-first/cost-first policy), and waiting time (0.0025/0.0015 USD for time-first/cost-first policy), 500 m × 500 m
Δ^i_{tid}, ξ^i_{euc}, ϕ^i_{qgr}, Π^i_{mat}, μ^i_{secu}, χ^i_{spp}, χ^i_{uw}, χ^i_{spw}, Υ^i_{use}, φ^i_{udn}, total user device, and task count	Task implementation delay (ms, vary), user devices energy usage value (mJ, vary), QoS guarantee ratio (0%–100%), maximum achievable throughput (Mbps, vary), service execution cost for users (USD, vary), total task execution profit for ISP and cloud providers (USD, vary), user welfare (0–100, vary), service providers welfare (0–100, vary), users average survived energy (mJ, vary), and the total number of the capable or alive user device (vary), 500, 26–195

5–50 KB, 1–20 MB, 1–20 MB, 1–20 KB, 1–20 KB, and 1–20 KB range, respectively. The task data amount for MEC-processed result in high-speed railway task, offloaded amount for high-speed railway task, transfer from users to the receiver device, global deep learning model for FL task, offloaded data size for FL task, finally updated global deep learning model, initial blockchain registration operation, e-healthcare task input, digital twin–based processed result, doctor device–processed result, transferred block data size, transferred verified block data size, a user accessed e-healthcare task result, and downloaded video files are varied between 1 and 10 KB, 1–10 KB, 1–50 KB, 1–15 MB, 1–5 MB, 1–15 MB, 1–15 MB, 5–25 KB, 5–25 KB, 5–25 KB, 5–25 KB, 5–25 KB, and 5–25 KB, respectively. The task data size for offloaded XR-based education learning task, MEC-processed XR task result, offloaded smart agriculture task, MEC-processed smart agriculture task result, offloaded video surveillance task, MEC-processed video surveillance task result are chosen on random basis within 1–10 MB, 1–10 MB, 1–20 KB, 1–20 KB, and 1–5 MB, respectively.

The workload amount for FW, DPI, and NAT operation, client device–based avatar creation data capture, IDS, avatar creation by MEC, avatar movement, client device–based conversation data capture for avatar, virtual node–based avatar conversation message–playing operation, second client device–based conversation data capture, and virtual node–based second avatar conversation message–playing operation are 300, 300, 300, 100, 300, 500, 100, 100, 500, and 100 CPU cycles/bit, respectively. The distance from one avatar to another during movement are chosen randomly between 5 and 500 m. The workload amount for client device–based holographic task data capture, MEC servers' holographic data processing, receiver device–based holographic data processing, receiver device–based holographic data visualization, client device–based electronic vehicle task data capture, EV charging task data processing at MEC server, brain data capture work, brain–computer interaction task data processing at MEC server, receiver device–based wheelchair movement operation, entering game state, haptic feedback task input data collection, haptic input task data processing at MEC server, and receiver device–based haptic feedback reception are 50, 1 K, 50, 10, 100, 1 K, 1 K, 1 K, 1 K, 50, 500, 1 K, and 10 CPU cycles/bit, respectively. The workload amount for supplying raw material to robots, user manufacturing operations, checking robots' work by using human devices, robot-based rechecking operations, checking robots' final work by using human devices, base station selection at the MEC server, load balancing operations, encryption operations at the receiver base stations, decryption

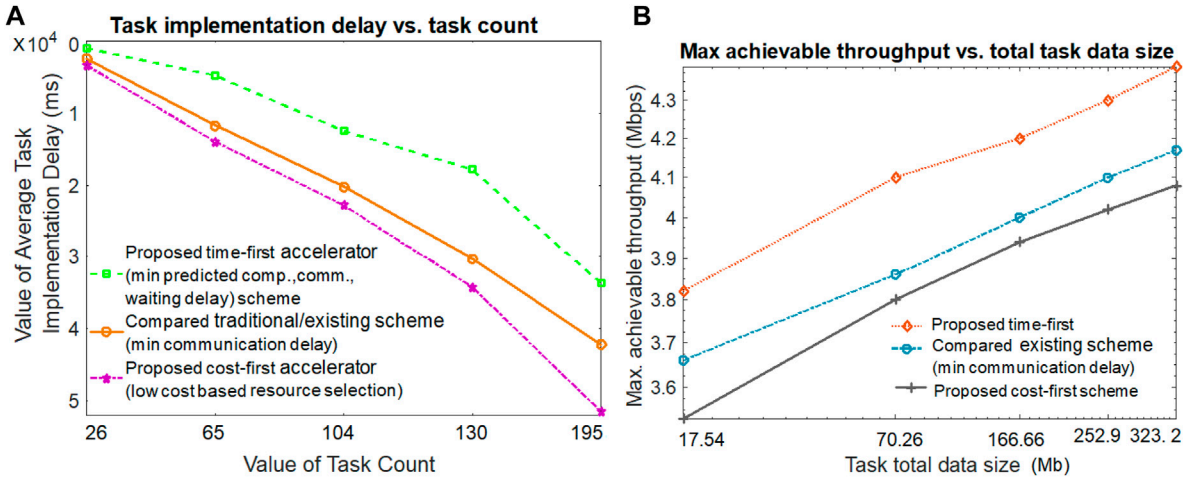

FIGURE 3
Task implementation delay and throughput achievable value.

operations, task output data displays at the receiver, client device–based road data captures, local model data training, global model updates, MEC blockchain server work and local client during preliminary blockchains, user e-healthcare data collection, e-healthcare task data processing at digital twin, e-healthcare task data processing at doctor devices, e-healthcare block creation, block verification, ledger update, receiver device–based e-health data display are 100, 100, 50, 100, 50, 1,000, 200, 200, 200, 100, 50, 50, 1 K, 100, 100, 100, 1 K, 100, 100, 100, 100, and 10 CPU cycles/bit, respectively. The workload amount for cache lookup, XR task input data collection, XR task data processing at MEC, receiver device during XR data reception, IoT device for task input data collection, agriculture task processing at MEC, receiver device–based agriculture data reception, video surveillance input data collection, video surveillance task data processing at MEC, video surveillance task resultant data reception are 50, 50, 1 K, 10, 500, 1 K, 100, 50, 1 K, and 10 CPU cycles/bit, respectively.

