Edge Computing

Subjects: Computer Science, Information Systems
Contributor: Nour Alhuda Sulieman

Edge computing is a distributed computing paradigm such that client data are processed at the periphery of the network, as close as possible to the originating source.

Keywords: edge computing ; cloud computing ; industrial internet of things

1. Optimal Placement of Servers in Mobile Edge Computing

Research has recently started giving attention to the challenge of performing *mobile edge computing* [1] far from the main central system architecture, as well as of mitigating latency to messages and transmission response times through the network.

Li and Wang in [2] study the problem of edge servers' placement and propose an advanced algorithm in which the locations of the edge servers are optimized in such a way as to improve the utilization of the installed servers, in addition to reducing the number of servers which work in the idle mode. In order to solve this problem, the authors assume that the mobile edge network behaves as an undirected graph $\mathscr{G} = (\mathscr{V}, \mathscr{E})$, where V represents the set of edge nodes with the potential locations of edge servers, and E accounts for the communications among them. Although the energy consumption of edge servers is due to several sources such as CPU, hard drive memory, and other hardware elements, CPU is the most consuming resource based on the studies, which is why the authors consider CPU usage as the target cost function when it comes to saving energy through the mobile edge network.

Another perspective is given in [3][4], where the authors adjust the traditional particle swarm optimization (PSO) algorithm to make it applicable in discrete optimization problems such as the above-mentioned mobile edge network scenario. They challenge the conventional concept according to which each edge node should have a corresponding edge server, and they look for an optimal solution increasing the profiteering of servers and at the same time decreasing their total number. The proposed solution defines a *one-to-many* assignment relationship between an installed server in the network, on the one hand, and edge nodes, on the other hand, and finds the best positioning solution that decreases the energy consumption of servers' CPUs while satisfying the following assumptions:

- CPU power consumption is based on the value of the server's power in both idle and full states of utilization;
- Each node should be assigned to one server, and each server could serve one or more nodes;
- Latency time should be always less than or equal to a pre-defined threshold.

The proposed algorithm for positioning edge servers has been tested on a real dataset from Shanghai Telecom where 3233 edge nodes are considered. Tests are considered two cases: in the first experiment, the number of edge nodes is fixed, while the distance threshold between nodes and servers varies; in the second experiment, the threshold is fixed, and the number of edge nodes is increased. The proposed approach shows better results when compared to traditional algorithms, yielding up to 12% less power consumption in the first experiment and up to 13% less power consumption in the second one, thus proving that the available resources of edge servers can be exploited dynamically rather than statically assigning one node to only one server. Numerical analyses are also reported in [3], demonstrating the average number of edge servers for the edge server placement with respect to the edge server coverage.

In [5], Xiao et al. proposed a heuristic algorithm that predicts a strategic place for edge servers based on resource requirement forecasting. The authors manage to determine a suitable number of possible server locations for a specific data source (mobile, bus station, PC, gaming stations, etc.) by predicting the next destination of this data source, in addition to using a data naming mechanism between a data source and the possible linked server. Markov chain modeling is used in this respect, together with an approach for detecting the optimal server location based on the following three

parameters: migration time between two servers, processing time in the new server, and required time to transmit the result to the data source. A simple Kolmogorov equation is applied for the presented Markov chain modeling in this paper, while the optimization problem of a set of servers to be placed with the requirements of each server needs is formalized through different formulas in [5]. A data naming mechanism has been used as a connection method between data sources and edge servers in order to exchange the important mapping information (needed CPU power for the task, location of the data source, time, capacity, etc.) for the overall goal of enhancing edge computing by selecting the best service provider linked to the user's requests and available resources.

In [6], an efficient analytical model for self-adaptive service migration in edge computing networks is proposed. The aim of the authors is to keep the service location, which is provided by edge nodes, as close as possible to the end user place through dynamic adaptation targeted at maximizing the efficiency of the network performance. The mean absolute percentage error is evaluated as a key performance indicator while iteratively performing the following sequence of actions.

