INDERSCIENCE PUBLISHERS
Linking academia, business and industry through research

International Journal of Sensor Networks

ISSN online: 1748-1287 - ISSN print: 1748-1279
https://www.inderscience.com/ijsnet

Superactive: a priority, latency, and SLA-aware resource management scheme for software defined space-air-ground integrated networks

Mahfuzulhoq Chowdhury

DOI: 10.1504/IJSNET.2022.10050822

Article History:
Received: 01 May 2022
Accepted: 29 August 2022
Published online: 24 January 2023

Copyright © 2023 Inderscience Enterprises Ltd.

Superactive: a priority, latency, and SLA-aware resource management scheme for software defined space-air-ground integrated networks

Mahfuzulhoq Chowdhury

Computer Science and Engineering Department,
Chittagong University of Engineering and Technology,
Chittagong – 4349, Bangladesh
Email: mahfuzulhoq.cse05@gmail.com

Abstract: The software-defined space-air-ground integrated network (SAGIN) is regarded as future-generation networking solution due to its wide-area coverage and seamless communication support for ground networks and resource-intensive application support for space-air networks. The SDN-based literary works are restricted only to resource management for space-air or ground networks, not both. The resource scheduling for multi-users caching, computing and data-transfer task accomplishment over software-defined SAGIN were out of their investigations by taking proactive local SDN, heterogenous users and applications, SLA requirements, priority, latency, and users' budget into account. To outperform the problems, this paper advocates a proactive SDN-based resource management scheme considering SLA requirements, latency, priority, and users' budget for multi-user task completion over software-defined SAGIN. This paper contributes an analytical model that covers task completion time, energy expenditure, financial cost, and SLA fulfilment metrics. The performance results illustrate that proposed scheme produces 72% time and 69% financial gain over the compared scheme.

Keywords: superactive resource management; software-defined networking; SDN; space-air-ground integrated networks; SAGIN; task completion time; profit; throughput.

Reference to this paper should be made as follows: Chowdhury, M. (2023) 'Superactive: a priority, latency, and SLA-aware resource management scheme for software defined space-air-ground integrated networks', *Int. J. Sensor Networks*, Vol. 41, No. 1, pp.23–41.

Biographical notes: Mahfuzulhoq Chowdhury received his PhD (Doctoral) in Telecommunications at the University of Quebec, INRS, Montreal, Canada in 2019. He received his Master's and Bachelor's in Computer Science and Engineering (CSE), in 2010 and 2015, respectively. He is an faculty member at the Department of Computer Science and Engineering, Chittagong University of Engineering and Technology from september 2010. His research interests are related to resource allocation in wireless networks, software-defined networking (SDN), network function virtualisation, space-air-ground integrated networks (SAGIN), optimisation, cloud computing, future generation networks, machine learning, deep learning, artificial intelligence, blockchain, network security, IoT, UAV, VANET, mobile application and high speed communication, among others. He has published several research papers at different IEEE journals, transactions, IEEE magazines, and IEEE conferences as well as chapters of books.

1 Introduction

With the expeditious improvement of the ground network and satellite technology, high and low altitude platforms, gateways, unmanned aerial vehicle type devices or UAVs, terrestrial network devices, and communication technologies, the space-air-ground integrated network (SAGIN) comes as a suitable solution for many real-time applications (i.e., sensing, data transmission, processing, navigation) by offering not only the wide range of communication services but also both seamless and low-latency service requirement satisfaction facilities (Zhou et al., 2019). The space-air network can complement the ground network by providing wide-area coverage (i.e., communication facilities for complex terrains/places such as mountains and sea), communication opportunities and infrastructure during emergency scenarios and natural disasters (i.e., earthquakes), and by lowering communication delay during mission-critical (i.e., military) applications (Bi et al., 2019; Cui et al., 2022). Whereas, the ground network can benefit the space-air network (i.e., sustainability, power saving, and capability) by

Copyright © 2023 Inderscience Enterprises Ltd.

minimising their spectrum scarcity issue and by assisting their resource-intensive (i.e., high computation load, storage) applications execution (Zhang et al., 2017). For example, the computation workload of a space-air network task such as collaborative monitoring/sensing/navigation can be offloaded to the powerful cloud device or computing server (i.e., central or edge-cloud) on the ground network for processing and result generation.

At present, the fruitful execution of SAGIN real-time application suffers from a wide range of challenges. The question of how to coordinate the resources from ground networks [i.e., mobile users' (MUs) devices, network devices, and cloud server resources] and space-air networks (i.e., satellites and UAV) is one of the noticeable challenges (Yang et al., 2021). An additional fundamental research question is how to assign the task of several users (i.e., MUs, robots, sensors, and vehicles) of SAGIN to suitable actors or devices for execution by taking into account heterogeneous devices capability, terrestrial network technologies (i.e., cellular and Wi-Fi) and cloud computing resource availability, among others.

To this end, due to its flexibility, control, and programmability nature, the software-defined networking (SDN) technology has emerged as a key technology to resolve resource coordination problems for different communication paradigms related to real-time application executions under different networks. Currently, numerous research challenges exist in the literary works that considered SDN technology for resource and network task coordination. Zhang et al. (2020b) incorporated SDN technology to decide whether the vehicle's task is offloaded or transferred to a cloud actor device (i.e., mobile-edge cloud server or remote server) or another powerful vehicle for processing in a MEC-empowered vehicular network. Nazari et al. (2016) utilised SDN technology for multiple satellite resource sharing and load balancing purposes in naval networks. In Sheng et al. (2017), the SDN technology is used to assign multiple resources for different earth observation missions as well as network device coordination operations. In Li et al. (2020b), SDN assisted secured authentication scheme is proposed for IoT-based healthcare applications.

To address the challenges of a large number of users/mobile devices, applications, and data in the 5G era, in Tadros et al. (2020) the network services are managed by the SDN technology. However, this work is limited only to ground network and MUs rather than management of both SAGIN and heterogenous users (i.e., robots and vehicles) at the same time. To minimise latency for vehicles moving from one network to another, Abugabah et al. (2020) proposed SDN based intelligent traffic engineering facility with a routing path selection scheme for vehicular networks. To predict and balance the traffic load of IoT servers, Montazerolghaem et al. (2020) implemented a heuristic scheme in the SDN controller which is based on fuzzy logic and time-series data analysis. To automate the network services, in Nyanteh et al. (2021) the SDN-based controller is utilised to coordinate both physical and virtual network infrastructure. However, the investigations on different types of users and applications service level agreement (SLA) satisfaction for SAGINs were out of their consideration. In Nkenyereye et al. (2021), the SDN controller selects the roadside unit (RSU) coordinator for vehicle safety message dissemination along with vehicle road network discovery, cloud computing node selection, and cluster formation activities, among others. Zhu et al. (2017) used spectrum sharing and interference management mechanisms for integrated terrestrial-satellite networks. However, they did not investigate the role of SDN technology in task allocation for heterogeneous users. To manage the workload of cloudlet servers and control traffic overhead, Nithya et al. (2020) proposed a cyber foraging framework for resource coordination based on SDN technology. To optimise the network resource usage and routing path planning, Ling et al. (2019) formulated an SDN-based multi-objective optimisation problem. However, this work is limited only to satellite networks. To improve the energy usage of sensors, Chen et al. (2020) investigated an SDN-based resource scheduling scheme based on particle swarm optimisation. It is perceived that this work is limited only to sensor network users.

From the previously discussed research studies, it can be deduced that almost most of the SDN-based works investigated the resource scheduling scheme either for ground networks or satellite networks rather than SDN-based SAGIN. Further, the suitable resource-to-task mapping for different types of users (i.e., MUs, sensors, robots, and vehicles) was out of their analysis. The research question of how resources can be assigned for different types of SAGIN user's tasks (i.e., blockchain, computing, caching and data transmission task) and how to arrange the scheduling model for resources were out of their assessment by taking into account different SDN controller deployment model, different user types with their priority, latency and SLA requirements, pricing cost for services, service provider profit, different ground, and space-air networks nodes capability, nodes status collection, among others. Moreover, the existing work is mainly limited only to a centralised SDN (CSDN) controller. To avoid additional control signal overhead, proper SDN controller-based network coordination is imperative by investigating both central and decentralised SDN controller overhead. The existing literature works did not investigate any suitable network architecture for SDN-based SAGIN by considering sensor, robot, vehicle, and human MUs' device and their applications at the same time.

To resolve the existing issues, in this paper a decentralised SDN controller-based superactive resource scheduling scheme is introduced for SAGIN. The key features and contributions of this paper are noted below:

- This paper inaugurates a new SDN-based network architecture for integrated by taking into account different types of users including sensors, robots, vehicles, and human users' devices.

- This paper introduces a superactive resource scheduling and task assignment scheme for software-defined SAGIN applications by investigating their priority, latency, financial cost, and SLA requirements along with network devices capability and heterogeneous types of task (i.e., blockchain-based computing task, caching task and data transmission task). Our proposed scheme based on SDN technology also investigates proper SDN controller selection for different types of task and resource assignment planning, by taking into account their associated latencies and ability to provide the best possible solution.
- To determine the proposed superactive resource management schemes performance, this paper devises a flexible mathematical model that includes several performance metrics such as mean task completion time, mean SLA fulfilment ratio, service provider profit, time and financial reward ratio (FRR), mean network throughput, finance cost for users, energy expenditure cost, among others. The proposed mathematical model properly incorporates different types of costs and delays associated with systems resource allocation and task execution.
- To estimate the efficiency of the proposed superactive scheme for heterogeneous users' task execution in the SDN-based SAGIN, this paper contrasts the proposed scheme performance with the traditional hybrid reactive SDN (HRSDN) controller-based FCFS scheduling and traditional CSDN-controller-based scheduling.

Next, the key discussion associated with important literature works is depicted in Section 2. The design, modelling, and methodology discussion associated with the proposed superactive resource management scheme and software-defined SAGIN are explained in Section 3. The mathematical model associated with the proposed scheme is delineated in Section 4. The simulation results with reasoning are presented in Section 5. The findings of this research paper are outlined in the conclusion of Section 6.

2 Related works

This section details existing literature work on SDN-based heterogeneous networks with their respective limitations and merits. Nkenyereye et al. (2021) presented an efficient RSU selection scheme using a utility function at the SDN controller for fog-based vehicular networks along with a cluster head selection scheme. To address the routing path selection problem, Bano et al. (2021) come up with a robust routing architecture solution by integrating an SDN controller and wireless mesh network. Akin et al. (2019) examined the suitability of static and dynamic link costs for the existing routing schemes in SDN-assisted networks. However, to acquire the overall networking and routing-related information, the authors rely on the centralised controller. To offload the vehicle's computation task to the best available processor, Ali Shah et al. (2020) discussed an SDN-assisted cooperative computation offloading scheme by taking both remote and edge device/cloud into account. To offer intelligent path selection and other traffic engineering activities, Abugabah et al. (2020) examined the necessity of a multi-path routing scheme in SDN-enabled vehicular networks. To improve the routing performance, Younus et al. (2022) combined reinforcement learning techniques with SDN technology in wireless sensor networks. To balance the load on a different link and lower the packet loss, Chen et al. (2021) presented a dynamic routing path control and link selection strategy using both link preference load balancing in SDN. To improve the network's bandwidth utilisation and task completion success percentage, Chen et al. (2020) introduced a dynamic task mapping and resource scheduling technique in WSNs. In Alfoudi et al. (2019), an SDN-based resource assignment mechanism is presented for each isolated slice in LTE networks. However, the authors did not consider different types of users and decentralised controllers. Xia et al. (2020) select the computing node for IoV's delay-tolerant data transfer in the cloud-fog network by using a CSDN controller-based Markov decision (partially observable) process. Yuan et al. (2020b) studied a joint offloading and migration problem for SDN-based wireless body area networks (WBANs). To get the global view of cloud, fog servers, and local devices resource values, the authors used a central SDN controller for offloading the decision-making process. To improve the network management and QoS of users, Bi et al. (2019) investigated the benefits and challenges associated with integrated space and terrestrial networks with SDN technology. Liu et al. (2015) discussed a flexible network architecture for future generation PONs by integrating fibre-wireless convergence with SDN technology. However, the authors did not introduce any SDN-based resource management scheme for different types of users. To maximise social welfare by satisfying small cell users' service requirements, Sultana et al. (2019) discussed a spectrum auction algorithm for SDN-based C-RAN. The author presented a price charging and winner selection mechanism, which is inspired by the well-known VCG auction algorithm. To provide different slices associated with different 5G applications (i.e., smart grid and vehicular), Bektas et al. (2018) presented an end-to-end slicing solution. However, the authors did not provide any mathematical model or resource scheduling algorithm. To optimise the network resource usage and to offer flexible manufacturing data forwarding facility, Wan et al. (2020) instigated a cross-network resource scheduling and fusion scheme for CSDN controller-empowered heterogeneous smart factory networks. However, the authors did not consider different types of users and applications other than smart factories. The authors also

did not investigate the performance difference between centralised and decentralised SDN controllers. To optimise the random access for machine-to-machine communication in SDN-enabled cellular networks, Li et al. (2017) used a Markov decision (partially observable or POMDP) process. The authors considered only a single type of user rather than multiple types. Doan et al. (2020) optimised the MEC applications state transfer latency problem by using a meta-heuristic algorithm (i.e., Tabu search). The work is mainly suitable for moving vehicles/users applications rather than both static and movable users applications cost optimisation. Marotta (2019) developed a cooperative scheme by taking into account the adaptive collaboration and control of both software-defined mobile and access networks for the flexible functional split (i.e., distribution and reconfiguration), wavelength, and bandwidth allocation in the passive optical networks (i.e., PON front hauling). Azaly et al. (2020) utilised a CSDN controller for a dynamic spectrum resource allocation scheme in cognitive radio networks. The author did not consider several other users including vehicles, robots, and sensors, among others. Huang et al. (2018) investigated the performance of vehicle-to-vehicle data offloading for cellular networks using the collaboration of SDN and MEC technology. To reduce the migrated traffic demand and latency, Zhang et al. (2020a) proposed a suitable ONU assignment scheme for SDN configurable passive optical networks by taking global load-balancing criteria into account. Gao et al. (2020) proposed a load-aware and interference-free medium access control protocol for VANET using SDN technology. However, the missing part is that the aforementioned literature works do not investigate any superactive resource management scheme for software-defined SAGIN by taking different users (human users devices, sensors, robots, vehicles) and different types of tasks (computing, caching, data transmission) along with users' SLA requirements, priority, users budget, and low-latency requirements. The difference between the proposed scheme and related works is given in Table 1.