In Figure 3A, this work first compares the average task implementation delay (i.e., average task execution time) of the proposed time-first and cost-first scheme against the traditional scheme (minimum communication delay–based worker selection) by varying the task count number. The task implementation delay is defined by taking the sum of all delays associated with task initiation and task execution completion. As we can see from the figure, when the task count value is smaller, the task implementation delay is smaller in all proposed and compared schemes. The increment in task count value produces a higher task implementation delay in all proposed and compared schemes. The figure shows that the proposed time-first accelerator scheme produces the lowest task implementation delay when compared with the others. Although the proposed cost-first scheme receives the highest task implementation delay, and the compared (minimum communication delay) scheme offers the second-lowest task implementation delay. The proposed time-first accelerator scheme adopts the best virtual and physical worker selection with a minimum predicted computation delay, communication delay, and waiting delay for their SFC-based application execution. The proposed time-first scheme also ensures a task QoS guarantee by offering worker and resource allocation based on the smallest deadline on a task-first basis. On the contrary, the proposed cost-first scheme adopts worker and resource allocation based on their lowest payment costs, thus receiving the third place. If multiple workers or resources offer the same lowest cost, the best worker or communication resource is selected in the cost-first scheme based on their highest computation capability or link data rate. The compared scheme selects a worker or resource for application execution based on the lowest communication delay (i.e., the nearby worker or resource), thus receiving the second place. Due to its only communication delay–based selection, the task execution in the compared scheme may receive the second-highest waiting delay. The task implementation delay is the highest in the cost-first scheme because workers or resources with the lowest monetary cost are preferred (e.g., remote cloud), thus experiencing the highest computation delay, waiting delay, and communication delay. In Figure 3A, if the task number is 195, the average task implementation delay for the proposed time-first, compared scheme (minimum communication delay), and proposed cost-first scheme is 33,658 ms, 42,164 ms, and 51,441 ms, respectively.

The maximum achievable throughput for both proposed and compared methods by varying task total data size values is investigated in Figure 3B. The throughput value is calculated by taking the ratio of the total exchanged task data value and the makespan task execution delay value. Figure 3B hints that a large amount of the total task data exchange produces a higher achievable throughput value than its smaller amount task data counterparts in both the proposed and compared schemes. Hence, the task implementation makespan delay is the lowest in the proposed time-first scheme; it produces the highest achievable throughput value. Due to the second- and third-highest task implementation makespan delay values, the compared scheme (minimum communication delay) and proposed cost-first offer the second-best and third-best throughput values, respectively. In Figure 3B, if the task data size number is 252.9 Mb, the achievable throughput for

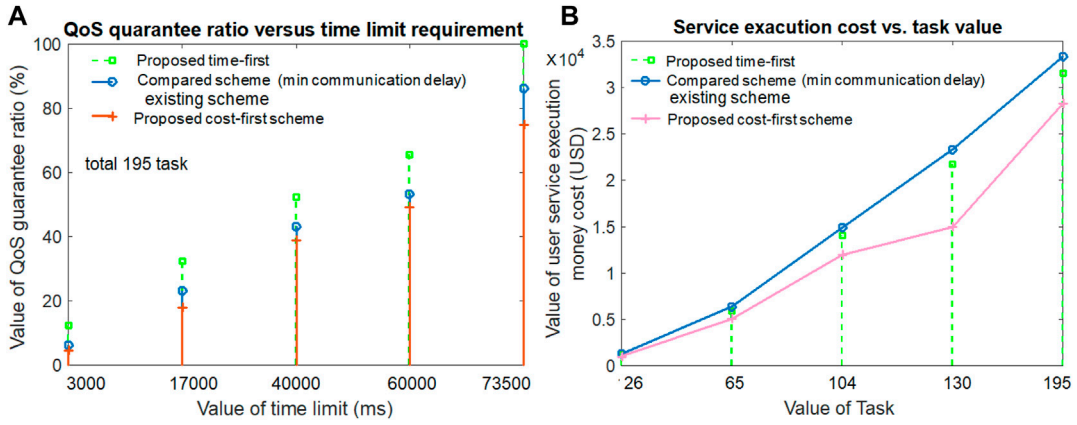

FIGURE 4
Service quality guarantee ratio and user money or service execution cost.

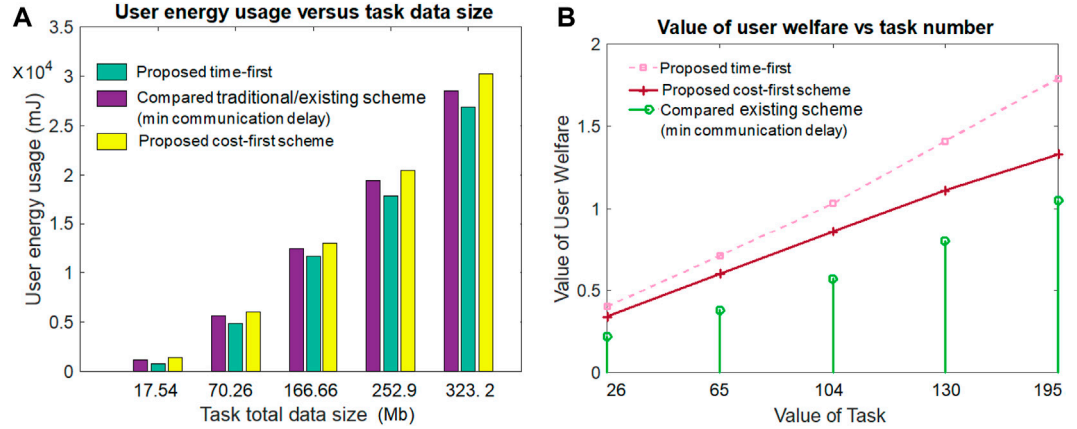

FIGURE 5
Energy usage and user welfare value.

the proposed time-first, compared scheme (minimum communication delay), and proposed cost-first scheme are 4.34 Mbps, 4.10 Mbps, and 3.95 Mbps, respectively.

Figure 4A examines the QoS guarantee ratio of our proposed scheme by varying the value of the time limit. It is shown in Figure 4A that when the task execution time limit is higher, the QoS guarantee ratio is also large in all three schemes (proposed and compared). When the task execution time limit is smaller, the QoS guarantee ratio is also smaller in both the proposed and compared schemes. The proposed time-first scheme offers the highest QoS guarantee ratio. This is because it experiences lower computation, communication, and waiting delays for task execution than the other compared schemes due to its appropriate resource and worker selection scheme. The cost-first scheme receives the third position as it experiences the highest computation, communication, and waiting delay for task execution compared to other schemes. The cost-first scheme receives the second position as it experiences the second-best computation and waiting delay for task execution. From Figure 4A, it can be noted that when the task time limit value is 73,500 ms, the achievable QoS guarantee ratios of the proposed time-first scheme, compared scheme (minimum communication delay), and proposed cost-first scheme are 100%, 86.15%, and 74.87%, respectively.