- Monitor: this action saves useful information, allowing one to define which edge node in the network is able to provide the requested service.
- Analyze and Plan: this action allows for deciding whether the service should be migrated to another node or not, based on the distance between user mobility and the edge node that hosts the service.
- Execute: this action launches the migration process.

By using basic probability and renewal theory arguments, different closed-form expressions for the average cost per unit time experienced by a mobile device or by the system are derived and listed in [6]. The resulting optimization model, defined as the "regional planning pattern", incorporates: a distance model, evaluating the distance between each device and each service provider; a time model, which describes the events of interest (device mobility) that could affect the distance model; a cost model, accumulating the costs of a specific device each time it is evaluated to have a non-optimal distance; and a control model, according to which all non-optimal distances of devices are reported to a centralized planning component. Such a regional planning pattern is expected to have positive effects in the field since it presents less data traffic throughout any edge computing network arranged according to a fully distributed architecture.

2. Security of Data Transferred through Edge Networks

Another relevant emerging research topic is related to the security of data transferred over edge networks.

In [7], the security of the data shared across different domains using edge computing technology is investigated. The authors consider each edge node and its related equipment as a domain, where the centralized cloud connects all the domains of edge nodes with each other through the backhaul network. Four main components are considered in this model: the central cloud, which is the shared pool of all computational resources; edge servers, which are placed closer to end users' locations; the data owner, who can encrypt a message to all the users; and the data user, who receives messages from the owner. Entities in the proposed paradigm follow the traditional system architecture of edge computing, noting that both the data user and data owner are located in the first level (edge devices). To ensure the security of shared data between different edge domains, the RSA algorithm and CP-ABE are used. For example, the data owner of a specific edge domain (A) defines its privacy policy and attribute list with other domains and creates public and private keys. It encrypts and sends its public key with the request to edge domain (A) that it wants to share data with domain (B). Edge domain (A) validates the source information and sends the sharing request and the related public key to the cloud service provider. The latter finds the target domain (B) and forwards it the request (A wants to share data with B), in addition to the public key of (A). When (B) receives the request, it also uses its own attributes to generate security keys. It uses symmetric encryption to encrypt the previous attributes, and it uses the public key of (A) to encrypt this symmetric key. In this way, domain (A) can use its key to decrypt the attribute list of (B) and obtain its public key in order to be able to decrypt the shared data. As a result, only domains that satisfy the privacy policy can access the data and decrypt the shared messages, which proves the efficiency of the proposed model.

As part of the increasing focus on security issues in edge computing paradigms, a survey has been conducted in [8] to highlight the main security vulnerabilities and possible attacks while using virtualization technologies in the edge/fog computing paradigm. The authors analyze Unikernel, real-time operating systems, and containerization technologies and approaches while they were deployed on real edge computing use cases (smart cities, smart home, e-health, etc.), and then they depict the level of security impacts caused by different possible attacks. In their analysis, they defined five categories of vulnerabilities: vulnerabilities directly linked to the docker; vulnerabilities inside images; vulnerabilities of the

Linux kernel; insecure production system configurations; and vulnerabilities in the image distribution, verification, decompression, and storage process. Many possible attacks may occur when docker technology is applied such as remote code execution, MAC flooding, DoS on images, data leakage, ARP spoofing, Zip-bomb-like attacks, changing image behavior, viruses, privilege escalation, container escape attacks, and others. On the other hand, several attacks have been detected when using Unikernel such as privilege escalation, attacks related to hypervisor vulnerabilities, and DoS attacks. The results proposed in this work show that Unikernel has a security isolation advantage over containers, while the latter have setup facility as a distinguishing advantage. Real-time operating systems have good performance in many applications, but they suffer from update difficulties, and they can be adopted only in such cases on constrained devices. One major point regards the level of security impact caused by the previously mentioned possible attacks. In other words, taking an e-health system as an example, attacks perpetrated against a specific system could be the reason for a person's death, if compared to other systems where the same type of attack has a lower impact. As a result of the analyses, strategic guidelines have been proposed at the end of the work in order to reduce the risks of existing vulnerabilities. Unikernel could provide less attack risk in cloud gaming applications and better code integrity, while in smart cities, real-time operating systems could be a more feasible option.