3 Proposed superactive resource management scheme

3.1 Network architecture and considerations

Figure 1 visualises the overall network architecture for the proposed software-defined SAGIN. Our proposed architecture consists of cooperative and integrated network components such as LEO satellites (space components), UAV (air network components), mobile devices, cloud servers (both edge cloud server and remote server), human users mobile devices, robots, vehicles, cellular and WLAN network devices (in the ground). The MUs' task data can be transferred/received to/from the satellites through the ground backbone networks or UAV devices via inter-satellite links or ISLs (acting as a relay), similar to Jia et al. (2021).

Algorithm 1 Proposed superactive algorithm

```
for each local SDN controller in each round do
    sends the time cycle start Beacon message to its associate access layer network devices along with their cluster members' information.
    collects the resource information from the nodes under its coverage and neighbouring nodes.
    assigns control slots to access layer devices (base station, Wi-Fi access points) and receives the user's task and resource request information from them.
    exchanges the task information with other SDN controllers and updates the resource assignment schedule (for users).
    sorts the user's task first based on their priority and rearranges the sorted task request by using their predicted task completion time first basis while satisfying SLA and budget requirements.
    informs the resource slot schedule to users and actors (i.e., cloud servers, sensors, robots).
    monitors and updates the process (if necessary)
    for each access layer network device do
        forms the cluster group by exchanging hello and reply messages with the associated devices/MUs
        receives the time cycle beacon message from the local SDN controller and sends the control slot info. to the users for task info. submission.
        receives the task info. from users and sends them to the associated local SDN controller
        receives the resource assignment slot from the controller and sends it to the users
        assists the task data forwarding process (i.e., task input and task output data)
        for each actor (cloud server/robot/sensor) do
            executes the assigned task
            transfers the task output to the recipient user
        end for
        for each user do
            sends the respective task info. to access layer device (base station, access point)
            receives the respective time-slot schedule info (to transfer/receive task input/output data)
        end for
    end for
    for each blockchain network device do
        assists the blockchain-based authentication process (i.e., registration confirmation, block generation, validation, smart contract, block confirmation, and data sharing)
    end for
    for each ground and satellite device do
        cooperates with SDN controller
        receives the request from a user, forwards the task data (i.e., if selected as forwarding node), and executes the task (if selected as actor)
    end for
end for
```

The proposed integrated network model consists of both space-air network and ground network components. The ground network consists of several components such as access layer device (i.e., cellular base station and Wi-Fi access point), MU devices (smartphones, robots, sensors, vehicles), backbone network devices, wired and wireless communication link, OpenFlow switches, mobile-edge computing (MEC) cloud server, local SDN controller,

central SDN controller, ground earth station, and remote cloud/central cloud server. The space network consists of LEO satellites. The satellites communicate/connects with the ground network through the ground stations. The air network consists of UAV types devices (i.e., balloons, aerial platforms). With the help of ground backbone network devices and cellular base stations, the UAV-type devices can be connected to the ground network. In this work, the local SDN controller is located near the cellular base station (i.e., a fibre link is used as a communication medium). Whereas, the central SDN controller is placed multiple hops away from the local SDN controller. Both the local and central SDN controllers can collect the network nodes' status information and cloud server resources information (i.e., via request and response message). The well-known open flow protocol is used to transfer data forwarding rules and initiate control operations (i.e., through the network). The SDN controllers are connected through the fibre link. In this work, to get the appropriate resource status (i.e., computing and communication), the local SDN controller communicates with other local SDN controllers and the central SDN controller. The local SDN controller (near the cellular base station) collects the task information from the MU's device (i.e., smartphone) under its coverage and provides resources (i.e., computing, storage, communication) based on the priority of each task, SLA requirements, low latency based selection, users resource usage budget, and resource availability. Based on the assigned task slot, the MUs (i.e., human user devices, sensors, vehicles, and robots) task raw data/result data can be uploaded/downloaded from the suitable processing server and transferred to the destination device. Similarly, MUs can access the caching task data from the selected server and task data can be transferred from the sender to the receiver. For moving devices (i.e., vehicles, robots), the local SDN controller predicts the future location of the moving device ($x_{t+1} = x_t + u_s * \delta_t$) by taking both current location (x_t), journey duration (δ_t), and moving speed into account (u_s). In this work, the MU's devices (i.e., smartphones, robots, sensors, vehicles) can be connected with the access layer devices (i.e., base station, access points) through the wireless link. The edge network nearby computing server is placed near the cellular base station and Wi-Fi access points (i.e., wired/fibre link is used for communication medium). In this work, we have assumed that the channel, cloud resources, and time slot information can be updated through the local SDN controller information exchange with other network devices. This work also assumed that the user's task request information includes the task workload, task input and output data information, SLA information, task priority, user movement information, and budget, among others. The available resources status and capability information are collected by the local SDN controller through the control plane before the resource scheduling for task execution. We have also assumed that the corresponding local SDN controller can exchange and update their scheduling information after exchanging the task and available resource information with other local and central SDN controllers. In this work, three different types of tasks are considered computing type task, caching type task, and only data transfer type task. To cope with the network's dynamic nature, the task-related information, network node status, and resource information are generated randomly for simulation. The associated input and output task data values, SLA, workload per task, and other simulation parameters are discussed more briefly in Section 5 and Table 2. Moreover, to investigate the network congestion status and queueing delays during task completion, we have utilised the M/D/1 queuing model (Amreen et al., 2017).

By ensuring cooperation among SDN controllers and managing the task and resource coordination locally, this local SDN controller can lower the control overhead and minimise the task completion delay for different applications. The information collection phase duration is limited and can be performed through the control plane. Moreover, the information collection phase is done once for all arrived application requests before the resource scheduling phase. Thus, the overhead of the proposed scheme (local SDN controller-based) communications and information collection is lower compared with the CSDN controller-based system. Moreover, the use of multiple local SDN controllers can support a huge amount of applications at the network edge. In case of one local SDN controller failure, the user's devices application (i.e., MUs, vehicles, robots computing, data transmission, and caching task) can be supported by another stable nearby local SDN controller through the request and reply messages. Thus, the proposed system is robust and can avoid single point of failure issues. To avoid any kind of malfunctioning activities, the user identity information can be checked during the initial registration process and regularly during task data transmission time. Similarly, the local SDN controller can monitor the traffic of the user's application to avoid issues like users' application attempts to get more resources and providing wrong information. However, we want to clarify that the detection of malfunctioning users, selfish applications, or malicious attackers is out of the scope of this paper and can be considered for future work.

3.2 Task and resource coordination scheme

This subsection contains the task and resource management process for software-defined SAGIN. In particular, within a simulation round or time-cycle, the proposed superactive scheme activities can be categorised into three major parts:

1 task and resource information collection phase

2 resource scheduling and actor assignment phase

3 multi-user task accomplishment phase.

The detailed time frame model is visualised in Figure 2. Whereas, the proposed superactive resource assignment decision procedure for multi-user task completion and the role of the different participants (i.e., users, actors, and network access layer devices) are outlined in Algorithm 1.

In this work, network access layer devices are cellular base stations and Wi-Fi access points. The local SDN controller decides the best processing actor nodes (i.e., robot, sensor, and cloud server) for MUs' tasks. To understand the overall task execution process, the resource allocation scheme describes a sequential procedure.

Note that, in our work, the control plane functions are done in the control plane and data plane operations are done in the data plane, separately. First, the local SDN controller sorts the task based on priority, latency, and SLA requirements (see Algorithm 1). Then, the controller collects the resource information and determines the delays based on the analytical model (also checks whether they satisfy SLA requirements). Then, based on the task arranging order (lower deadline first basis for multiple task) and associated predicted delays, the SDN controller can determine the necessary resources and assign the number of resources to the user's task to meet the corresponding SLA (please see Subsections 3.3, 3.4 and 3.5 along with Algorithm 1 for more details). To avoid unnecessary delays, we have collected up-to-date information by setting the information collection phase time limit. If there is any failure or problem regarding information collection, the scheduler can rely on other controller information via cooperation. In case of server failure, the SDN controller can monitor the task progress and select another server for task execution. In this work, only the information gathered within a time limit from the servers and devices would be considered.

3.3 Information collection phase

The first phase of the proposed superactive scheme is the resource and task information collection phase. Initially, the network access device receives the time cycle initiation Beacon message from the associated local SDN controller and sends it to associated devices (i.e., robot, sensor, smartphone, and vehicle). Next, the local SDN controller collects the resource information from different nodes (i.e., edge/remote cloud device, processing server, mobile device, robot, vehicle, and UAV) within the network via cooperation along with the associated communication link status. Later, the SDN controller assigns the control slot to users under the network access layer devices (i.e., cellular base station, and Wi-Fi access points) and the users send the task information message (i.e., task specifications, workload, input/output task data information, budget, SLA requirement, and priority) to the associated local SDN controller (i.e., placed near the cellular base station). Note that, some user's task accomplishment requires only cloud device processing and some task requires collaboration from the robot, sensor, and vehicle actors along with cloud device/server actors. The detail regarding the heterogenous task types is discussed in the previous sub-section. After that, the local SDN controller exchanges information with other SDN controllers regarding collected task requests and resource status. Next, by getting the best possible knowledge, the local SDN controller starts the resource scheduling phase. The computing and cache task (Drolia et al., 2017) can be executed with the help of a selected server (local cloud server or remote server) based on the system capability (e.g., high processing speed of the available server and available caching task data in the selected server). The selected server information can be obtained before the task assignment.

3.4 Resource scheduling and actor assignment

The most important phase of our proposed scheme is the resource scheduling phase. In this phase, after analysing the collected tasks associated information along with available resource information, the local associated SDN controller first sorts all task requests based on their task priority basis (high to low). Next, the SDN controller again rearranges the already sorted task request based on their lower predicted task completion time first basis while maintaining the SLA (i.e., required task execution time limit) and budget cost (per task) requirement satisfaction. The predicted task completion time calculation is detailed in the analytical section. All delays regarding task data processing (i.e., computation, caching task) in the best actor device, task data transfer and reception activities, blockchain operation, control, and waiting overhead are included in the task completion delay calculation. Before the task completion delay calculation, the local SDN controller chooses the best available actor (i.e., edge/remote computing cloud device, robots, sensors) for each task completion with minimum latency, best routing path, and best preferable communication link for each task data transfer. After the completion of multiple received task request sorting processes, the local SDN controller schedules time slot order for the completion of all tasks by including actors' task processing period, data transfer time (i.e., input and output) period, and predicted additional overhead period. After the completion of the slot assignment process per task, the local SDN controller informs the users (i.e., human users devices, vehicles) about its slot duration (i.e., slot for transfer input and receiving output data) along with the network access layer device (i.e., for forwarding task data) and actors (i.e., for processing and transferring task data).

3.5 Multi-user task accomplishment phase

The final phase is the multi-user task accomplishment phase. In this phase, the users send their associative task information (i.e., input data) to the selected task processing actors. The task processing actors (i.e., cloud device, robot, and sensor) executes the task and finally send the result to users during their respective slot schedule. Note that, in this work for human users (i.e., smartphone users), three different types of tasks are considered. These are blockchain-based computation and information processing type tasks (i.e., word counter in a passage and face identification from image), caching type tasks (i.e., streaming video), and only E2E data transfer type tasks (i.e., end-to-end sender to receiver message transfer).