The service execution cost value for both the proposed scheme and the compared scheme by varying the value of the task number is highlighted in Figure 4B. The figure visualizes that a large amount of task execution requires the highest amount of service execution cost for users, and a small amount of task execution requires a comparatively lower amount of service execution payment cost for users in all three compared schemes. However, the proposed cost-first scheme allocates resources with the smallest monetary cost for task execution, thus offering better service execution cost results than others. The proposed time-first scheme requires resources for a smaller amount of time than the compared (minimum communication delay) scheme. Thus, the proposed time-first scheme gives the second-best service execution cost for task execution, and the compared scheme gives the third-best service execution cost results. From Figure 4B, it is shown that when the task value is 104, the service execution monetary payment cost for the

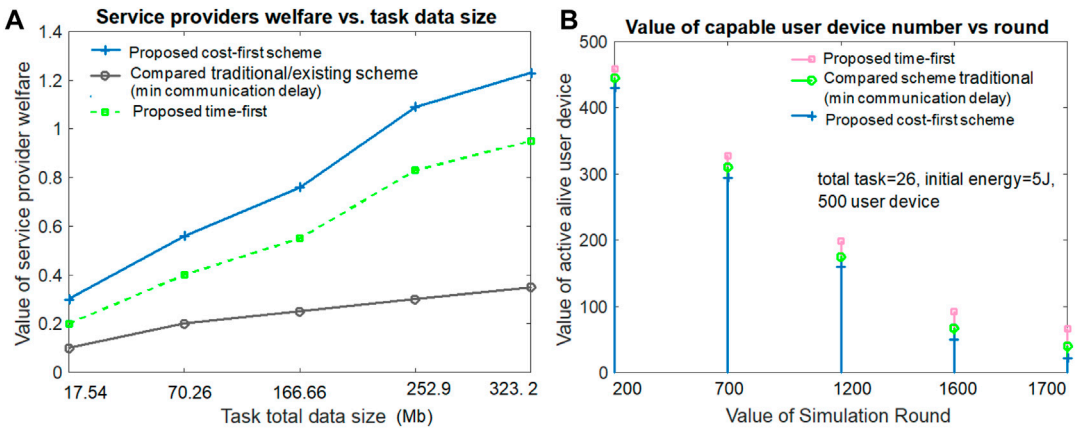

FIGURE 6
Service provider welfare and the alive user number.

proposed time-first scheme, compared scheme (minimum communication delay), and proposed cost-first scheme is 14,050 USD, 14,900 USD, and 11,960 USD, respectively.

Figure 5A gives the user's energy usage or expense value by varying the task data size for all three schemes. Figure 5A notes that the increment in task data size value requires a higher energy usage value than its smaller task data size counterparts in all three schemes. For both large and small task data sizes, the proposed time-first scheme requires the smallest user device energy usage value for task execution compared to both cost-first and compared schemes. The main reason behind this result is that the proposed time-first scheme experiences a lower amount of task implementation delay, thus requiring a smaller amount of energy than others. Although the cost-first scheme experiences a large amount of task implementation delay, thus requiring a higher amount of energy than others, the compared scheme (minimum communication delay) gives the second-best energy expense or usage value due to its second-best task implementation delay results. From Figure 5A, it can be deduced that when the task data size value is 323.2 Mb, the energy usage cost for the proposed time-first scheme, compared scheme (minimum communication delay), and proposed cost-first scheme is 27,626 mJ, 29,326 mJ, and 30,200 MJ, respectively.

Figure 5B notifies the results concerning the value of user welfare *versus* the task number for all three comparable schemes. Overall, the value of user welfare increases with the incremental value of task number. The user welfare value is determined by taking the sum of task implementation delay gain, energy usage gain, and service execution cost gain. The proposed time-first scheme produces a higher amount of user welfare value than both the proposed cost-first scheme and the compared scheme (min communication delay). The proposed cost-first scheme secures the second position and the compared scheme secures the third position in terms of the value of user welfare. The major reason behind this result is that the time-first scheme gives the highest task implementation delay gain, the highest energy usage gain, and the second highest service execution cost gain. Although the compared scheme (minimum communication delay) gives the highest service execution cost gain but the lowest task implementation delay gain and energy usage gain, the compared scheme gives the second highest task implementation delay gain, the second highest energy usage gain, and the lowest service execution cost gain. From Figure 5B, it is seen that when the task value is 65, the user welfare value for the proposed time-first, compared scheme (minimum communication delay), and proposed cost-first scheme are .71, .38, and .60, respectively.

Figure 6A hints that the service provider welfare increases with the large task total data size value in both the proposed and compared schemes. The service provider's welfare is determined by taking the sum of revenue regarding the user's task execution, computation delay, communication delay, and waiting delay. It can be noted from the figure that the proposed cost-first scheme offers a greater service provider welfare value than the others. The results depict that the compared scheme secures the third position and the proposed time-first scheme secures the second position in terms of service provider welfare. This is because the resource purchase and maintenance costs (e.g., remote cloud) are lower in the proposed cost-first scheme than the others. From Figure 6A, when the task data size value is 252.9 Mb, the service provider welfare value for the proposed time-first, compared scheme (minimum communication delay), and proposed cost-first scheme is 0.83, 0.30, and 1.09, respectively.

Figure 6B examines the active and capable user device number *versus* the value of the simulation round for all three schemes. Figure 6B hints that the number of capable user devices reduces with the large simulation round value in all three schemes. The proposed time-first scheme gives the best results in terms of alive and capable user device numbers due to its lowest energy usage value during per-round task execution. The proposed cost-first scheme gives the lowest alive and capable user device number value due to its highest energy usage value during per-round task execution. The existing compared scheme (minimum communication delay) gives the second-best alive user device number due to its second-highest energy expense value per simulation round. From Figure 6B, when the task number is 26 and the simulation round is 1,700, the alive and capable user device values for the proposed time-first scheme, compared scheme (minimum communication delay), and proposed cost-first scheme are 66, 40, and 22, respectively.