In [9], Alrowaily and Lu introduce the main security concepts that should be taken into consideration in the safeguarding of edge computing networks. Keeping users' personal data safe and secure is one of the most important challenges emerging with the high growth of edge computing networks since all components communicate with each other and exchange significant amounts of data. This paper summarizes what should be considered in order to manage edge network privacy with respect to the following concepts: pseudonyms, unobservability, unlinkability, and anonymity. However, with the ever-rising amount of transferred data through the network, there is always a need to check that the functionality concepts which were mentioned before are working in the proper way. Therefore, four evaluating components are presented in this paper, and by ensuring their investigation, network security is guaranteed: (1) confidentiality, (2) integrity, (3) availability, and (4) access control and authentication. A thorough discussion is provided with respect to using these four components with the aim of reducing the relevant risks that may occur. For instance, multiple security algorithms are proven to yield superior results in many IoT applications that extensively rely on an edge computing networks.

3. Distributed Edge Computing Platforms and the Relevance of Hybrid Edge-Cloud Computing

An innovative computing platform by the name of DNR (Distributed Node Red) has been proposed in [10] as an extension to the well-known Node Red open source tool, which can provide a visual data flow for IoT applications. Three versions of this platform have been already tested, each one being used to solve a specific challenge. In more detail, Node Red is a visual tool where developers can evolve their applications by dragging and dropping entities and wiring them one to another. However, Node Red usually deals with single processes and does not support distributed applications such as edge computing networks. DNR-v1 already tried to solve this problem by enabling the execution of nodes on any device within the network (edge CPU, cloud server, etc.). This was managed by adding the concept of device id to the Node Red paradigm. Hence, in DNR-v1, each node in the flow is assigned to a unique device id representing an identification for the device the node will be deployed on. Launching the ability of allowing nodes to be executed on different devices cast a new challenge related to break-up cases in the data flow. For this reason, the two concepts of wire-in and wire-out were added. Nodes that cannot be run on the assigned device id are replaced with remote wires using a publish/subscribe mechanism. Wire-in nodes use the subscribe technique to receive data from another node while wire-out nodes receive data from another node and publish it in a communication broker so that the data are received by other parties. Then, DNR-v2 was released, with two main targets: enabling more complex and larger geo-distributed applications to be applied using this platform and executing nodes on multiple devices in parallel. In order to meet the previous requirements, a primary restriction parameter is added to each node in the flow in which not only is the type of device where the process should be implemented defined, but it is also allowed to choose the device based on other parameters such as memory size, physical location, CPU power, and others. Two other notions have been added to face the previous challenges: the first notion is the wire cardinality to solve the problem of parallel execution by allowing hosting instances of the same node by different devices, and the second notion is the wire fragmentation which is useful when one or more instances are accessible while all data sources contact the same one. Later, DNR-v3 introduced a communication layer to the previous platform, which is able to coordinate the connection with external software components to achieve the application objectives. This is implemented by adding new coordination nodes to the existing wires in the paradigm in addition to placing a centralized coordinator capable of receiving all control messages and passing them to the coordination nodes, these latter taking the responsibility of the execution process to either pull or push data from other software components, and updating the status of coordination nodes after finishing the process execution. DNR experiments devised new

solutions for having an exogenous platform in order to increase the computation efficiency through edge networks. Recent technological openness imposes a harder effort in the research field in order to obtain the best integrated technological solutions in order to increase the application efficiency. It is clear from the evolution of the DNR platform that the distributed platform of edge computing is an attractive topic in research, especially in that it can be integrated with containerization technology, which relevantly simplifies the development and execution of distributed applications.