Note that, the first step of all types of users task is a blockchain-based authentication process. Any computation, caching, and data transfer type tasks users need to be authenticated to ensure security. Initially, when the users send the transaction request, a block generation (i.e., transaction creation) is completed by the actor device (i.e., cloud/local edge server) along with the user registration, authentication key delivery process (i.e., actors to users for validation), and smart-contract creation process. Next, the created block information is transferred to all other actors and network devices. After the completion of the transaction validation by all participants (i.e., actors and network devices), the task workload (i.e., computation, caching, data transfer) is executed by the actors. Then, the users receive the transaction completion data. After the data sharing/transfer, the blockchain is updated by the actor's node and updated data can be transferred to other participants. Note that, other than the blockchain part, the general computation task consists of task data transfer (i.e., from the sender user device to the actors processing node), uploaded task instructions/data processing (i.e., by an actor), and output/actor processed data transfer (i.e., from actors to user device). Whereas, the caching task consists of users' task information transfer to the actor (i.e., edge-cloud device), delay associated with actors cache search (i.e., for a match) or look up delay, and caching task data transfer delay (i.e., from actors cache server to users device).

Note that, for the sensor actor associated task, the cluster head of associated sensor actors collects the sensing type data from sensor actors. Next, the cluster head (i.e., sensor node) transfers the sensing type task data to the selected device/actor (i.e., cloud device/actor for sensing type data processing) for processing the remaining computation task. Later, the human user's device receives the processed task result (i.e., the output after processing sensing type data) from the selected actor/cloud device. For the robot actors' associated task, two different types are considered (i.e., computation and data transfer task). For the first type of robotic task, a suitable robot actor (i.e., selected by the local SDN controller based on task requirements) first processes the workload (i.e., packaging in the manufacturing factory) of the task by itself after receiving the human user's instruction. Next, after task completion, the robot actor transfers the confirmation data to the human user's device. For the second type of robotic task, each associated robot actor partially completes the task (i.e., taking some pictures using the camera at a certain location) and transfers the captured data to the cloud actor (i.e., edge/remote server) for the completion of remaining sub-part of the task (i.e., useful information extraction from image). Later the task result would be transferred from the cloud actors to human users' devices (i.e., smart-phone) via the selected communication link. For the vehicle-associated task, three different types of tasks (i.e., computing, caching, and data transmission) are examined. Vehicle users can request a computing type task and receive the task output result after assigned actor (i.e, edge/central cloud computing device) processing. Similarly, the vehicle users can access the caching type task data (i.e., audio/video file) from the selected actors (i.e., cloud actor/server) via the proposed network infrastructures. Similarly, data transfer type tasks for vehicle users (i.e., one device to another) can be possible by using the proposed algorithm. Note that, all the actors (i.e., for task execution), routing path (i.e., sender to receiver), and communication link selection for each task are selected by the associated local SDN controller.

4 Numerical performance analysis model

This section assimilates the numerical analysis model. The numerical analysis can help us to determine system performance quickly with multiple performance metrics, deeper understanding of the systems, find approximate accurate results properly with real-time systems settings, and easily compared with other performance analysis. The model contains some important and well-known metrics including mean task completion time, energy expenditure, mean network throughput, mean SLA fulfilment ratio, service provider profit, financial expenditure, time reward ratio (TRR) or TRR, and finance reward ratio or FRR, among others. In this work, the analytical model is used only to show the calculation process. Note that, the result determination process via simulation is not static due to varying data size, link rate, task processing capability, and workload, among others. We have considered the fact that the bandwidth is not fixed and can be changed with time. To cope with the dynamic network scenarios, we have used the varying communication link rate via random selection during the simulation and selected the best possible link for communication purposes. Thus, transfer delays are different during the data transfer process due to varying link status, data size, queuing delays, etc. Similarly, the task data size, task processing capability of actor nodes and workload per task are selected via random selection. The general users can reproduce the results by simulating the steps mentioned in Algorithm 1 and the analytical model can be used to determine different performance metrics. The necessary assumptions regarding tasks, users, networks, and resources are given in Subsection 3.1 and Section 5. The simulation notations and values can be found in Table 2.

4.1 Mean task completion time

The mean task completion time ($\phi^u_{md,i} = \frac{\sum_{i=1}^{n} \phi^u_{ad,i}}{n}$) can be computed by taking the ratio of all users task completion time ($\phi^u_{ad,i}$) sum value and users task count number (n). The task completion time includes all delays associate with different users (i.e., human users device, robot, sensors, and vehicles under the local SDN controller) computing ($\phi^u_{co,i}$), caching ($\phi^u_{ca,i}$), and data transmission ($\phi^u_{dt,i}$) type task completion delay along with initial resource allocation phase delay ($\phi^u_{cp,i}$) and blockchain operation ($\phi^u_{bc,i}$) delay.

Table 1 Comparison of the proposed scheme with related works

Scheme	*Main features*	*Task types and user types*	*SDN type*	*Network type*	*Priority, SLA, low-latency-based resource scheduling*
Zhang et al. (2020b)	SDN-based task offloading for vehicular network	Only single type user and task	Centralised	Only ground network	not considered (all)
Zhou et al. (2019)	Bidirectional offloading for mobile users task	Only single type task and users	Centralised	SAGIN network	Not considered (all)
Huang et al. (2018)	V2V data offloading using SDN and MEC	Only single type	Centralised	Only ground network	Not considered (all)
Abugabah et al. (2020)	Intelligent traffic engineering using multi-path routing	Single type	Centralised	Only ground	Not considered (all)
Li et al. (2020b)	Secured framework for healthcare system using MEC	Single type	Centralised	Only ground	Not considered (all)
Yuan et al. (2020b)	SDN-based offloading/migration for WBAN	Single type	Centralised	Only ground	Only priority-based
Ali Shah et al. (2020)	Cooperative approach for vehicular task offloading	Single type	Decentralised (reactive)	Only ground	Not considered (all)
Montazerolghaem et al. (2020)	Load balancing-based IoT task execution using ILP formulation	Only computing and physical task	Centralised	Only ground	Only QoS-aware
Nazari et al. (2016)	SDN-based satellite communication	Single type	Centralised	Only space and ground	Not considered (all)
Tadros et al. (2020)	SDN-based network management for 5G services	Single type	Decentralised (reactive)	Only ground	Not considered (all)
Sultana et al. (2019)	SDN-based spectrum auction	Single type	Centralised	Only ground	Not considered (all)
Wang et al. (2022)	Prediction of network traffic based on learning approach	Single type	Centralised	Only ground	Not considered (all)
Cao et al. (2021)	Resource allocation for IoV using SDN and MEC	Single type	Decentralised (reactive)	Only ground	Not considered (all)
Our proposed superactive approach	Priority, latency, and SLA-aware resource management scheme	Multiple types of users and tasks	Hybrid SDN (proactive)	Space, air, and ground integrated networks	Considered with the suitable task scheduling order

$\phi^{u}_{md,i}$ calculation can be estimated as follows:

$$\phi^{u}_{md,i} = \frac{\sum_{i=1}^{n} \phi^{u}_{co,i} + \phi^{u}_{ca,i} + \phi^{u}_{dt,i} + \phi^{u}_{cp,i} + \phi^{u}_{bc,i}}{n} \quad (1)$$

where n is the total users task count and u is the user type (i.e., human user device, robot, sensor, vehicle).

The task completion ($\phi^{hd}_{co,i}$) delay for human device-cloud device (i.e., actor) cooperation-based computing task is ascertained as follows:

$$\begin{aligned}\phi^{hd}_{co,i} &= \phi^{hd}_{rt,i} + \phi^{hd}_{ft,i} + \phi^{hd}_{wc,i} + \phi^{hd}_{qd,i} + \phi^{hd}_{pt,i} \\ &= \frac{\Omega^{hd}_{os,i} * d^{hd}_{ha}}{\sigma^{hd}_{hd\to a}} + \frac{\Omega^{hd}_{fs,i} * d^{hd}_{ha}}{\sigma^{hd}_{a\to hd}} + \frac{\Pi^{hd}_{wl,i}}{\lambda^{hd}_{a,i}} \\ &+ \phi^{hd}_{qd,i} + \phi^{hd}_{pt,i} \end{aligned} \quad (2)$$

where $\phi^{hd}_{rt,i}$, $\phi^{hd}_{ft,i}$, $\phi^{hd}_{wc,i}$, $\phi^{hd}_{qd,i}$, $\phi^{hd}_{pt,i}$ are humans users computing task input file/data transfer communication delay (i.e., human user device to cloud device/actor), output file/data transfer delay (i.e., actor to human user device), task instruction (i.e., workload) computing/process delay in the actor server (i.e., to produce output data from input), delay associated with queuing, delay associated with task data propagation latency, respectively. Note that, in this work, we have incorporated the M/D/1 queuing model for queuing delay calculation (Amreen et al., 2017). If the traffic load of the system (ρ) and service rate (υ) are known, the queuing delay can be calculated as follows: $\phi^{hd}_{qd,i} = \frac{\rho}{2\upsilon(1-\rho)}$. $\Omega^{hd}_{os,i}$ and $\Omega^{hd}_{fs,i}$ are the input (i.e., offload data) and output (i.e., actor processed data) data file size of

computing task, respectively. d_{ha}^{hd} is the distance between the human user's device and the actor device/server. $\sigma_{hd\rightarrow a}^{hd}/\sigma_{a\rightarrow hd}^{hd}$ is the link rate (i.e., human users device to actor and vice versa) of computing task data transfer. $\Pi_{wl,i}^{hd}$ and $\lambda_{a,i}^{hd}$ are the computing task instructions (i.e., workload) count and actors processing power, respectively.

The caching task completion ($\phi_{ca,i}^{hd}$) delay for human user device-cloud device (i.e., actor) cooperation-based task is assessed by:

$$\phi_{ca,i}^{hd} = \phi_{rt,i}^{hd} + \phi_{acd,i}^{hd} + \phi_{dd,i}^{hd} + \phi_{cqd,i}^{hd} + \phi_{cpt,i}^{hd} \tag{3}$$

where $\phi_{rt,i}^{hd}$ is the caching task request data transfer (i.e., from human users device to actor/cloud device) delay ($\phi_{rc,i}^{hd} = \frac{\Omega_{rc,i}^{hd} * d_{ha}^{hd}}{\sigma_{hd\rightarrow a}^{hd}}$). $\Omega_{rc,i}^{hd}$ caching type task request (i.e., instructions) size. $\phi_{acd,i}^{hd}$ is the delay associated with cache searching operation (i.e., actors). ($l_{cd,i}^{hd} + z_{cm} * (l_{ald}^{hd} + l_{nld}^{hd})$). $l_{cd,i}^{hd}$ is the looking overhead delay associated with actors cache searching process. The delays associated with caching process is similar as Drolia et al. (2017). z_{cm} is the ratio associated with cache data search failure or miss. l_{ald}^{hd} and l_{nld}^{hd} are latency associated with actors look-up operation (cache search) and network communication (i.e., data transfer) latency. $\phi_{dd,i}^{hd}$ is the cached file (i.e., audio/video data) download (i.e., from actor to humn user device) delay ($\phi_{dd,i}^{hd} = \frac{\Omega_{fc,i}^{hd} * d_{ha}^{hd}}{\sigma_{a\rightarrow hd}^{hd}}$). $\Omega_{fc,i}^{hd}$ is the downloaded file (i.e., cached data) size. $\phi_{cqd,i}^{hd}$ and $\phi_{cpt,i}^{hd}$ are queuing and propagation latency associated with caching type task, respectively.

Next, only data transfer type task completion delay (i.e., end-to-end only data transfer from sender to receiver) is assessed as follows:

$$\begin{aligned}\phi_{dt,i}^{hd} &= \frac{\Omega_{dts,i}^{hd} * d_{he}^{hd}}{\sigma_{hd\rightarrow en}^{hd}} + \tau * \frac{\Omega_{dts,i}^{hd} * d_{enr}^{hd}}{\sigma_{en\rightarrow re}^{hd}} \\ &+ \frac{\Omega_{dts,i}^{hd} * d_{fer}^{hd}}{\sigma_{fn\rightarrow re}^{hd}} + \phi_{dqd,i}^{hd} + \phi_{dpt,i}^{hd}\end{aligned} \tag{4}$$

$\Omega_{dts,i}^{hd}$ is the end-to-end delivery (sender to receiver) data size or data amount. τ is the communication hop number distance between (sender network access device to receiver network access device). In this work, the network access device is either cellular base station or Wi-Fi access point. d_{he}^{hd}, d_{fer}^{hd}, and d_{enr}^{hd} are communication distance (human user device to senders network edge device), communication distance (final receiver associated network edge device to receiver users device), and communication distance (sender network edge device to receiver associated network edge device), respectively. $\sigma_{hd\rightarrow en}^{hd}$, $\sigma_{en\rightarrow re}^{hd}$, and $\sigma_{fn\rightarrow re}^{hd}$ are communication link (human user device to senders network edge device) associated data transfer rate, communication link rate (sender associated network edge device to receiver associated network edge device), and communication link rate (receiver associated network edge device to receiver users device), respectively.