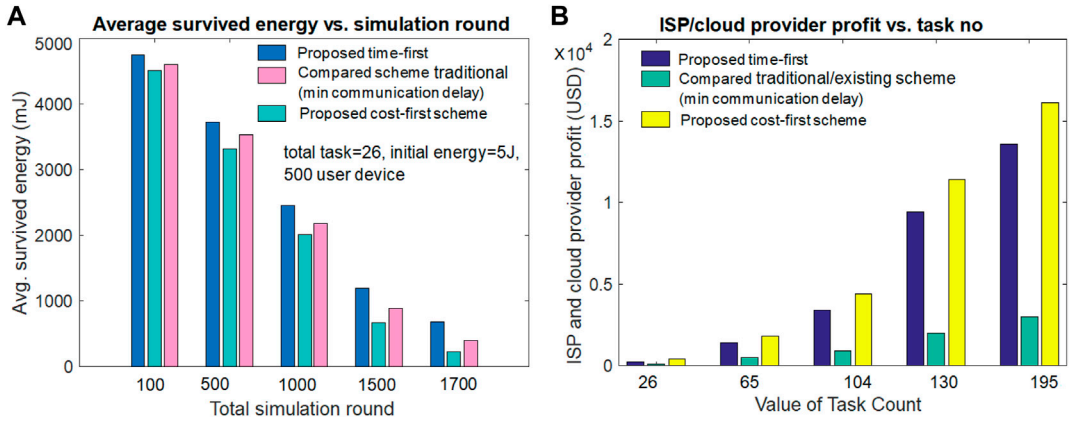

FIGURE 7
User survived energy and ISP/cloud service provider profit value.

TABLE 3 Comparative performance analysis for task number = 195.

Scheme name	Average task implementation delay (ms)	Users' service execution cost	Users' energy usage cost (mJ)
Proposed time-first accelerator scheme (with minimum predicted delay-based work node/resource selection)	33,658	31,548 USD	27,626
Proposed cost-first scheme (with minimum predicted monetary cost-based resource selection)	51,441	28,288 USD	30,200
Compared scheme 3: traditional scheme with minimum communication delay–based resource slicing (e.g., Sun et al., 2020; Siasi et al., 2020; Marotta et al., 2017; Cai et al., 2022; Przybylski et al., 2021)	42,164	33,350 USD	29,326
Compared scheme 4: traditional scheme with high computational power–based work node selection with casual task scheduling (e.g., Zhang et al., 2015; Ma et al., 2022; Demchenko et al., 2015; Angui et al., 2022)	57,845	35,096 USD	30,980

The average survival or remaining user device energy value performance *versus* the simulation round for both the proposed and existing schemes is given in Figure 7B. It can be seen from Figure 7A that the average survival or leftover user device energy decreases with the incremental value of the simulation round in all three schemes. Hence, due to the lower amount of user device energy per task execution, the proposed time-first scheme offers the highest average survived energy value among others. It can also be noticed from Figure 7A that due to the second-best and worst energy consumption during task implementation, the existing compared scheme and the proposed cost-first scheme give the second-best and lowest average survival energy value. From Figure 7A, it can be noted that when the simulation round is 1,500 and task number is 26, the average survived energy value for the proposed time-first scheme, compared scheme (minimum communication delay), and proposed cost-first scheme is 1,190 mJ, 935 mJ, and 785 MJ, respectively.

Figure 7B discusses the ISP/cloud service providers' profit result comparison by varying the task implementation value for both the proposed and compared schemes. Figure 7B depicts that the ISP/cloud provider profit increases with the incremental value of the task implementation number in all proposed and compared schemes. Due to lower resource costs, the proposed cost-first scheme provides the best ISP/cloud provider profit to others. The figure also shows that the proposed time-first scheme receives the second-best ISP/cloud provider profit due to its second-best task service execution cost. Although the compared scheme (minimum communication delay) gives the lowest ISP/cloud provider profit due to its highest task service execution cost and waiting delay for service. From Figure 7B, it can be pointed out that when the implemented task number is 195, the ISP/cloud provider profit value for the proposed time-first scheme, compared scheme (minimum communication delay), and proposed cost-first scheme is 13,600 USD, 3010 USD, and 16,100 USD, respectively.

5.1 Detailed comparison and performance gain analysis

Table 3 gives a comparative analysis (when the task number is 195) by taking three performance metrics results for the proposed time-first scheme, the proposed cost-first scheme, the traditional scheme with minimum communication delay-based resource selection (compared scheme 3), and the traditional scheme with high computational power-based resource selection

(compared scheme 4). It can be seen from the table that the proposed time-first scheme offers the best possible average task implementation delay and users' energy cost results. Although the proposed cost-first scheme shows the best possible user service execution cost results, the proposed cost-first scheme secures the third position in terms of average task implementation delay and users' energy usage cost. The traditional scheme with minimum communication delay-based resource selection (compared scheme 3) secures the second position, whereas the traditional scheme with high computational power-based resource selection (compared scheme 4) achieves the fourth position among all compared schemes in terms of average task implementation delay and user service execution monetary cost results. The reason behind the supremacy of the proposed time-first scheme is that it selects the work node or resources based on the lowest predicted delay basis, which includes the associated computation delay, communication delay, and waiting delay. Although the proposed cost-first scheme selects the resource with the lowest cost for different task executions, the compared scheme 3 selects suitable resources by examining the lowest possible communication delay. The compared scheme 4 achieves worse results due to its random resource selection nature without examining different delays and costs for each task execution. The compared scheme selects work node based on high computational power. From Table 3, it is seen that when the implemented task number is 195, the average task implementation delay gain in the proposed time-first scheme over compared scheme 3 (minimum communication delay), proposed cost-first scheme, and compared scheme 4 (high computational power) are 25.27%, 52.83%, and 71.8%, respectively. Table 3 also reveals that when the implemented task number is 195, the user service execution cost gain in the proposed cost-first scheme over compared scheme 3 (minimum communication delay), proposed time-first scheme, and compared scheme 4 (high computational power) are 17.89%, 11.52%, and 24.06%, respectively. Table 3 also shows that when the implemented task number is 195, the user energy usage cost gain in the proposed time-first scheme over compared scheme 3(minimum communication delay), proposed cost-first scheme, and compared scheme 4 (high computational power) are 6.15%, 9.31%, and 12.14%, respectively.

5.2 Computational complexity analysis

The computational complexity value of the proposed accelerator (both time-first and cost-first schemes) can be determined by $O(y * \phi_{rn} + y * \phi_{br})$, where y is the total value of the 6G and non-6G application request amounts. ϕ_{rn} is the work node (physical and virtual resource) selection number per application request. ϕ_{br} is the number of communication link resource selections per application execution. Thus, the computational complexity of the proposed accelerator (both time-first and cost-first schemes) is $O(y * \phi_{rn} + y * \phi_{br})$ because the proposed accelerator scheme selects the best resources per application with minimum predicted task implementation delay cost and minimum monetary cost value. The best possible work node resources are selected by examining all available resource node statuses (i.e., $O(y*\phi_{rn})$). The best possible communication link is selected for each application data transfer activity by examining all communication link statuses (i.e., $O(y * \phi_{br})$). Although the computational complexity of compared scheme 3 (minimum communication delay) or compared scheme 4 (maximum computational power) is $O(y * 1)$, for traditional scheme 3, the first nearby resource node is selected for each application execution. Similarly, for the traditional scheme 4, the work node with maximum power is selected for task execution. Thus, $O(1)$ time is required for resource selection per application request in the traditional scheme 3 or 4. Hence, the proposed accelerator scheme requires more computational time complexity than the traditional scheme for best resource selection.