In [11], a theoretical model is proposed with the aim of deciding when an edge-only or hybrid edge-cloud set up is to be used and also when it is better to rely on traditional cloud architecture. The proposed model is mainly based on the following parameters: selectivity ratio, computation-communication ratio, and cloud processing speed-up. As intuition suggests, the paper proves that when the cloud speed-up parameter is low and the computation-to-communication ratio is small, edge-only systems are faster than cloud-only ones. When comparing hybrid edge-cloud computing with cloud-only systems, it emerges that the hybrid setup is faster when the relevant hybrid edge-cloud speed-up parameter is greater than one. The authors analyze the performance of the previous three system architectures on a specific framework where two MapReduce functions are run on a hybrid system architecture, the former on the edge and the latter in the cloud. The performance is then tested on seven real applications, three of which do not support the hybrid execution. AWS is chosen as the relevant cloud-computing technology for this experiment, while two low-power devices are used to represent edge components. Only one application exposes a faster execution time when bandwidth is low in cloud computing compared to edge-only systems, while for large bandwidth, all the other applications have faster processing time on edge-only systems compared to cloud-only systems. The cloud-only setup is always faster when the application has low-data movement, whereas the hybrid edge-cloud system is always faster for large inputs and a small intermediate data size. All in all, the proposed model provides a useful framework for deciding whether to execute tasks using the traditional scenario of cloud computing or move the workload to edge-only or hybrid edge-cloud.

In [12], the main challenges in terms of application development relying on edge computing technology are investigated, and an efficient solution is proposed using containerization technology. Implementing an application on edge networks needs the satisfaction of some specific hardware requirements in terms of network connectivity, CPU power, and capacity that are significantly higher than in the case of applications running on normal desktop computers. In particular, the dependency of edge computing applications on obtaining real-time data from other vendors' applications poses a new time consumption challenge in terms of data suppliers when the latter have problems in their data sources. Additionally, there emerges the repetitive process after any change is made in the application, which consists of restarting the life cycle of the containerization process from scratch and sending the new version of the code to edge devices. The authors propose a remote debugging method which creates a docker container with all required programming libraries and packages being included in the remote edge node. This scenario enables application updates at the edge device without any need to create new containers each time a change happens, thanks to the presence of a secure copy of the application inside the remote debugging container. This proposal was tested in a production area for a CNC (computer numeric control) machine, and it showed better performance in terms of time consumption in addition to security proof while moving data among containers. Data migration happened by using the Docker Compose tool. The proposed approach showed positive results in C++ applications, and it is expected to yield the same favorable effect for applications in other programming languages, thus affecting the efficiency of the increasing number of edge computing applications. The advantages of edge computing technology increasingly prove its effectiveness and worthiness in many applications and show the capability to answer users' requests fluently and locally using the installed servers that are placed physically closer to end users. These advantages make edge technology one of the best choices for some critical applications which need real-time actions such as tasks with hard deadlines in energy and manufacturing plants.

Moreover, relative to the field of medical emergency services, an application of video-based heart rate detection on an edge computing platform is proposed in [13], consisting of four entities: a smartphone (Huawei Honor 8), a base station (according to the open source project of the 5G SoftRAN cellular system), an edge server, and a cloud server. The application test starts from sending facial videos from a user's smart phone to the edge server where data preprocessing happens, while other features are sent to the cloud for more complex analysis where cloud servers take this responsibility. Eventually, the combined result returns a reliable heart rate measurement to the smartphone. Experimental results emphasized that the response time of this hybrid edge-cloud architecture is 20% and 40% less than in the case of using edge technology and cloud technology only, respectively. This is a sign that despite many of the current expectations that edge computing will replace cloud technology, practical experiments are continuously showing that the best performance can be achieved through their combination.