Whereas, the task completion ($\phi_{co,i}^{r}$) delay for robot-based computing task execution by using human device-robot actor cooperation ($\phi_{co,i}^{r} = \phi_{rco,i}^{r}$) and human-robot actor-cloud actor device ($\phi_{co,i}^{r} = \phi_{aco,i}^{r}$) cooperation is enumerated as follows:

$$\begin{aligned}\phi_{aco,i}^{r} &= \phi_{rt,i}^{r} + \phi_{rwc,i}^{r} + \phi_{ot,i}^{r} + \phi_{awc,i}^{r} \\ &+ \phi_{aft,i}^{r} + \phi_{aqd,i}^{r} + \phi_{apt,i}^{r} \\ &= \frac{\Omega_{rt,i}^{r} * d_{hr}^{r}}{\sigma_{hd\rightarrow r}^{hd}} + \frac{\Pi_{rwl,i}^{r}}{\lambda_{r,i}^{r}} + \frac{\Omega_{os,i}^{r} * d_{ra}^{r}}{\sigma_{r\rightarrow a}^{r}} \\ &+ \frac{\Pi_{awl,i}^{r}}{\lambda_{a,i}^{r}} + \frac{\Omega_{fs,i}^{r} * d_{ah}^{r}}{\sigma_{a\rightarrow hd}^{r}} + \phi_{aqd,i}^{r} + \phi_{apt,i}^{r}\end{aligned} \tag{5}$$

where $\phi_{rt,i}^{r}$, $\phi_{rwc,i}^{r}$, $\phi_{ot,i}^{r}$, $\phi_{awc,i}^{r}$, $\phi_{aft,i}^{r}$, $\phi_{aqd,i}^{r}$, $\phi_{apt,i}^{r}$ are task request transfer delay, robot-based task instruction/workload execution/computation, robot actor to cloud actor task data upload delay, cloud actor workload/task instruction processing (i.e., from uploaded data), and cloud actor to human device output data (i.e., cloud actor processed data) transfer delay, associated queuing, and propagation delay for robot-cloud associated task, respectively.

$$\begin{aligned}\phi_{rco,i}^{r} &= \phi_{rt,i}^{r} + \phi_{rwc,i}^{r} + \phi_{ft,i}^{r} + \phi_{qd,i}^{r} + \phi_{pt,i}^{r} \\ &= \frac{\Omega_{rt,i}^{r} * d_{hr}^{r}}{\sigma_{hd\rightarrow r}^{hd}} + \frac{\Pi_{rwl,i}^{r}}{\lambda_{r,i}^{r}} + \frac{\Omega_{fs,i}^{r} * d_{rh}^{r}}{\sigma_{r\rightarrow hd}^{r}} \\ &+ \phi_{qd,i}^{r} + \phi_{pt,i}^{r}\end{aligned} \tag{6}$$

$\phi_{ft,i}^{r}$ is the robot processed task final data (output) transfer delay (i.e., robot actor to human device). $\phi_{qd,i}^{r}$ and $\phi_{pt,i}^{r}$ are associated queuing and propagation delay (i.e., overall task completion), respectively.

Further, the computing task completion ($\phi_{co,i}^{s}$) delay by using human device-sensor actor-cloud actor device cooperation is measured by:

$$\begin{aligned}\phi_{co,i}^{s} &= \phi_{rt,i}^{s} + \phi_{swc,i}^{s} + \phi_{ot,i}^{s} + \phi_{awc,i}^{s} \\ &+ \phi_{aft,i}^{s} + \phi_{qd,i}^{s} + \phi_{pt,i}^{s} \\ &= \frac{\Omega_{rt,i}^{s} * d_{hs}^{s}}{\sigma_{hd\rightarrow s}^{s}} + \frac{\Pi_{swl,i}^{s}}{\lambda_{s,i}^{s}} + \frac{\Omega_{os,i}^{s} * d_{sa}^{s}}{\sigma_{s\rightarrow a}^{s}} \\ &+ \frac{\Pi_{awl,i}^{s}}{\lambda_{a,i}^{s}} + \frac{\Omega_{fs,i}^{s} * d_{ah}^{s}}{\sigma_{a\rightarrow hd}^{hd}} + \phi_{qd,i}^{s} + \phi_{pt,i}^{s}\end{aligned} \tag{7}$$

$\phi_{rt,i}^{s}$, $\phi_{swc,i}^{s}$, $\phi_{ot,i}^{s}$, $\phi_{awc,i}^{s}$, $\phi_{aft,i}^{s}$, $\phi_{qd,i}^{s}$, and $\phi_{pt,i}^{s}$ are human users device to sensors task request/instruction transfer delay, sensor-based task workload/instruction processing, sensor actor to cloud actor data (i.e., sensors processed) upload delay, cloud actor workload/task instruction processing (i.e., processing of sensors data) delay, and cloud actor to human device final output data (i.e., cloud actor processing of sensors uploaded data) transfer delay, associated queuing, and propagation delay for sensor-cloud collaborative task, respectively.

Figure 1 Proposed software defined SAGIN architecture (see online version for colours)

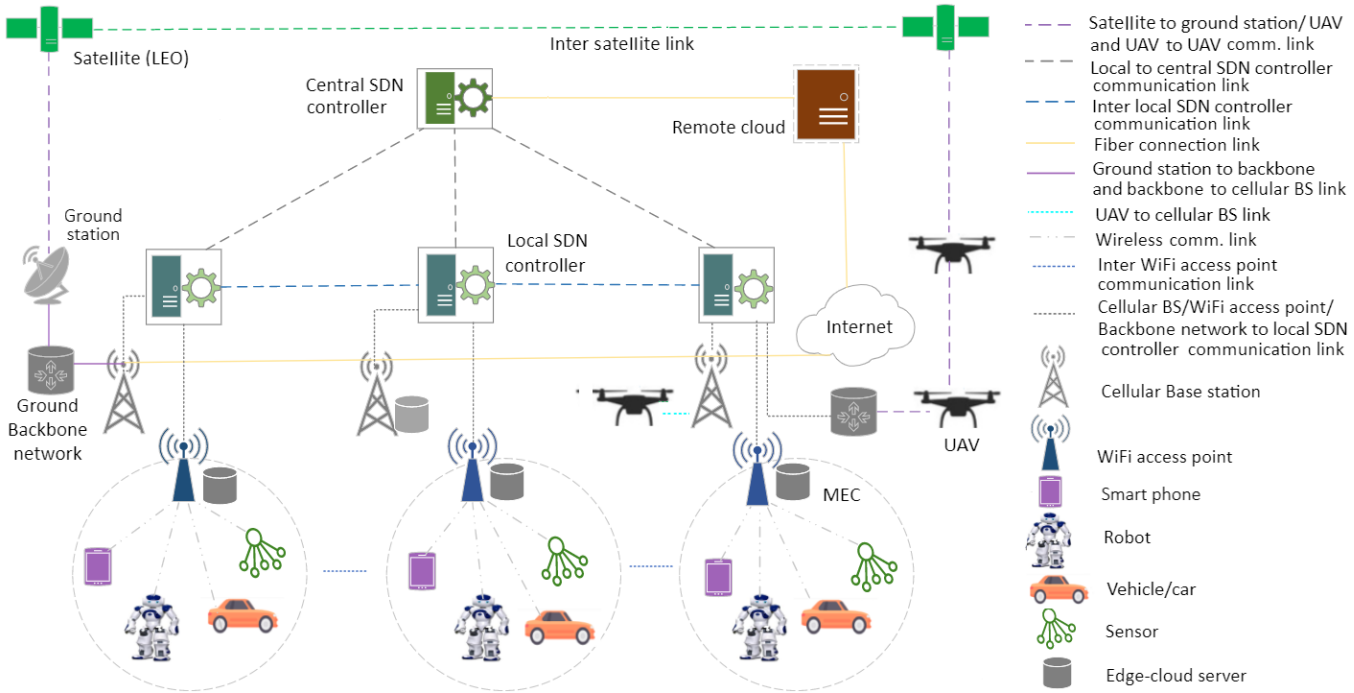
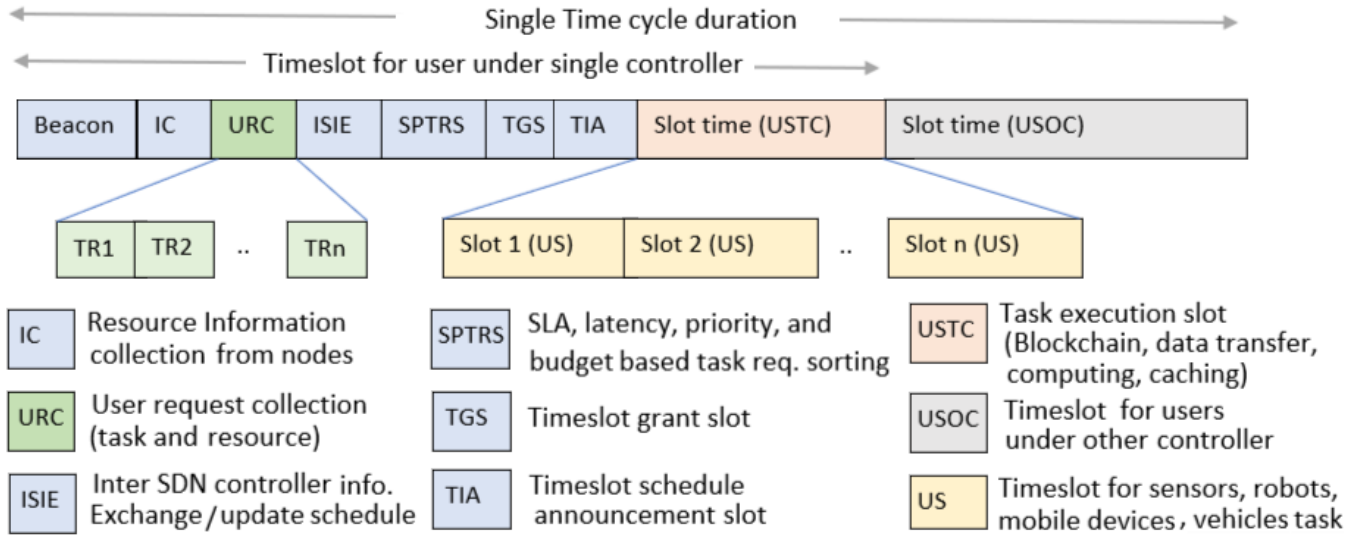


Figure 2 Proposed superactive resource allocation timeframe for multi-user task execution (see online version for colours)



Similarly, the computing ($\phi^v_{co,i}$) and caching ($\phi^v_{ca,i}$) task completion delay calculation by using human device-vehicle-cloud actor cooperation is ascertained by:

$$\begin{aligned}\phi^v_{co,i} &= \phi^v_{rt,i} + \phi^v_{vwc,i} + \phi^v_{ot,i} + \phi^v_{awc,i} \\ &+ \phi^v_{aft,i} + \phi^v_{qd,i} + \phi^v_{pt,i} \\ &= \frac{\Omega^v_{rt,i} * d^v_{hv}}{\sigma^v_{hd\to v}} + \frac{\Pi^v_{vwl,i}}{\lambda^v_{v,i}} + \frac{\Omega^v_{os,i} * d^v_{va}}{\sigma^v_{v\to a}} \\ &+ \frac{\Pi^v_{awl,i}}{\lambda^v_{a,i}} + \frac{\Omega^v_{fs,i} * d^v_{ah}}{\sigma^{hd}_{a\to hd}} + \phi^v_{qd,i} + \phi^v_{pt,i} \end{aligned} \tag{8}$$

The caching task completion ($\phi^v_{ca,i}$) delay by using human user device-vehicle-cloud device cooperation is assessed by:

$$\phi^v_{ca,i} = \phi^v_{rt,i} + \phi^v_{acd,i} + \phi^v_{dd,i} + \phi^v_{cqd,i} + \phi^v_{cpt,i} \tag{9}$$

where $\phi^v_{rt,i}$, $\phi^v_{acd,i}$, $\phi^v_{dd,i}$, $\phi^v_{cqd,i}$, $\phi^v_{cpt,i}$ are the caching request transfer (from human device to cloud actor via vehicle device) delay, cache search delay by cloud actors, cache file download delay (from actor to user device via vehicle), overall queuing, and propagation delay, respectively.

Whereas, the vehicle-based data transfer ($\phi^v_{dt,i}$) task completion delay (i.e., from vehicle sender to human device receiver) calculation is devised as follows:

$$\begin{aligned}\phi^v_{dt,i} &= \phi^v_{rd,i} + \frac{\Omega^v_{dts,i} * d^v_{ve}}{\sigma^v_{v\to en}} + \tau * \frac{\Omega^v_{dts,i} * d^v_{enr}}{\sigma^v_{en\to re}} \\ &+ \frac{\Omega^v_{dts,i} * d^v_{fer}}{\sigma^v_{fn\to re}} + \phi^v_{dqd,i} + \phi^v_{dpt,i} \end{aligned} \tag{10}$$

where $\phi^{v}_{rd,i}$, $\phi^{v}_{dqd,i}$, and $\phi^{v}_{dpt,i}$ are the data transfer request communication (from human device to vehicle device) delay, overall queuing, and propagation delay, respectively.

$\Omega^{v}_{dts,i}$ is the transferred task data amount (sender vehicle to nearby access layer network device). d^{v}_{ve}, d^{v}_{fer}, and d^{v}_{enr} are communication distance (sender vehicle device to nearby network access device), communication distance (receiver associated network access device to receiver users device), and communication distance (sender associated network access layer device to receiver associated network access layer device), respectively. $\sigma^{v}_{v\rightarrow en}$, $\sigma^{v}_{en\rightarrow re}$, and $\sigma^{v}_{fn\rightarrow re}$ are communication link rates between vehicle device to senders network access layer device, between sender associated access layer device to a receiver associated network access device, and between receiver associated network access layer device to receiver user device, respectively.