5.3 Feasibility of the practical implementation of the work

In Section 3.1, we discussed the network model, such as considerations and technical standards. We utilized currently available devices and IEEE standards. Thus, the proposed model is practically feasible. Algorithm 1 and Figure 2, as well as Section 3.2, describe how we implemented our work. Section 4 illustrates the mathematical model used for evaluation. In Section 5, we presented and discussed the simulation results, which included their advantages and disadvantages. Table 2 and Section 5 provide detailed information on the simulation parameters. The simulation results clearly show that the proposed system outperforms existing systems in terms of task implementation delay, energy consumption cost, service execution cost for users, quality guarantee ratio, throughput, and service provider welfare outcomes. Thus, the proposed system is both legally, technically, performance-wise, and economically feasible for practical implementation.

6 Conclusion

This work introduces a task execution time priority-first and monetary cost priority-first policy based on time slot scheduling, virtual and physical workers, and bandwidth resource assignment algorithm for different 6G and non-6G application execution over ZTNs. To speed up the 6G and non-6G application execution over the ZTN, the proposed network model integrates different technologies (e.g., SDN, NFV, blockchain, digital twin, and MEC), different types of communication links (e.g., wired and wireless links), and different user devices (e.g., IoT devices and robots). To examine the proposed scheme's performance over a ZTN, this paper gives a performance analysis model that includes task implementation delay, energy cost, QoS guarantee ratio, and monetary cost metrics. It provides an accelerator-based task coordination and resource scheduling algorithm. The simulation results highlight that when the task number is 104, the average task implementation delay of the proposed time-first scheme, compared scheme 3 (minimum communication delay based), and the proposed cost-first resource selection policy are 12,455 ms, 20,202 ms, and 22,750 ms, respectively. For the data size value of 252.9 MB with 130 task number executions, the

user's energy usage value of the proposed time-first scheme, compared scheme 3 (minimum communication delay based), and the proposed cost-first resource selection policy are 18,329 mJ, 19,930 mJ, and 20,430 mJ, respectively. For the task number of 130, the service execution monetary cost value of the proposed time-first scheme, compared scheme 3 (minimum communication delay based), and proposed cost-first resource selection policies are 21,752 USD, 23,354 USD, and 14,990 USD, respectively. The evaluation results highlight that the proposed time-first scheme can offer maximum task implementation delay gain compared to other compared schemes. The simulation results also revealed that the proposed cost-first scheme can provide maximum service execution monetary cost gain compared to other compared schemes.

This work's future research extensions include deep learning–based ZTN failure prediction, service request arrival prediction, quantum cryptography–based security enhancement, and machine learning–based congestion control for SDN- and NFV-enabled ZTN. The work's limitation is that it did not investigate failure recovery, age-of-information-aware resource selection, or cost-effective VNF placement problems for ZTN-based 6G and non-6G application execution using DRL techniques. Furthermore, it did not look into blockchain and FL-based security and privacy checks for ZTN-based application execution. A semantic communication-aware resource-slicing framework can be developed in the future by taking into account more emerging next-generation application scenarios (e.g., industry 5.0), dynamic network scenarios, different attacks and trusted collaboration node selection, game theory–based resource sharing policy, and heterogeneous requirements satisfaction (e.g., load balancing and reliability guarantee) for ZTNs.

Data availability statement

The original contributions presented in the study are included in the article/supplementary material; further inquiries can be directed to the corresponding author.

Author contributions

MC: writing—original draft and writing—review and editing.

Funding

The author declares that no financial support was received for the research, authorship, and/or publication of this article.

Conflict of interest

The author declares that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Publisher's note

References

Abbas, K., Afaq, M., Ahmed Khan, T., Rafiq, A., and Song, W. C. (2020). Slicing the core network and radio access network domains through intent-based networking for 5G networks. *Electronics* 9 (10), 1710–1715. id 1710. doi:10.3390/electronics9101710

Adhikari, M., Munusamy, A., Kumar, N., and Srirama, S. (2022). Cybertwin-driven resource provisioning for IoE applications at 6G-enabled edge networks. *IEEE Trans. Industrial Inf.* 18 (7), 4850–4858. doi:10.1109/tii.2021.3096672

Alsabah, M., Naser, M. A., Mahmmod, B. M., Abdulhussain, S. H., Eissa, M. R., Al-Baidhani, A., et al. (2021). 6G wireless communications networks: a comprehensive survey. *IEEE Access* 9, 148191–148243. doi:10.1109/access.2021.3124812

Alwis, C. D., Kalla, A., Pham, Q. V., Kumar, P., Dev, K., Hwang, W. J., et al. (2021). Survey on 6G Frontiers: trends, applications, requirements, technologies and future research. *IEEE Open J. Commun. Soc.* 2, 836–886. doi:10.1109/ojcoms.2021.3071496

Amreen, Srinivas, P., Rao, N. T., Bhattacharyya, D., and Kim, H. j. (2017). Performance evaluation in cloud computing model using queuing models. *Int. J. Grid Distributed Comput.* 10 (3), 15–24. doi:10.14257/ijgdc.2017.10.3.02

Angui, B., Corbel, R., Rodriguez, V. Q., and Stephan, E. (2022). "Towards 6G zero touch networks: the case of automated Cloud-RAN deployments," in *IEEE 19th annual consumer communications & networking conference (CCNC)*, 1–6.

Ashraf, I., Zikria, Y. B., Garg, S., Park, Y., Kaddoum, G., and Singh, S. (2022). Zero touch networks to realize virtualization: opportunities, challenges, and future prospects. *IEEE Netw.* 36 (6), 251–259. doi:10.1109/mnet.001.2200029

Basu, D., Kal, S., Ghosh, U., and Datta, R. (2022). SoftChain: dynamic resource management and SFC provisioning for 5G using machine learning. *IEEE Globecom Work. (GC Wkshps)*, 280–285. doi:10.1109/gcwkshps56602.2022.10008691

Boškov, I., Yetgin, H., Vucnik, M., Fortuna, C., and Mohorcic, M. (2020). Time-to-Provision evaluation of IoT devices using automated zero-touch provisioning. *IEEE Glob. Commun. Conf.*, 1–7. doi:10.1109/globecom42002.2020.9348119

Brik, B., and Ksentini, A. (2020). "On predicting service-oriented network slices performances in 5G: a federated learning approach," in IEEE 45th Conference on Local Computer Networks (LCN), Sydney, NSW, 164–171. doi:10.1109/LCN48667.2020.9314849

Cai, Y., Llorca, J., Tulino, A. M., and Molisch, A. F. (2022). "Compute- and data-intensive networks: the key to the Metaverse," in *1st international conference on 6G networking (6GNet)*, 1–8.