Another interesting practical experiment is proposed in [14]. This experiment runs a neural network on an edge computing platform with the support of VPU (vision processing unit); it was shown how it woks when using devices from different operating systems and with different powers. An RPi3B was used as an edge device. It has a 1.2 GHz 64bit quad-core

Cortex-A53 (ARMv8) CPU, four USB 2.0 ports via an on-board five-port USB hub, and 1 GB low-power DDR2 SDRAM, and it draws a maximum of 6.7 W at peak load. However, an NCS (Neural Compute Stick) was used as the deep learning device of the experiment powered by the same powerful and performance features of VPU. Ubuntu 16.04 is installed on a physical x86 64 system, and the Debian Stretch runs on a Raspberry Pi 3 Model B. The Neural Compute SDK comes with C++ and Python (2.7/3.5) APIs. The model has been tested on the Google MobileNets neural network, and the results showed that the RPi3B is able to effectively recognize objects in real time in combination with the embedded deep learning device. On the other hand, another test has been done to compare the performance of RPi3B with the Ubuntu system when they work separately. The results showed that running a separated Ubuntu gives 9.3 frames per second with a single NCS attached, while the RPi3B gives 5.7 FPS. With two sticks, Ubuntu gives 6.6 frames per second, while the RPi3B produces 3.5 frames per second. This positive result sheds light on the ability of edge computing to be used in many critical applications such as self-driving cars or any of the recent robotics applications which do not require human engagement.

4. Simulation Platforms for Evaluating the Performance of Edge Computing

The Castnet framework has been proposed in [15] as a new simulation platform able to evaluate real edge solutions and explore new possible scenarios. The proposed framework aims to basically provide a high level of abstraction through the independent queues that are responding to the requests coming from each entity in the network. This improves response time with respect to the used shared queue by other existing simulators such as iFogSim and ns-3. iFogSim is considered as an extension of the CloudSim framework, and it provides modeling support for an edge infrastructure; it is designed to be used as a single event queue, which makes it inefficient for larger scale simulations. Likewise, ns-3 lacks native abstractions to describe any application-level functionality which is to be carried out along the network paths. Instead, Castnet is a lightweight, extendable, fast, and easy platform to use for edge computing simulations. The high level of abstraction lets Castnet support all entities in an edge computing scenario with different edge infrastructure deployment models and characteristics (latency, computation, and storage). The Castnet abstraction is an incorporation of five levels which form the simulator input:

- Local computational power;
- Edge infrastructure;
- Edge functionality;
- Edge nodes placement;
- Links between nodes in addition to a workload engine able to define the pattern of the incoming request.

Each created request through the network is linked with a timestamp and handled to elements with a processing time that is less than or similar to its timestamp: this distribution helps exploit the limited resources of the network in the best way. The Castnet framework has been implemented in C/C++ using boost and nlohmann json external libraries, while Python has been used for generating the synthetic workload. The first evaluation test of Castnet was simple; it considered an edge local server as a proxy, receiving requests and forwarding them to another node/server for processing. The results of this test showed similar results between the three simulators Castnet, iFogSim, and ns-3. As a second validation experiment, the authors tested the functionality of the edge caching capacity and its effect on the response time; the test was done on three levels: non-configured cache and 50% and 100% cache capacity. The obtained results showed the following measures for responding time in milliseconds for the three simulators Castnet, iFogSim, and ns-3 in sequence: 5.00, 5.01, and 5.43. As the maximum taken time to handle the request, 95.05, 112.02, and 110.9 were registered for the three simulators, consequently, while for the average needed time, the following results were registered in sequence: 13.63, 17.80, and 16.32.

Another relevant comparison in [15] is related to the number of code lines which is needed to implement an edge application in Castnet compared to iFogSim and ns-3: 188, 1199, and 714, respectively. The scalability evaluation also has been compared between Castnet and ns-3 while increasing the number of edge nodes from 10 through 100 to 500; iFogSim was not considered in this test because it needs a much longer time. The comparison results proved that ns-3 has a 17.31–44.83% longer time. From all previous experiments, it can be confirmed that Castnet outperforms the other two simulators even with changing conditions. This poses this new platform as an easy, flexible, and extendable platform for edge computing applications. However, even with all the benefits and flexibility features in the proposed Castnet infrastructure, it is still has some limitations. For instance, it provides a means to address tasks and operations that should

be done on the edge, but it is not able to do the automatic division of the application logic between the edge and cloud. Recent research has started to focus on the ability of obtaining benefits from the available resources of edge nodes in the best way in order to achieve the desired insights of the IoT application.