4.2 Resource allocation phase and blockchain delay time

The resource allocation phase delay ($\phi^{u}_{cp,i}$) is estimated by taking the sum of delays associated with beacon tranfer delay ($\phi^{u}_{btd,i}$), resource information collection from different service provider nodes ($\phi^{u}_{ric,i}$), users request regarding task information collection ($\phi^{u}_{urc,i}$), inter SDN information exchange delay ($\phi^{u}_{isc,i}$), users request (i.e., task information) sorting delay ($\phi^{u}_{ursd,i}$) based on SLA, latency, priority, and budget, resource slot assignment (i.e., from local SDN controller to access layer network device) message delay ($\phi^{u}_{rsa,i}$), and slot schedule announcement (i.e., from network access layer device to users device) delay ($\phi^{u}_{ssa,i}$).

Whereas, the users blockchain delay ($\phi^{u}_{bc,i}$) associated with the task accomplishment is figured out by taking the sum of delays associated with users registration process confirmation ($\phi^{u}_{urpc,i}$), block (i.e., transaction) creation and validation ($\phi^{u}_{bcv,i}$), smart contract process completion ($\phi^{u}_{scpc,i}$), block shiftment process to other service providers ($\phi^{u}_{bspc,i}$), block confirmation process by service providers ($\phi^{u}_{bcp,i}$), access permission completion process ($\phi^{u}_{cap,i}$). This blockchain process completion is necessary in order to ensure user authentication before data sharing, computation, caching task execution process.

4.3 Energy expenditure value calculation

In this work, energy expenditure value is another crucial parameter for SDN-based systems performance evaluation (Fang et al., 2022; Li et al., 2020a). The energy expenditure value for the user's task accomplishment is presented in this subsection. The users total energy expenditure calculation ($\psi^{u}_{ad,i}$) consists of users (i.e., human user device, robot, vehicles, sensors) energy expenditure during resource scheduling or control phase ($\psi^{u}_{cp,i}$), blockchain phase ($\psi^{u}_{bc,i}$), computation ($\psi^{u}_{co,i}$), caching ($\psi^{u}_{ca,i}$), and only data transfer type ($\psi^{u}_{dt,i}$) task execution. Users generally dissipate energy for their task data transmission, data reception, workload/task instruction processing, and idle listening type operation. $\psi^{u}_{ad,i}$ is articulated by:

$$\psi^{u}_{ad,i} = \sum_{i=1}^{n} \psi^{u}_{co,i} + \psi^{u}_{ca,i} + \psi^{u}_{dt,i} + \psi^{u}_{cp,i} + \psi^{u}_{bc,i} \tag{11}$$

where n and u are the total users (i.e., task number) request and type of users (i.e., human users device, robot, vehicles, sensors), respectively. The energy expenditure value ($\psi^{hd}_{co,i}$) value for human device-cloud device cooperation-based computing task execution is conveyed by:

$$\begin{aligned}\psi^{hd}_{co,i} &= \omega^{hd}_{tr,i} * (\phi^{hd}_{rt,i} + \phi^{hd}_{pt,i}) + \omega^{hd}_{rp,i} * \phi^{hd}_{ft,i} \\ &+ \omega^{hd}_{it,i} * \phi^{hd}_{wc,i} + \omega^{hd}_{it,i} * \phi^{hd}_{qd,i}\end{aligned} \tag{12}$$

where $\omega^{hd}_{tr,i}$, $\omega^{hd}_{rp,i}$, and $\omega^{hd}_{it,i}$ are the average energy expenditure value (per second) during human users device data transmission operation, receive, and idle (i.e., waiting) operation, respectively. The other notations are different delays (i.e., $\phi^{hd}_{rt,i}, \phi^{hd}_{pt,i}, \phi^{hd}_{ft,i}, \phi^{hd}_{wc,i}, \phi^{hd}_{qd,i}$) associated with human-cloud actor device coopearation-based computing task completion [i.e., see equation (2) for details].

The energy expenditure ($\psi^{hd}_{ca,i}$) value for human users caching task completion based on human user device-cloud device/actor cooperation is assessed by:

$$\begin{aligned}\psi^{hd}_{ca,i} &= \omega^{hd}_{tr,i} * (\phi^{hd}_{rt,i} + \phi^{hd}_{cpt,i}) + \omega^{hd}_{it,i} * \phi^{hd}_{acd,i} \\ &+ \omega^{hd}_{rp,i} * \phi^{hd}_{dd,i} + \omega^{hd}_{it,i} * \phi^{hd}_{cqd,i}\end{aligned} \tag{13}$$

The average energy expenditure notations (i.e., $\omega^{hd}_{tr,i}$, $\omega^{hd}_{rp,i}$, and $\omega^{hd}_{it,i}$) are discussed earlier [see equation (12)]. Whereas, other notations (i.e., $\phi^{hd}_{rt,i}$, $\phi^{hd}_{cpt,i}$, $\phi^{hd}_{dd,i}$, $\phi^{hd}_{acd,i}$, $\phi^{hd}_{cqd,i}$) indicate different delays associated with caching task completion [see equation (3) for more details].

Next, the energy expenditure value ($\psi^{hd}_{dt,i}$) associated with human users only data transfer type task completion (i.e., from sender to receiver) is estimated by:

$$\begin{aligned}\psi^{hd}_{dt,i} &= \omega^{hd}_{tr,i} * \left(\frac{\Omega^{hd}_{dts,i} * d^{hd}_{he}}{\sigma^{hd}_{hd\rightarrow en}} + \phi^{hd}_{dpt,i} \right) + \omega^{hd}_{it,i} \\ &* \left(\tau * \frac{\Omega^{hd}_{dts,i} * d^{hd}_{enr}}{\sigma^{hd}_{en\rightarrow re}} + \phi^{hd}_{dqd,i} \right) + \omega^{hd}_{rp,i} \\ &* \left(\frac{\Omega^{hd}_{dts,i} * d^{hd}_{fer}}{\sigma^{hd}_{fn\rightarrow re}} \right)\end{aligned} \tag{14}$$

The average transmit, receive, idle energy expenditure notations (i.e., $\omega^{hd}_{tr,i}$, $\omega^{hd}_{rp,i}$, and $\omega^{hd}_{it,i}$) are discussed earlier [see equation (12)]. Whereas, other notations indicate different delays associated with data transfer task completion [see equation (4) for more details].

The energy expenditure ($\psi^{r}_{co,i}$) cost for computing task execution by using human device-robot actor ($\psi^{r}_{co,i} = \psi^{r}_{rco,i}$) and human-robot actor-cloud actor device ($\psi^{r}_{co,i} = \psi^{r}_{aco,i}$) cooperation is denoted by:

$$\begin{aligned}\psi^{r}_{aco,i} &= \omega^{hd}_{tr,i} * (\phi^{r}_{rt,i} + \phi^{r}_{apt,i}) + \omega^{r}_{op,i} * \phi^{r}_{rwc,i} \\ &+ \omega^{r}_{tr,i} * \phi^{r}_{ot,i} + \omega^{hd}_{it,i} * (\phi^{r}_{awc,i} + \phi^{r}_{aqd,i}) \\ &+ \omega^{hd}_{rp,i} * \phi^{r}_{aft,i}\end{aligned} \tag{15}$$

$\psi^r_{rco,i}$ is determined by using the following equation:

$$\psi^r_{rco,i} = \omega^{hd}_{tr,i} * (\phi^r_{rt,i} + \phi^r_{pt,i}) + \omega^r_{op,i} * \phi^r_{rwc,i} + \omega^{hd}_{rp,i} * \phi^r_{ft,i} + \omega^{hd}_{it,i} * \phi^r_{qd,i} \quad (16)$$

where $\omega^r_{op,i}$, $\omega^r_{tr,i}$, and $\omega^{hd}_{it,i}$ are the average energy expenditure cost (per second) for robots task instruction processing, robots data transmission, and human devices idle waiting operation, respectively. The associated delay notations definition can be found in equations (5) and (6).

Further, the energy expenditure ($\psi^s_{co,i}$) cost calculation for human device-sensor actor-cloud actor device cooperation-based computation task completion is delineated by:

$$\psi^s_{co,i} = \omega^{hd}_{tr,i} * (\phi^s_{rt,i} + \phi^s_{pt,i}) + \omega^s_{op,i} * \phi^s_{swc,i} + \omega^s_{tr,i} * \phi^s_{ot,i} + \omega^{hd}_{it,i} * (\phi^s_{awc,i} + \phi^s_{qd,i})\omega^{hd}_{rp,i} * \phi^s_{aft,i} \quad (17)$$

where $\omega^s_{op,i}$ and $\omega^s_{tr,i}$ are the average energy expenditure cost for sensors task instruction processing and data transmission operation, respectively. The readers may be referred to equation (7) for other remaining delay notation definitions.

Similarly, the user's energy expenditure value for computing ($\psi^v_{co,i}$) and caching ($\psi^v_{ca,i}$) task completion by using human device-vehicle-cloud actor cooperation is ascertained by:

$$\psi^v_{co,i} = \omega^{hd}_{tr,i} * (\phi^v_{rt,i} + \phi^v_{pt,i}) + \omega^v_{op,i} * \phi^v_{vwc,i} + \omega^v_{tr,i} * \phi^v_{ot,i} + \omega^{hd}_{it,i} * (\phi^v_{awc,i} + \phi^v_{qd,i}) + \omega^{hd}_{rp,i} * \phi^v_{aft,i} \quad (18)$$

$$\psi^v_{ca,i} = \omega^{hd}_{tr,i} * (\phi^v_{rt,i} + \phi^v_{cpt,i}) + \omega^{hd}_{it,i} * (\phi^v_{acd,i} + \phi^v_{cqd,i}) + \omega^{hd}_{rp,i} * \phi^v_{dd,i} \quad (19)$$

where $\omega^v_{op,i}$ and $\omega^v_{tr,i}$ are the average energy expenditure cost (per second) for vehicles task instruction processing and data transmission operation, respectively. The other remaining delay notations definition can be found in equations (8) and (9).

Users energy expenditure value calculation for vehicles data transfer ($\psi^v_{dt,i}$) task completion (i.e., vehicle sender to human device receiver) is devised by:

$$\psi^v_{dt,i} = \omega^{hd}_{tr,i} * (\phi^v_{rd,i} + \phi^v_{dpt,i}) + \omega^v_{tr,i} * \frac{\Omega^v_{dts,i} * d^v_{ve}}{\sigma^v_{v\to en}} + \omega^{hd}_{it,i} * \left(\tau * \frac{\Omega^v_{dts,i} * d^v_{enr}}{\sigma^v_{en\to re}} + \phi^v_{dqd,i}\right) + \omega^{hd}_{rp,i} * \frac{\Omega^v_{dts,i} * d^v_{fer}}{\sigma^v_{fn\to re}} \quad (20)$$

where $\omega^{hd}_{tr,i}$, $\omega^v_{tr,i}$, $\omega^{hd}_{it,i}$, $\omega^{hd}_{rp,i}$ are the average energy expenditure (per second) during the human users device data transmission, vehicle users data transmission, idle status due to waiting, and data receive operation, respectively. The other remaining delay notations definition can be found in equation (10).

The energy expenditure for resource allocation phase ($\psi^u_{cp,i}$) is estimated by taking the sum of energy expenditure values associated with beacon reception ($\omega^{hd}_{rp,i} * \phi^u_{btd,i}$), SDN controllers resource information collection ($\omega^{hd}_{it,i} * \phi^u_{ric,i}$), users request (task information) transfer ($\omega^{hd}_{tr,i} * \phi^u_{urc,i}$), inter SDN controller information exchange ($\omega^{hd}_{it,i} * \phi^u_{isc,i}$), users request (task) sorting ($\omega^{hd}_{it,i} * \phi^u_{ursd,i}$), resource slot assignment ($\omega^{hd}_{it,i} * \phi^u_{rsa,i}$), and slot schedule message duration ($\omega^{hd}_{rp,i} * \phi^u_{ssa,i}$).

The energy expenditure cost for users blockchain phase ($\psi^u_{bc,i}$) is figured out by taking the sum value of energy cost (i.e., due to transmission, reception, idle duration) associated with users registration process ($\omega^{hd}_{tr,i} * \phi^u_{utrpc,i} + \omega^{hd}_{rp,i} * \phi^u_{urrpc,i} + \omega^{hd}_{it,i} * \phi^u_{uirpc,i}$), block creation and validation ($\omega^{hd}_{tr,i} * \phi^u_{btcv,i} + \omega^{hd}_{rp,i} * \phi^u_{brcv,i} + \omega^{hd}_{it,i} * \phi^u_{bicv,i}$), smart contract process completion ($\omega^{hd}_{it,i} * \phi^u_{scpc,i}$), block shiftment to other service providers ($\omega^{hd}_{it,i} * \phi^u_{bspc,i}$), block confirmation by service providers ($\omega^{hd}_{tr,i} * \phi^u_{btcp,i} + \omega^{hd}_{rp,i} * \phi^u_{brcp,i} + \omega^{hd}_{it,i} * \phi^u_{bicp,i}$), access permission completion process ($\omega^{hd}_{rp,i} * \phi^u_{cap,i}$).