Cao, H., Du, J., Zhao, H., Luo, D. X., Kumar, N., Yang, L., et al. (2021). Resource-ability assisted service function chain embedding and scheduling for 6G networks with virtualization. *IEEE Trans. Veh. Technol.* 70 (4), 3846–3859. doi:10.1109/tvt.2021.3065967

Chen, J., Cheng, X., and Zhang, H. (2022). A lightweight SFC embedding framework in SDN/NFV-enabled wireless network based on reinforcement learning. *IEEE Syst. J.* 16 (3), 3817–3828. doi:10.1109/jsyst.2021.3111972

Chen, J., Deng, Q., and Yang, X. (2023). Non-cooperative game algorithms for computation offloading in mobile edge computing environments. *J. Parallel Distributed Comput.* 172 (xx), 18–31. doi:10.1016/j.jpdc.2022.10.004

Chen, J., Li, K., Deng, Q., Li, K., and Yu, P. S. (2019). Distributed deep learning model for intelligent video surveillance systems with edge computing. *IEEE Trans. Industrial Inf.*, 1–8. doi:10.1109/tii.2019.2909473

Chen, W., Wang, Z., Zhang, H., Yin, X., and Shi, X. (2021). "Cost-efficient dynamic service function chain embedding in edge clouds," in 17th International Conference on Network and Service Management (CNSM), Izmir, Turkey, 310–318. doi:10.23919/CNSM52442.2021.9615590

Chergui, H., Blanco, L., Garrido, L. A., Ramantas, K., Kuklinski, S., Ksentini, A., et al. (2021). Zero-touch AI-driven distributed management for energy-efficient 6G massive network slicing. *IEEE Netw.* 35 (6), 43–49. doi:10.1109/mnet.111.2100322

Chowdhury, M. (2022). An energy harvesting, blockchain, and QoS-aware intelligent healthcare task coordination policy for IoT-assisted networks. *Int. J. Embed. Syst.* 15 (4), 313–325. doi:10.1504/ijes.2022.10050472

Collet, A., Banchs, A., and Fiore, M. (2022). "LossLeaP: learning to predict for intent-based networking," in IEEE INFOCOM 2022 - IEEE Conference on Computer Communications, London, United Kingdom, 2138–2147. doi:10.1109/INFOCOM48880.2022.9796918

Coronado, E., Behravesh, R., Subramanya, T., Fernandez-Fernandez, A., Siddiqui, M. S., Costa-Perez, X., et al. (2022). Zero touch management: a survey of network automation solutions for 5G and 6G networks. *IEEE Commun. Surv. Tutorials* 24 (4), 2535–2578. doi:10.1109/comst.2022.3212586

Coronado, E., Behravesh, R., Subramanya, T., Fernandez-Fernandez, A., Siddiqui, M. S., Costa-Perez, X., et al. (2022). Zero touch management: a survey of network automation solutions for 5G and 6G networks. *IEEE Commun. Surv. Tutorials* 24 (4), 2535–2578. doi:10.1109/comst.2022.3212586

Dalgkitsis, A., Mekikis, P. -V., Antonopoulos, A., Kormentzas, G., and Verikoukis, C. (2020). "Dynamic resource aware VNF placement with deep reinforcement learning for 5G networks," in GLOBECOM 2020 - 2020 IEEE Global Communications Conference, Taipei, Taiwan, 1–6. doi:10.1109/GLOBECOM42002.2020.9322512

Demchenko, Y., Filiposka, S., Tuminauskas, R., Mishev, A., Baumann, K., Regvart, D., et al. (2015). "Enabling automated network services provisioning for cloud based applications using zero touch provisioning," in IEEE/ACM 8th international conference on utility and cloud computing (UCC), Limassol, Cyprus, 458–464. doi:10.1109/UCC.2015.82

Drolia, U., Guo, K., Tan, J., Gandhi, R., and Narasimhan, P. (2017). Cachier: edge-caching for recognition applications. *IEEE ICDCS*, 276–286. doi:10.1109/icdcs.2017.94

El Houda, Z. A., Brik, B., and Khoukhi, L. (2022). Ensemble learning for intrusion detection in SDN-based zero touch smart grid systems. *IEEE 47th Conf. Local Comput. Netw. (LCN)* 2022, 149–156. doi:10.1109/lcn53696.2022.9843645

Fathalla, A., Li, K., and Salah, A. (2022). Best-KFF: a multi-objective preemptive resource allocation policy for cloud computing systems. *Clust. Comput.* 25 (1), 321–336. doi:10.1007/s10586-021-03407-z

Feng, L., Zi, Y., Li, W., Zhou, F., Yu, P., and Kadoch, M. (2020). Dynamic resource allocation with RAN slicing and scheduling for uRLLC and eMBB hybrid services. *IEEE Access* 8, 34538–34551. doi:10.1109/access.2020.2974812

Grasso, C., Raftopoulos, R., and Schembra, G. (2021). Smart zero-touch management of UAV-based edge network. *IEEE Trans. Netw. Serv. Manag.* 19 (4), 4350–4368. doi:10.1109/tnsm.2022.3160858

Gu, Y., Hu, Y., Ding, Y., Lu, J., and Xie, J. (2019). Elastic virtual network function orchestration policy based on workload prediction. *IEEE Access* 7, 96868–96878. doi:10.1109/access.2019.2929260

Hantouti, H., Benamar, N., and Taleb, T. (2020). Service function chaining in 5G & beyond networks: challenges and open research issues. *IEEE Netw.* 34 (4), 320–327. doi:10.1109/mnet.001.1900554

Hermosilla, A., Zarca, A. M., Bernabe, J. B., Ortiz, J., and Skarmeta, A. (2020). Security orchestration and enforcement in NFV/SDN-Aware UAV deployments. *IEEE Access* 8, 131779–131795. doi:10.1109/access.2020.3010209

Hu, J., Li, K., Liu, C., Chen, J., and Li, K. (2021). Coalition formation for deadline-constrained resource procurement in cloud computing. *J. Parallel Distributed Comput.* 149 (xx), 1–12. doi:10.1016/j.jpdc.2020.10.004

Huang, L., Pan, Y., Yang, J., Shen, D., Chen, S., and Huang, L. (2023) A hybrid meta-heuristic algorithm with fuzzy clustering method for IoT smart electronic applications. *Int. J. Embed. Syst.* 16:(01), 57–66. doi:10.1504/ijes.2023.10059681

Jalalitabar, M., Wang, Y., and Cao, X. (2019). "Branching-aware service function placement and routing in network function virtualization," in IEEE conference on network function virtualization and software defined networks (NFV-sdn), Dallas, TX, 1–6. doi:10.1109/NFV-SDN47374.2019.9039981

Ksentini, A. (2021). Tutorial: zero touch management and orchestration of network slices in 5G and beyond networks. *IEEE Int. Conf. Commun.* 2021, 1–6.