In [16], the authors propose the SaRa model as a probabilistic model to approximate the reliability of edge resources in a volunteer cloud (e.g., cuCloud). It is considered to be one of the recent volunteer solutions of cloud computing where no data center exists, while it relies on multiple dedicated centers. The proposed model assigned a reliability factor to each node in order to represent the trustiness level of this node in providing high quality services. This factor is calculated from two outcomes: the success of VM in satisfying QoS and the failure in terms unsatisfied QoS. The observed results were classified into different classes based on many features such as latency and priority in order to be used later when defining trusted and untrusted nodes. A Google cluster usage trace has been used to test the approach with 100 physical machines, where their highest failure rate was in the range of [5–117] patterns during a life cycle of 29 days. The SaRa model only used the first 26 days of trace data, while the failure rate of the last three days was calculated to validate the estimation accuracy between each compared method. The correlation coefficient has been used to compare the traditional methods (Traditional Reputation, RD Reputation [17], and RBCC [18]) with the newly proposed SaRa model based on the obtained reliability values of the first 26 days by SaRa and the failure rate of the last three days. By considering four different classes of the failure rate, SaRa shows in most of the cases better values of the correlation coefficient compared to the other approaches. The most important achieved feature by this model in distinguishing trusted from untrusted nodes has a great impact on the platform itself since it gives high flexibility to its nodes in joining and leaving the model. However, some other crucial issues should be addressed in order to improve the model performance, such as fault tolerance and the distrust volunteer nodes.

5. Advances for the Performance Improvement of Edge Networks

With the rapid integration between IoT and the human physical world, the SDEC-based open edge computing system architecture has been proposed in [19] as an innovative approach to increase the possibility of having a novel edge-computing framework able to utilize the available resources in the best way. The proposed approach uses pooling, virtualization, and abstraction technologies to split software and hardware layers in the traditional edge-computing scenario, which helps cache the complexity of the hardware layer and manage all edge resources and services by the software layer. SDEC is divided into five main parts: SDED, SDESto, SDECR, SDESer, and the SDEC controller.

- SDED virtualizes and abstracts edge physical devices, so it can be considered as the digital copy of the physical devices in the edge network.
- SDESto is a very important model in the proposed infrastructure because it helps manage edge storage capacity and use it in the best manner. This happens by abstracting and pooling edge storage in a virtualized pool, and then fast dynamic mapping can be done between edge storage and application requirements.
- SDECR is responsible for classifying and abstracting all computing devices in the edge network (CPU, RAM, AI chip, etc.), which eases finding the device that matches the computing requirements of a specific application.
- SDEC controller is a central managing and controlling unit where an intelligent managing of the edge resources happens and guarantees edge system flexibility.
- SDESer is the software copy of the edge services' functionality; it abstracts all services in a virtualized pool and enables sharing the physical hardware by different edge services.

Finally, researchers can summarize SDEC-based open edge computing system architecture into four main parts: the edge devices, the local network that connects edge devices with each other and with the SDEC platform, the SDEC platform, and the smart edge application which is directed to end users. This approach provides an extendable, automatic, and more flexible platform compared to the traditional edge computing ones, and it enables the management of all edge resources in the best way in order to improve the performance and reduce energy consumption. However, the need for an updated scenario targeting the software models and updating their characteristics automatically is one of the main challenges of this approach since the manual updates for these software definition models are almost impossible due to the number and type of objects involved in the SDEC paradigm.

An innovative scheme has been proposed in [20], where the tasks for processing data are divided between edge devices and fog nodes, aiming to decrease the trained data samples and communication cost. This scheme creates the initial training model by applying federated learning in the middleware fog devices using a specific number of data samples.