4.4 Mean network throughput

The mean network throughput is another crucial metric for the evaluation of SDN-based network performance (Zhang et al., 2022; Tadros et al., 2020). The network throughput is determined by using the ratio value of the total received packet ($\zeta^u_{su\to ru}$) number at the receiver and task data transfer (from sender to receiver) completion time ($\Phi^u_{su\to ru}$). Whereas, the mean throughput ($\mu^u_{ntut,i}$) is assessed by the ratio of the total net. throughput value and the total number of users requested task (n).

$$\mu^u_{ntut,i} = \frac{\zeta^u_{su\to ru}}{n * \Phi^u_{su\to ru}} = \frac{1}{n} * \frac{\sum_{i=1}^{n} \xi^u_{su\to ru}}{\Phi^u_{su\to ru}} \quad (21)$$

where $\xi^u_{su\to ru}$ is the received packet (at the receiver) number associated with one user's task.

4.5 Mean SLA fulfilment ratio

The mean SLA requirement fulfilment ratio ($\eta^u_{sla,i}$) is determined by dividing the successful number (λ_{st}) of the task that satisfies the given time limit for task completion with total users (i.e., human device, sensors, robot, vehicle) requested task number (λ_{tt}).

$$\eta^u_{sla,i} = \frac{\lambda_{st}}{\lambda_{tt}} = \frac{\lambda_{tt} - \lambda_{ft}}{\lambda_{tt}} \quad (22)$$

where λ_{ft} is the user's task that is unsatisfied with the SLA requirement (i.e., the time limit for task completion).

Table 2 Simulation notations and values

Notations	*Name, values, and units*
$\omega^u_{tr,i}$, $\omega^u_{rp,i}$, $\omega^u_{it,i}$, $\omega^u_{op,i}$	User device ($u \in r$, v, hd, s) average (one second cost) transmit (0.37 W), receive (0.31 W), idle state (0.005 W), task instruction processing (0.5 W) energy expenditure
$\epsilon^u_{acu,i}/\epsilon^u_{bwu,i}/\epsilon^u_{cpo,i}/\epsilon^u_{bco,i}$, $\epsilon^u_{idw,i}$, $\alpha^u_{spc,i}$	Users avg. payment (i.e., per minute) for actors/bandwidth/control phase, and blockchain phase resources (USD0.4 for time sensitive/USD0.3 for delay tolerant task), waiting time/idle time operation (USD0.1), service provider avg. cost (USD0.25 for resource usage)
$\Omega^u_{os,i}/\Omega^u_{fs,i}$, $\Pi^u_{wl/rwl/awl/swl/vwl,i}$, λ^u_i, $\phi^{hd}_{acd,i}$, $\phi^u_{rt,i}$, $d^u_{s\to r}$	Transferred data size for offload input/output (5–100 KB for computation, 1–20 KB for caching, 5–50 KB for data transfer task), workload (1 K cycle per bit processing), actor/user device processing speed (max. 3 GHz for cloud device, 1 GHz for vehicle/robot/human device, 500 MHz for sensor), cache search delay (ms), request transfer delay (ms), varied distance between sender to receiver (1–1,000 m)
$\sigma^u_{s\to r}$, $\phi^u_{qd/aqd/cqd/dqd,i}$, $\phi^u_{pt/apt/cpt/dpt,i}$, τ	Data rate per link value (for Wi-Fi connection 100–300 Mbps, for cellular link 30–100 Mbps, for fibre link max. 1 Gbps, satellite associated link 1–10 Mbps, UAV associated link 1–25 Mbps), queuing delay (ms), propagation delay (ms), varied intermediate hop (sender to receiver) number (1–5)
$\phi^u_{btd,i}$, $\phi^u_{ric,i}$, $\phi^u_{urc,i}$, $\phi^u_{isc,i}$, $\phi^u_{ursd,i}$, $\phi^u_{rsa,i}$, $\phi^u_{ssa,i}$	Beacon transfer delay (ms), resource information collection delay (1–5 ms), users request/task info. collection delay (ms), inter sdn controller information exchange delay (1–10 ms), user request sorting delay (ms), slot assignment message delay (ms), schedule announcement delay (ms)
n, $\phi^u_{urpc,i}$, $\phi^u_{bcv,i}/\phi^u_{scpc,i}$, $\phi^u_{bspc,i}/\phi^u_{bcp,i}$, $\phi^u_{cap,i}$, distance between network access device and cloud, users to network access device, network access device to local SDN controller, local to remote SDN controller, moving speed of robot/human user, vehicle	Total users (vary), blockchain user registration completion delay (1–5 ms), block creation, validation/smart contract arrangement delay (1–50 ms), block shifment/confirmation delay (1–10 ms), access permission delay (1–10 ms), 100 m − 1Km, 1–100 m, 100 m−1Km, 1–5 Km, 1–10 m/s, 1–20 m/s
$\phi^u_{md,i}$, $\psi^u_{ad,i}$, $\mu^u_{ntut,i}$, $\eta^u_{sla,i}$, $\theta^u_{fc,i}$, $\beta^u_{spp,i}$	Mean task completion time (ms, vary), energy expenditure value (mW, vary), mean network throughput (Kbps), mean SLA fulfilment ratio (%), users financial payment cost (USD), profit value for service provider (USD)

4.6 Users financial payment value

The users total financial payment value ($\theta^u_{fc,i}$) is determined by summing up the financial cost value for actor resource usage ($\delta^u_{ac,i}$), network (i.e., bandwith/storage/forwarding) resource usage ($\delta^u_{bw,i}$) for task data transfer, control phase scheduling ($\delta^u_{co,i}$), blockchain operation ($\delta^u_{bo,i}$), idle waiting time ($\delta^u_{iu,i}$) during computing, caching, and data transmission task. $\theta^u_{fc,i}$ calculation is estimated by:

$$\begin{aligned}\theta^u_{fc,i} &= \delta^u_{ac,i} + \delta^u_{bw,i} + \delta^u_{iu,i} + \delta^u_{co,i} + \delta^u_{bo,i} \\ &= \sum_{i=1}^{n} \phi^u_{acu,i} * \epsilon^u_{acu,i} + \phi^u_{bwu,i} * \epsilon^u_{bwu,i} + \phi^u_{idw,i} \\ &\quad * \epsilon^u_{idw,i} + \phi^u_{cpo,i} * \epsilon^u_{cpo,i} + \phi^u_{bco,i} * \epsilon^u_{bco,i} \end{aligned} \tag{23}$$

where $\epsilon^u_{acu,i}$, $\epsilon^u_{bwu,i}$, $\epsilon^u_{idw,i}$, $\epsilon^u_{cpo,i}$, and $\epsilon^u_{bco,i}$ are the average payment value (i.e., per minute service) for actors (e.g., cloud device) resource usage (i.e., processing/transfer), bandwidth resource (e.g., for upload, download data transfer) usage, idle waiting (i.e., for service) time, control phase opeartion, and blockchain operation, respectively.

4.7 Service provider profit value

In this work, the service provider profit value is estimated by taking the difference value between the collected revenue (i.e., users' financial cost for service) and service providers' cost (i.e., associated with deployment, maintenance, and resource buying). If $\alpha^u_{cr,i}$ is the collected revenue from users and $\alpha^u_{spc,i}$ is the service provider cost (i.e., perusers task execution service), then service provider profit value ($\beta^u_{spp,i}$) is ascertained by:

$$\beta^u_{spp,i} = \sum_{i=1}^{n} (\alpha^u_{cr,i} - \alpha^u_{spc,i}) \tag{24}$$

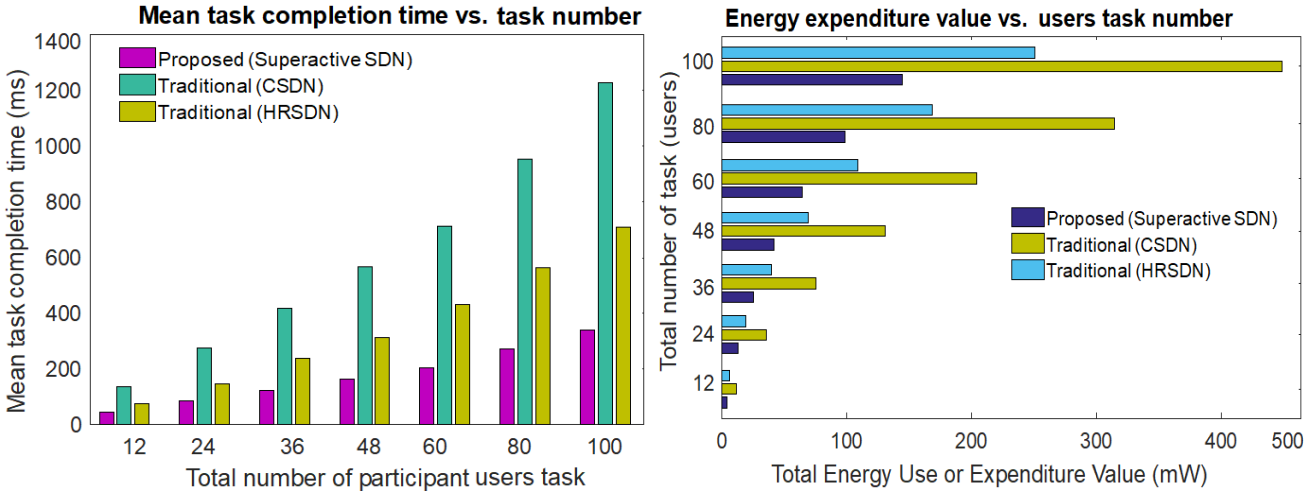
4.8 Time reward ratio

The TRR can be ascertained by using the ratio value between the total task accomplishment time gain (i.e., difference between the task completion time and time limit value) value ($\kappa^u_{tg,i}$) and task time limit value ($\kappa^u_{tl,i}$) for task completion.

$$TRR = \frac{\kappa^u_{tg,i}}{\kappa^u_{tl,i}} = \sum_{i=1}^{n} \frac{(\kappa^u_{tl,i} - \phi^u_{ad,i})}{\kappa^u_{tl,i}} \tag{25}$$

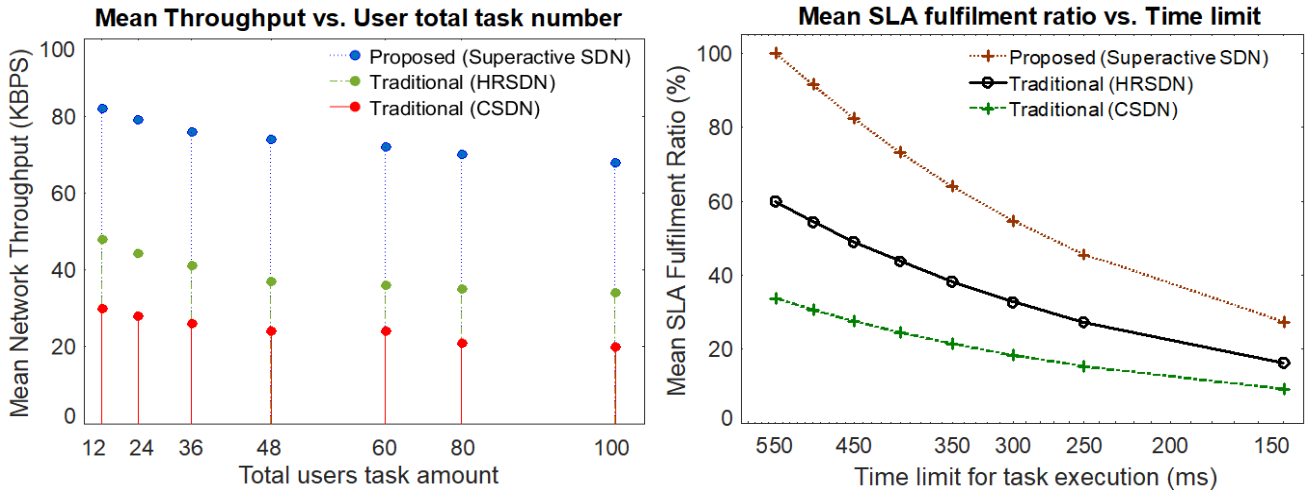
where $\phi^u_{ad,i}$ is the user's task completion time and n is the task (i.e., users task) number.