Lin, R., He, L., Luo, S., and Zukerman, M. (2023). Energy-aware service function chaining embedding in NFV networks. *IEEE Trans. Serv. Comput.* 16 (2), 1158–1171. doi:10.1109/tsc.2022.3162328

Lin, R., Luo, S., and Zukerman, M. (2022). Service function chaining embedding in hybrid optical-electronic networks. *J. Light. Technol.* 40 (15), 4922–4933. doi:10.1109/jlt.2022.3176473

Lin, S.-C., Lin, C.-H., and Chen, W.-C. (2022). Zero-touch network on industrial IoT: an end-to-end machine learning approach. *IEEE Netw.*, 1–8. Submitted for publication in the journal.

Liyanage, M., Pham, Q. V., Dev, K., Bhattacharya, S., Maddikunta, P. K. R., Gadekallu, T. R., et al. (2022). A survey on Zero touch network and Service Management (ZSM) for 5G and beyond networks. *J. Netw. Comput. Appl.* 203 (103362), 103362–103427. doi:10.1016/j.jnca.2022.103362

Luque-Schempp, F., Panizo, L., Gallardo, M. d. M., Merino, P., and Rivas, J. (2022). Toward zero touch configuration of 5G non-public networks for time sensitive networking. *IEEE Netw.* 36 (2), 50–56. doi:10.1109/mnet.006.2100442

Ma, J., Guo, Y., Fang, C., and Zhang, Q. (2022). Digital twin-based zero-touch management for IoT. *Electronics* 11 (4104), 4104. doi:10.3390/electronics11244104

Marotta, A., Zola, E., D'Andreagiovanni, F., and Kassler, A. (2017). A fast robust optimization-based heuristic for the deployment of green virtual network functions. *J. Netw. Comput. Appl.* 95, 42–53. doi:10.1016/j.jnca.2017.07.014

Martini, B., Gharbaoui, M., and Castoldi, P. (2022). Intent-based zero-touch service chaining layer for software-defined edge cloud networks. *Comput. Netw.* 212 (109034), 109034–109115. doi:10.1016/j.comnet.2022.109034

Mohamad, A., and Hassanein, H. S. (2022). Prediction-based SFC placement with VNF sharing at the edge. *IEEE 47th LCN*, 26–33. doi:10.1109/lcn53696.2022.9843704

Mohammadpour, A., Lombardo, C., Bolla, R., Bruschi, R., Davoli, F., and Ivaldi, L. (2022). "A zero-touch as-a-Service active monitoring framework for virtualized network environments," in IEEE 8th international Conference on network softwarization (NetSoft), Milan, Italy, 103–108. doi:10.1109/NetSoft54395.2022.9844069

Multiple Authors, (2022a). Number connected-iot devices Available at: https://iot-analytics.com/number-connected-iot-devices/," acessed 01 december 2022.

Multiple Authors (2022b). *Zero touch networks opportunities challenges and potential.* Available at: https://www.comsoc.org/publications/magazines/ieee-network/cfp/zero-touch-networks-opportunities-challenges-and-potential.

Nibouchaet, R., Saad, S. B., Ksentini, A., and Challal, Y. (2023). Zero-touch security management for mMTC network slices: DDoS attack detection and mitigation. *IEEE Internet Things J.* 10 (9), 7800–7812. doi:10.1109/jiot.2022.3230875

Okwuibe, J., Haavisto, J., Kovacevic, I., Harjula, E., Ahmad, I., Islam, J., et al. (2021). SDN-enabled resource orchestration for industrial IoT in collaborative edge-cloud networks. *IEEE Access* 9, 115839–115854. doi:10.1109/access.2021.3105944

Pei, J., Hong, P., Xue, K., Li, D., Wei, D. S. L., and Wu, F. (2020). Two-phase virtual network function selection and chaining algorithm based on deep learning in SDN/NFV-enabled networks. *IEEE J. Sel. Areas Commun.* 38 (6), 1102–1117. doi:10.1109/jsac.2020.2986592

Przybylski, B., Zuk, P., and Rzadca, K. (2021). "Data-driven scheduling in serverless computing to reduce response time," in 2021 IEEE/ACM 21st International Symposium on Cluster, Cloud and Internet Computing (CCGrid), Melbourne, Australia, 206–216.

Red Hat (2024). What is NFV? Available at: https://www.redhat.com/en/topics/virtualization/what-is-nfv.

Rico-Palomo, J. J., Galeano-Brajones, J., Cortes-Polo, D., Valenzuela-Valdes, J. F., and Carmona-Murillo, J. (2022). Chained orchestrator algorithm for RAN-slicing resource management: a contribution to ultra-reliable 6G communications. *IEEE Access* 10, 113662–113677. doi:10.1109/access.2022.3218061

Roy, S., Chergui, H., Sanabria-Russo, L., and Verikoukis, C. (2022). A cloud native SLA-driven stochastic federated learning policy for 6G zero-touch network slicing. *IEEE Int. Conf. Commun.* 2022, 4269–4274. doi:10.1109/icc45855.2022.9838376

Saha, D., Shojaee, M., Baddeley, M., and Haque, I. (2020). "An energy-aware SDN/NFV architecture for the internet of things," in IFIP Networking Conference (Networking), Paris, France 2020, 604–608.

Saha, M., Panda, S. K., and Panigrahi, S. (2023). A modified Brown and Gibson model for cloud service selection. *Int. J. Comput. Sci. Eng.* 26 (4), 430–444. doi:10.1504/ijcse.2023.132150

Salameh, A. I., and El Tarhuni, M. (2022). From 5G to 6G—challenges, technologies, and applications. *Future Internet* 14 (117), 117–135. doi:10.3390/fi14040117

Sebrechts, M., Volckaert, B., De Turck, F., Yang, K., and Al-Naday, M. (2022). Fog native architecture: intent-based workflows to take cloud native toward the edge. *IEEE Commun. Mag.* 60 (8), 44–50. doi:10.1109/mcom.003.2101075

Shaghaghi, A., Mokari, N., Javan, M. R., Behdadfar, M., and Jorswieck, E. A. (2021). Proactive and AoI-aware failure recovery for stateful NFV-enabled zero-touch 6G networks: model-free DRL approach. *IEEE TNSM*, 1–14.