Figure 3 Mean task completion time and energy expenditure cost (see online version for colours)



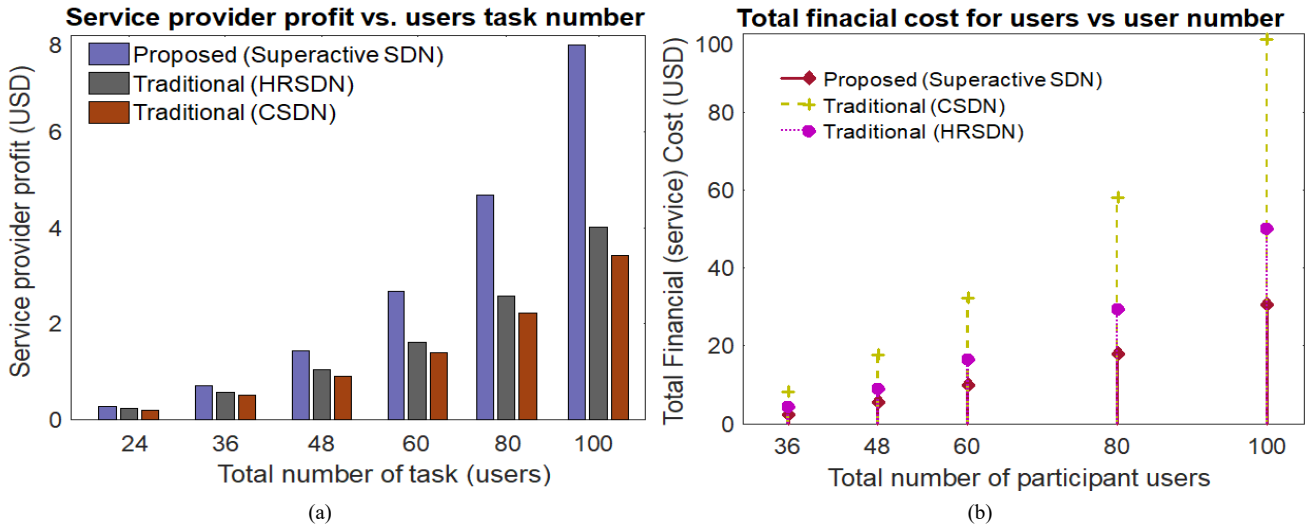
(a) (b)

Figure 4 Mean network throughput and SLA fulfilment ratio results (see online version for colours)



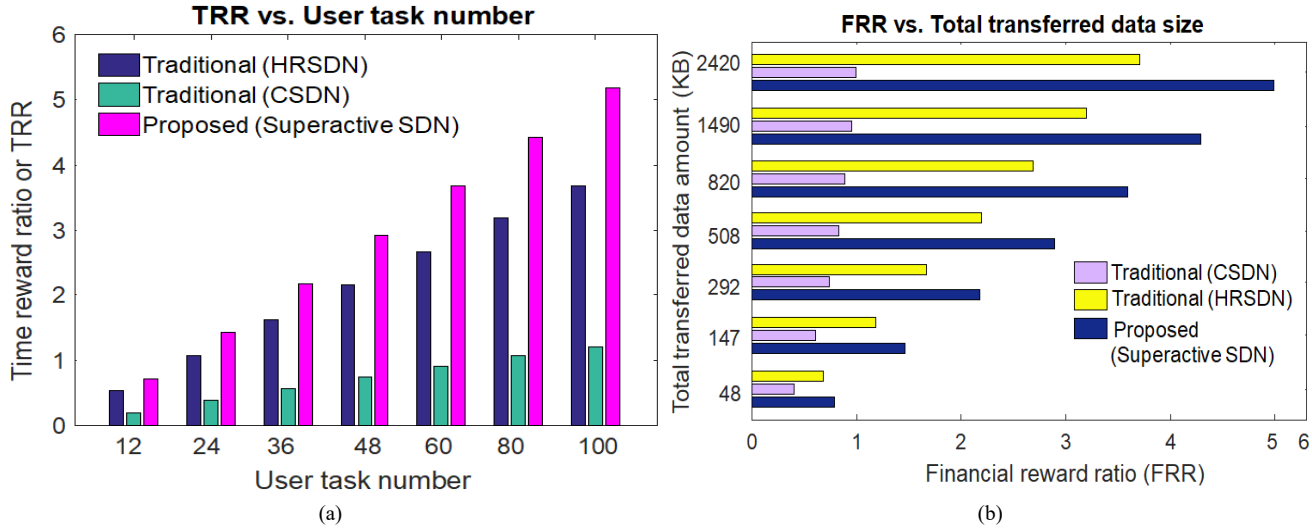
(a) (b)

Figure 5 Service provider profit and users financial cost for service results (see online version for colours)



(a) (b)

Figure 6 Users time and FRR results (see online version for colours)



(a) (b)

4.9 Financial reward ratio

The FRR is figured out by using the ratio value between the total financial cost gain (i.e., the difference between the user's budget and users' required financial cost for task execution) value ($\theta^u_{fg,i}$) and users budget value ($\theta^u_{bg,i}$) for task completion.

$$FRR = \frac{\theta^u_{fg,i}}{\theta^u_{bg,i}} = \sum_{i=1}^{n} \frac{(\theta^u_{bg,i} - \theta^u_{fc,i})}{\theta^u_{bg,i}} \tag{26}$$

where $\theta^u_{fc,i}$ is the users financial (i.e., payment value) cost for task completion (i.e., resource usage).

5 Results and performance analysis

This section illustrates the performance results of the superactive resource management scheme along with the the traditional CSDN scheme based on CSDN controller and first come first served service (e.g., Abugabah et al., 2020; Gao et al., 2020; Zhang et al., 2020c; Wang et al., 2022) and traditional HRSDN or hybrid reactive SDN controller-based and first come-first served-based service (e.g., Nkenyereye et al., 2021; Ali Shah et al., 2020; Cao et al., 2021).

For simulation, three different types of users task (i.e., only blockchain-based computation, caching, and data transmission task) are considered in our MATLAB-based simulation process. The user's task amount is chosen between the 12 to 100 number range based on an incremental basis. The computation task offloaded (i.e., input/output) data range is selected randomly between 5 KB to 100 KB range. The caching task data range (i.e., random selection) is selected between the 1 KB to 20 KB range. The data transmission (i.e., transferred from sender to receiver) data range is within (i.e., which is selected randomly for each task) the 1–50 KB range. In this work, the typical example of the computing task is an event or human face detection or identification from a captured image by a camera (installed in the human user's device or robots, vehicles). Another computing type task is the result generated from the sensory data (i.e., pollution detection). A typical example of the caching type task is a video download from a selected server. In this work, we have assumed that the task (computing and caching) workload is the total number of CPU cycles or instructions required to process the transferred data (per computing and caching task) by selected servers. Note that, the corresponding task input data (captured image or sensor data) is transferred or offloaded to the server for workload processing (i.e., result generation from the input data) by the corresponding device (i.e., mobile device, robot, sensor, and vehicle). In this work, the task workload is related to the task data input size (i.e., workload count is 1,000 cycles per bit processing). The workload count per requested task is thus varied between the 5 M cycle to 100 M cycle range. Other performance parameters and values associated with the numerical model calculation and results are given in Table 2. In the following this paper discusses important results consisting of mean task completion time, energy expenditure value, mean throughput, mean SLA fulfilment ratio, profit for the service provider, and financial payment cost (users) metrics, among others. For some example regarding the computing and caching type task data and workload selection, the readers may refer to Zhang et al. (2015b), Yuan et al. (2020a), Zhang et al. (2015a) and Drolia et al. (2017).

Figures 3(a) and 3(b) plot the mean task completion time and energy expenditure performance of the proposed superactive scheme, respectively. Figures 3(a) and 3(b) indicate an increasing trend of mean task completion time value and energy expenditure value with the increasing user's task amount value, respectively. The mean task completion time and energy expenditure cost increment with a large task execution number are expected due to a large number of workload processing and communication

overhead latency compared with its small number of task execution counterparts.

Importantly, the proposed superactive resource allocation scheme achieves lower mean task completion time and energy expenditure cost for multiple types of task execution over software-defined SAGIN networks than the traditional scheme based on CSDN controller and first come first served service and traditional HRSDN controller-based scheme with FCFS service, respectively. The underlying reason for such a result is that the proposed superactive scheme uses a suitable local SDN controller for decision-making by taking into account the low communication and computation overhead of centralised and local decentralised controllers (i.e., located near the users). Moreover, the proposed superactive decision-making process assigns resources to users by taking into account collaboration opportunities with different SDN controllers, different devices capability (i.e., both space-air and ground networks), available resources, best possible communication link selection for a different task, incorporation of suitable task sorting strategy based on users priority, low latency, budget and financial cost associated with the services, and SLA requirements, among others. The traditional CSDN scheme suffers from worse energy expenditure and mean task completion results than others. This is obvious as the traditional CSDN scheme causes higher control overhead for central SDN controller-based decision making for resource allocation. The traditional HRSDN controller-based scheme achieves second-best results in terms of energy expenditure and mean task completion results. The underlying reason is that the traditional HRSDN scheme relies on a decentralised controller for local task execution. Whereas, the traditional HRSDN scheme depends on the decentralised controller for local and centralised controller for non-local task execution and resource assignment process. Moreover, the decision-making process is reactive, not proactive. Further, the traditional HRSDN scheme experience higher waiting latency due to its first-come-first-served or FCFS manner-based task sorting process rather than the proposed schemes users' priority and SLA requirement basis. Different from the other schemes, the proposed scheme coordinates the decision-making process by collecting prior information on different users' statuses and other nearby SDN controllers. Figure 3(a) reveals that for 60 users (i.e., vehicles, sensors, robots, MU's blockchain-based computing, caching, data transfer) task amount, the mean task completion time cost for the proposed superactive scheme, traditional HRSDN, and traditional CSDN are 201.86 ms, 430 ms, and 714 ms, respectively.

Figure 4(a) investigates the mean network throughput for increasing heterogeneous task amount under three different schemes (i.e., proposed superactive, traditional HRSDN, and traditional CSDN scheme). The higher network throughput results for small and large task number execution validate the supremacy of the proposed superactive scheme over others. Due to increasing control overhead and task processing delay, the network throughput can be decreased slightly with the increment of heterogenous task amount. The figure notifies that due to higher control overhead and additional waiting time overhead, the traditional HRSDN and traditional CSDN scheme secures the second-best and worse task execution scheme in terms of higher mean network throughput. In Figure 4(a) when the task amount reaches 36, the mean network throughput for the proposed superactive SDN controller-based resource allocation scheme, traditional CSDN, and traditional HRSDN scheme are approximately 75 Kbps, 25 Kbps, and 40 Kbps, respectively.

The experimental results in Figure 4(b) visualises the mean SLA fulfilment ratio value of the proposed scheme along with its counterpart schemes for varying task completion time limit value. In all three schemes, the mean SLA fulfilment ratio value raises as the time limit value for task execution increases. As shown in Figure 4(b), due to user priority, low latency, and service execution requirement (per task) based on resource allocation, the mean SLA fulfilment ratio of the proposed superactive scheme is higher than the traditional CSDN and traditional HRSDN scheme. In both traditional CSDN and traditional HRSDN schemes, the resource allocation is based on an FCFS basis rather than multiple tasks with low latency, priority, and tasks service execution requirement basis. However, due to lower control, waiting, and communication overhead, the traditional HRSDN scheme can slightly improve the SLA fulfilment ratio of the traditional CSDN controller-based scheme. For example in Figure 4(b), when the time limit for task execution is 500 ms, the mean SLA fulfilment ratio of the proposed superactive scheme, traditional HRSDN, and traditional CSDN are approximately 91.58%, 55%, 31%, respectively.

The results in Figure 5(a) hint that the proposed superactive scheme can offer higher service provider profit than the compared traditional HRSDN and traditional CSDN schemes. Note that, the service provider's profit value depends on the revenue from the users (i.e., users' financial payment cost) and the service provider cost value difference. The figure also demonstrates that a large amount of service provider profit can be achieved in all compared schemes when the task execution request number is large. By contrast, with the decrease in the task amount, the service provider profit of the proposed superactive scheme, traditional HRSDN, and traditional CSDN scheme almost highlight a downward trend. It is highly desirable as the service provider profit reduces in the traditional HRSDN and traditional CSDN schemes due to higher waiting time during execution and control overhead. Whereas the waiting time and control overhead are relatively lower in the proposed superactive scheme, thus generating more profit for the service providers. The service provider profit is comparably lowest in the traditional CSDN scheme due to its inefficient resource allocation policy, large waiting delay, and higher control message exchange time overhead. In Figure 5(a), when the task number execution reaches 100, the service provider profit value for the proposed superactive resource allocation scheme, traditional CSDN, and traditional HRSDN schemes are approximately USD7.81, USD3.42 and USD4.01, respectively.

The results in Figure 5(b) display the financial cost for user upgrades with a higher number of participant users' task execution. This implies that more resources are required to complete a large number of task execution than the small amount of task execution. The figure also specifies that due to an efficient resource allocation policy, the proposed superactive scheme requires lower task completion delay (i.e., resource usage) and thus offers a small amount of financial cost than the compared traditional CSDN, and traditional HRSDN scheme. The traditional CSDN scheme requires a large amount of financial cost due to its additional task completion time and resource usage. In Figure 5(b), when the participant users task amount is 100, the user's financial cost (i.e., service provider revenue) for the proposed superactive scheme, traditional HRSDN, and traditional CSDN scheme task execution are approximately USD30.53, USD50.14 and USD98.72, respectively.