Siasi, N., Jasim, M., Aldalbahi, A., and Ghani, N. (2020). Delay-aware SFC provisioning in hybrid fog-cloud computing architectures. *IEEE Access* 8, 167383–167396. doi:10.1109/access.2020.3021354

Song, F., Li, J., Ma, C., Zhang, Y., Shi, L., and Jayakody, D. N. K. (2020). Dynamic virtual resource allocation for 5G and beyond network slicing. *IEEE Open J. Veh. Technol.* 1, 215–226. doi:10.1109/ojvt.2020.2990072

Sousa, N. F. S. d., and Rothenberg, C. E. (2021). "CLARA: closed loop-based zero-touch network management framework," in IEEE conference on network function virtualization and software defined networks (NFV-sdn), Heraklion, Greece, 110–115. doi:10.1109/NFV-SDN53031.2021.9665048

Suh, K., Kim, S., Ahn, Y., Kim, S., Ju, H., and Shim, B. (2022). Deep reinforcement learning-based network slicing for beyond 5G. *IEEE Access* 10, 7384–7395. doi:10.1109/access.2022.3141789

Sun, G., Xu, Z., Yu, H., Chen, X., Chang, V., and Vasilakos, A. V. (2020). Low-latency and resource-efficient service function chaining orchestration in network function virtualization. *IEEE IoT J.* 7 (7), 5760–5772. doi:10.1109/jiot.2019.2937110

Tamim, I., Jammal, M., Hawilo, H., and Shami, A. (2020). Introducing virtual security functions into latency-aware placement for NFV applications. *IEEE ICC, Dublin, Irel.*, 1–7. doi:10.1109/icc40277.2020.9149288

Tang, J., Duan, Y., Zhou, Y., and Jin, J. (2021). Distributed slice selection-based computation offloading for intelligent vehicular networks. *IEEE Open J. Veh. Technol.* 2, 261–271. doi:10.1109/ojvt.2021.3087355

Theodorou, V., Lekidis, A., Bozios, T., Meth, K., Fernández-Fernández, A., Tavlor, J., et al. (2021). "Blockchain-based zero touch service assurance in cross-domain network slicing," in Joint European Conference on Networks and Communications and 6G Summit (EuCNC/6G Summit), Porto, Portugal, 395–400. doi:10.1109/EuCNC/6GSummit51104.2021.9482602

Thiruvasagam, P. K., Chakraborty, A., and Murthy, C. S. R. (2021). Resilient and latency-aware orchestration of network slices using multi-connectivity in MEC-enabled 5G networks. *IEEE Trans. Netw. Serv. Manag.* 18 (3), 2502–2514. doi:10.1109/tnsm.2021.3091053

Tianran, D., Tian, G., Wei, J., and Liu, S. (2023). Blockchain-based collaborative intrusion detection scheme. *Int. J. Embed. Syst.* 26 (4), 418–429. doi:10.1504/ijcse.2023.132147

Tseng, H.-W., Yang, T. T., and Hsu, F. T. (2021). An MEC-based VNF placement and scheduling scheme for AR application topology. *IEEE WCNC*, 1–6. doi:10.1109/wcnc49053.2021.9417126

VMware (2024). What is software-defined networking? Available at: https://www.vmware.com/topics/glossary/content/.

Wang, P., Li, K., Xiao, B., and Li, K. (2022). Multiobjective optimization for joint task offloading, power assignment, and resource allocation in mobile edge computing. *IEEE IoT J.* 9 (14), 11737–11748. doi:10.1109/jiot.2021.3132080

Wang, Y., and Farooq, J. (2022). Zero touch coordinated UAV network formation for 360 degree views of a moving ground target in remote VR applications. *IEEE Mil. Commun. Conf. (MILCOM)* 2022, 950–955.

Wang, Y., Huang, C. K., Shen, S. H., and Chiu, G. M. (2021). Adaptive placement and routing for service function chains with service deadlines. *IEEE Trans. Netw. Serv. Manag.* 18 (3), 3021–3036. doi:10.1109/tnsm.2021.3086977

Wei, S., Zhou, J., and Chen, S. (2022). Delay-aware multipath parallel SFC orchestration. *IEEE Access* 10, 120035–120055. doi:10.1109/access.2022.3221744

Xu, Y.-H., Hua, M., Zhou, W., and Yu, G. (2022). Resource allocation for cellular zero-touch deterministic industrial M2M networks: a reinforcement learning-based scheme. *IEEE Sensors Lett.* 6 (8), 1–4. doi:10.1109/lsens.2022.3194141

Yoshino, M., Astawa, G., Trinh, T., Suzuki, H., Koswara, M., and Nguyen, B. (2021). Zero-touch multi-service provisioning with pluggable module-type OLT in access network virtualization testbed. *GLOBECOM 2020 - 2020 IEEE Glob. Commun. Conf.* 2020, 148800–148809. doi:10.1109/access.2021.3110249

Zahoor, S., Ahmad, I., Othman, M. T. B., Mamoon, A., Rehman, A. U., Shafiq, M., et al. (2022). Comprehensive analysis of network slicing for the developing commercial needs and networking challenges. *MDPI Sensors* 22 (17), 6623–6721. doi:10.3390/s22176623

Zahoor, S., Ahmad, I., Ur Rehman, A., Tag Eldin, E., A. Ghamry, N., and Shafiq, M. (2023). Performance evaluation of virtualization methodologies to facilitate NFV deployment. *Comput. Mater. Continua* 75 (1), 311–329. doi:10.32604/cmc.2023.035960

Zhang, D., Chen, X., Huang, Q., Hong, X., Wu, C., Zhou, H., et al. (2019). P4SC: a high performance and flexible framework for service function chain. *IEEE Access* 7, 160982–160997. doi:10.1109/access.2019.2950446

Zhang, W., Wen, Y., and Wu, D. O. (2015). Collaborative task execution in mobile cloud computing under a stochastic wireless channel. *IEEE Trans. Wirel. Commun.* 14 (1), 81–93. doi:10.1109/twc.2014.2331051

Zhong, X., Wang, Y., and Qiu, X. (2019). "Cost-aware service function chaining with reliability guarantees in NFV-enabled inter-DC network," in IFIP/IEEE Symposium on Integrated Network and Service Management (IM), Arlington, VA 2019, 304–311.

Zhou, S., Wang, G., Zhang, S., Niu, Z., and Shen, X. S. (2019). Bidirectional mission offloading for agile space-air-ground integrated networks. *IEEE Wirel. Commun.* 26 (2), 38–45. doi:10.1109/mwc.2019.1800290