The contrast between the TRR results and users' task execution number is depicted in Figure 6(a) for the proposed superactive SDN controller scheme based on priority, latency, and SLA-awareness along with traditional CSDN and traditional HRSDN scheme. Note that the TRR is determined by taking the ratio of task execution time difference value (i.e., the time difference between the task completion time and task execution time limit) and task execution time limit value. The results in Figure 6(a) clearly denote that the TRR gradually upgrades to a large number with the addition of the task execution number value. That is TRR value for small task execution numbers can not outperform a large number of task execution counterparts. Hence, for varying user task amount values, the TRR ratio of the proposed superactive scheme is highest than the traditional CSDN and traditional HRSDN schemes. Due to the maximal task completion time value, the traditional CSDN scheme affords the smallest TRR ratio value than others. Whereas, the second-highest TRR ratio is obtained in the traditional HRSDN scheme due to its comparatively lower control overhead and latency than the traditional CSDN scheme. Figure 5(b) exemplifies that if the user task request is 100, the user TRR ratio for the proposed superactive scheme, traditional HRSDN, and traditional CSDN scheme are approximately 5.18, 3.69, and 1.20, respectively.

Figure 6(b) outlines the contrast between the FRR and total end-to-end transferred data amount for users' task execution. The FRR ratio value can be acquired by taking the ratio of monetary cost difference value (i.e., between the user's budget and actual financial cost for task completion) and the user's budget value. Figure 6(b) notifies that a large data transfer amount value (i.e, data size) generates a higher FRR ratio than the smaller data transfer size value. It shows that due to smaller user financial cost value for task completion and resource usage, the proposed (superactive SDN) scheme can attain a higher FRR ratio than the traditional CSDN and traditional HRSDN schemes. Whereas, the traditional CSDN scheme achieves the smallest FRR ratio due to its highest user monetary expense/financial cost for services (i.e., resources and task execution). The traditional HRSDN scheme requires the second-lowest financial expense for services, thus producing a moderately higher FRR ratio (i.e., second-best) than the traditional CSDN scheme. Figure 6(b) specifies that when the transferred data amount (i.e., transferred data size) is 1,490 KB, the FRR ratio for the proposed superactive scheme, traditional HRSDN, and traditional CSDN scheme are approximately 4.29, 3.20, and 0.94, respectively. The simulation results demonstrate the beneficiary nature of the proposed superactive scheme for heterogeneous users' multiple task execution over software-defined SAGIN.

6 Conclusions

This paper introduces a superactive resource management scheme for software-defined SAGIN-based applications. The proposed superactive scheme assigns resources for different users' task accomplishments based on priority (i.e., whether the task is time-sensitive or delay-tolerant), SLA requirement and budget satisfaction, and low latency. To achieve the highest gain in terms of task completion delay, energy dissipation, and financial payment cost, this superactive resource management scheme investigates not only the most efficient actor/task processing (i.e., robot, cloud device, sensor, and vehicle) device selection but also best-preferred communication link selection for task completion. This works also presents a superactive resource assignment algorithm and timing model along with a network architecture model that coordinates local and remote SDN controller, ground network device, space and air network device, cloud device, and users device, among others. This paper briefly discusses a numerical analysis model that contains some salient metrics such as mean task completion delay, energy expenditure calculation model, financial payment cost, mean network throughput, task execution and resource service providers profit value, ratio regarding time, and financial reward, among others. The performance investigation and result regarding 60 users' task execution reveal that the proposed superactive scheme attains approximately 53.05% and 71.72% more mean task completion time gain than the traditional HRSDN, and traditional CSDN scheme, respectively. Whereas, the performance investigation and result also indicate that the proposed superactive scheme brings approximately 39.11% and 69.07% more financial payment cost gain than the traditional HRSDN, and traditional CSDN schemes, respectively. The future extension of this research would incorporate the congestion control and security threat prediction scheme using advanced deep learning/artificial intelligence techniques along with the convergence of SDN and network function virtualisation (NFV) technology for both physical and virtual resource assignment.

References

Abugabah, A. et al. (2020) 'Intelligent traffic engineering in software-defined vehicular networking based on multi-path routing', in *IEEE Access*, Vol. 8, pp.62334–62342.

Akin, E. et al. (2019) 'Comparison of routing algorithms with static and dynamic link cost in software defined networking (SDN)', in *IEEE Access*, Vol. 7, pp.148629–148644.

Alfoudi, A. et al. (2019) 'An efficient resource management mechanism for network slicing in a LTE network', in *IEEE Access*, Vol. 7, pp.89441–89457.

Ali Shah, S. et al. (2020) 'Software-defined vehicular networks: a cooperative approach for computational offloading', *30th International Telecommunication Networks and Applications Conference (ITNAC)*, pp.1–3.

Amreen et al. (2017) 'Performance evaluation in cloud computing model using queuing models', *International Journal of Grid and Distributed Computing*, Vol. 10, No. 3, pp.15–24.

Azaly, N. et al. (2020) 'Centralized dynamic channel reservation mechanism via SDN for CR networks spectrum allocation', in *IEEE Access*, Vol. 8, pp.192493–192505.

Bano, M. et al. (2021) 'Soft-mesh: a robust routing architecture for hybrid SDN and wireless mesh networks, *in IEEE Access*, Vol. 9, pp.87715–87730.

Bektas, C. et al. (2018) 'Towards 5G: an empirical evaluation of software-defined end-to-end network slicing', *IEEE Globecom Workshops*, pp.1–6.

Bi, Y. et al. (2019) 'Software defined space-terrestrial integrated networks: architecture, challenges, and solutions', in *IEEE Network*, Vol. 33, No. 1, pp.22–28.

Cao, B. et al. (2021) 'Resource allocation in 5G IoV architecture based on SDN and fog-cloud computing', in *IEEE Transactions on Intelligent Transportation Systems*, Vol. 22, No. 6, pp.3832–3840.

Chen, L. et al. (2020) 'Multi-task mapping and resource allocation mechanism in software defined sensor networks', *Int. Conf. on Wireless Communications and Signal Processing (WCSP)*, pp.32–37.

Chen, M. et al. (2021) 'Dynamic load balancing strategy based on link preference in SDN', *7th International Conference on Computer and Communications (ICCC)*, pp.1920–1924.

Cui, H. et al. (2022) 'Space-air-ground integrated network (SAGIN) for 6G: requirements, architecture and challenges', in *China Communications*, Vol. 19, No. 2, pp.90–108.

Doan, T.V. et al. (2020) 'FAST: flexible and low-latency state transfer in mobile edge computing', *IEEE Global Communications Conference*, pp.1–6.

Drolia, U. et al. (2017) 'Cachier: edge-caching for recognition applications', *IEEE ICDCS*, pp.276–286.

Fang, W. et al. (2022) 'Towards energy-efficient and secure data transmission in AI-enabled software defined industrial networks', in *IEEE Transactions on Industrial Informatics*, Vol. 18, No. 6, pp.4265–4274.

Gao, Y. et al. (2020) 'Cluster-based interference-free MAC protocol with load aware in software defined VANET', in *China Communications*, Vol. 17, No. 12, pp.217–234.

Huang, C. et al. (2018) 'V2V data offloading for cellular network based on the software defined network (SDN) inside mobile edge computing (MEC) architecture', in *IEEE Access*, Vol. 6, pp.17741–17755.

Jia, Z. et al. (2021) 'Joint HAP access and LEO satellite backhaul in 6G: matching game-based approaches', in *IEEE Journal on Selected Areas in Communications*, Vol. 39, No. 4, pp.1147–1159.

Li, M. et al. (2017) 'Random access and virtual resource allocation in software-defined cellular networks with machine-to-machine communications', in *IEEE Transactions on Vehicular Technology*, July, Vol. 66, No. 7, pp.6399–6414.

Li, G. et al. (2020a) 'SDN based computation offloading for industrial internet of things', in *16th International Conference on Mobility, Sensing and Networking (MSN)*, pp.402–409.

Li, J. et al. (2020b) 'A secured framework for SDN-based edge computing in IoT-enabled healthcare system', in *IEEE Access*, Vol. 8, pp.135479–135490.

Ling, T. et al. (2019) 'An optimized forward scheduling algorithm in a software-defined satellite network', *17th Int. Conf. on Software Engineering Research, Management and Applications (SERA)*, pp.27–32.

Liu, X. et al. (2015) 'Trends in PON – fiber/wireless convergence and software-defined transmission and networking', *OECC Conference*, pp.1–3.

Marotta, A. (2019) 'Exploiting flexible functional split in converged software defined access networks', in *Journal of Optical Communications and Networking*, Vol. 11, No. 11, pp.536–546.

Montazerolghaem, A. et al. (2020) 'Load-balanced and QoS-aware software-defined internet of things', in *IEEE IoT Journal*, Vol. 7, No. 4, pp.3323–3337.

Nazari, S. et al. (2016) 'Software defined naval network for satellite communications (SDN-SAT)', *IEEE Military Communications*, pp.360–366.

Nithya, S. et al. (2020) 'SDCF: a software-defined cyber foraging framework for CloudLet environment', in *IEEE Transactions on Network and Service Management*, Vol. 17, No. 4, pp.2423–2435.

Nkenyereye, L. et al. (2021) 'Efficient RSU selection scheme for fog-based vehicular software-defined network', in *IEEE Transactions on Vehicular Technology*, Vol. 70, No. 11, pp.12126–12141.

Nyanteh, A. et al. (2021) 'CloudSimHypervisor: modeling and simulating network slicing in software-defined cloud networks', in *IEEE Access*, Vol. 9, pp.72484–72498.

Sheng, M. et al. (2017) 'Toward a flexible and reconfigurable broadband satellite network: resource management architecture and strategies', *IEEE Wireless Commun.*, Vol. 24, No. 4, pp.127–133.

Sultana, A. et al. (2019) 'Two-tier architecture for spectrum auction in SDN-enabled cloud radio access network', in *IEEE Transactions on Vehicular Technology*, Vol. 68, No. 9, pp.9191–9204.

Tadros, C. et al. (2020) 'Software defined network-based management for enhanced 5G network services', in *IEEE Access*, Vol. 8, pp.53997–54008.

Wan, J. et al. (2020) 'Cross-network fusion and scheduling for heterogeneous networks in smart factory', in *IEEE Transactions on Industrial Informatics*, Vol. 16, No. 9, pp.6059–6068.

Wang, S. et al. (2022) 'A multi-task learning-based network traffic prediction approach for SDN-enabled industrial internet of things', in *IEEE Transactions on Industrial Informatics*, DOI: 10.1109/TII.2022.3141743.

Xia, B. et al. (2020) 'A delay-tolerant data transmission scheme for internet of vehicles based on software defined cloud-fog networks', in *IEEE Access*, Vol. 8, pp.65911–65922.

Yang, D. et al. (2021) 'Dynamic resource allocation of network function virtualization for space-air-ground integrated energy internet', *International Conference on Power System Technology (POWERCON)*, pp.1897–1903.

Younus, M. et al. (2022) 'Improving the software-defined wireless sensor networks routing performance using reinforcement learning', in *IEEE IoT Journal*, Vol. 9, No. 5, pp.3495–3508.

Yuan, X. et al. (2020a) 'Edge-enabled WBANs for efficient QoS provisioning healthcare monitoring: a two-stage potential game-based computation offloading strategy', in *IEEE Access*, Vol. 8, pp.92718–92730.

Yuan, X. et al. (2020b) 'An A3C-based joint optimization offloading and migration algorithm for SD-WBANs', *IEEE Globecom Workshops*, pp.1–6.

Zhang, H. et al. (2015a) 'Toward vehicle-assisted cloud computing for smartphones', *IEEE Transactions on Vehicular Technology*, Vol. 64, No. 12, pp.5610–5618.

Zhang, W. et al. (2015b) 'Collaborative task execution in mobile cloud computing under a stochastic wireless channel', *IEEE Transactions on Wireless Communications*, Vol. 14, No. 1, pp.81–93.

Zhang, N. et al. (2017) 'Software defined space-air-ground integrated vehicular networks: challenges and solutions', in *IEEE Commun. Mag.*, July, Vol. 55, No. 7, pp.101–109.

Zhang, C. et al. (2020a) 'Global load-balancing ONU assignment for a software-defined reconfigurable PON', in *Journal of Optical Communications and Networking*, Vol. 12, No. 7, pp.177–191.

Zhang, H. et al. (2020b) 'V2X offloading and resource allocation in SDN-assisted MEC-based vehicular networks', in *China Communications*, Vol. 17, No. 5, pp.266–283.

Zhang, J. et al. (2020c) 'Task offloading in vehicular edge computing networks: a load-balancing solution', in *IEEE Transactions on Vehicular Technology*, Vol. 69, No. 2, pp.2092–2104.

Zhang, Y. et al. (2022) 'Performance evaluation of software-defined network (SDN) controllers using Dijkstra's algorithm', *Wireless Netw.*, pp.1–14, Springer [online] https://doi.org/10.1007/s11276-022-03044-3.

Zhou, S. et al. (2019) 'Bidirectional mission offloading for agile space-air-ground integrated networks', in *IEEE Wireless Communications*, Vol. 26, No. 2, pp.38–45.

Zhu, X. et al. (2017) 'Resource allocation in spectrum-sharing cloud based integrated terrestrial-satellite network', *13th IWCMC Conf.*, Spain, pp.334–339.