Afterwards, the centralized fog node publishes the model to all edge devices where active learning is applied locally and separately using the maximal entropy function. As a result, new models are sent back to the fog node from the edge, noting that the number of these models is equal to the number of edge devices. Then, weights aggregation is done by choosing the best training model or by taking their average, which is considered as the input for the next round. This new method retains users' personal privacy for further data analysis processes by applying the active learning locally on the edge and decreasing communication costs thanks to federation learning, which is usually applied on the fog centralized node. A CNN (convolutional neural network) is used as a model for the above-mentioned training purposes, while all experimental methods are implemented in Python on the MNIST dataset, which counts 6000 handwritten images representing the training dataset and 1000 handwritten images representing the test dataset. Their experiments were applied on two levels: the first concerns the overall performance when applying active learning on edge devices compared to choosing data randomly, whereas the second experiment is aimed at demonstrating the overall performance while increasing the number of training data. The obtained results show that the proposed approach outperforms the centralized one by decreasing latency and reducing labeling and communication costs, possibly yielding privacy benefits. With the important potential benefits of this proposal, new efforts should be oriented to increase the number of edge devices used within the network and to study how the viable security risks could affect the overall performance of the system.

In the context of facing bandwidth and latency challenges in the existing implementations of web augmented reality (AR), a MEC-based web AR solution has been proposed in [21]. System components can be summarized as the following: the MEC platform is used to process the incoming requests and deploy web AR applications, while an abstract service layer for the web AR with AR cache forms the running environment of the web AR. However, the Docker technology has been used to dynamically schedule the diversity of web AR applications. Experiments were done by using a MI MIX 2 mobile phone 10 times, where frame sizes were 250 × 250 pixels, weighing on average 25.04 KB. The proposed architecture was compared with two existing solutions: cloud computing solutions and pure front-end solutions. The comparisons regarded three levels: latency, FPS, and power consumption. The obtained results show better performance in the new approach compared to other conventional solutions. However, web AR applications still need to balance benefits and costs, in addition to acquiring a higher efficiency in terms of sustainability issues.

In [22], the authors proposed an innovative architecture for edge computing in which WiFi routers are located close to edge devices, and they organize how edge nodes collaborate in catching and processing participatory sensing data. The presented system architecture in this work consists of a cloud-based main coordination server which sends the requirements and task directives to WiFi routers geographically distributed within the network. Each router distributes the execution of a set of micro-services between the nodes which are located in its area. At the end, edge devices execute the assigned tasks by downloading the needed execution packages from the cloud repository and later return the output to the router. An indoor navigation map use case has been used to evaluate the previous system architecture in which navigation output results are provided in pairs in a form (name of room's occupant, room number). As a result, a participatory sensing task is created to collect this information, which is basically a scenario of image recognition. Five main phases have been considered in the previous approach to execute the micro-service:

- Taking photos;
- Image content recognition;
- Verification of recognized data;
- Recognition model update;
- Distribution to participants' devices.

They evaluated the energy consumption carried out by the new architecture for multiple micro-services, in which energy consumption was registered as 949
Ah for taking photos, 80 Ah for image pre-processing, 23 Ah for data recognition, and 522

Ah for verifying photos. Evaluation results of this system architecture show that it is able to improve the performance of the network and to eliminate the possibility of network bottleneck when huge amounts of data are transferred or when researchers have mobile devices with a limited battery. This causes a reduction in network traffic, while achieving more accurate sensing results. From the initial results of this proposal, researchers think that it is going to be a promising solution, especially when dealing with rich data such as photos and videos.

Moreover, in [23], the authors propose and test a resource management technique which has two main objectives: task distribution between edge nodes, which helps save energy and ease privacy constraints by enabling each device in the network to follow its own privacy roles and its availability. Based on this proposal, the main complete objective of the IoT application is divided into many sub-tasks where each sub-task can be executed on a different node. This increases the feasibility and practicality of the application. In order to cover the dependencies between tasks, the application is represented as a directed acyclic graph, where the nodes are the application tasks, while the edges represent the dependencies between them. However, the resource management technical platform follows an auction house functionality and has two main models: the policy module and the bidding policy module. When the application is ready for execution, bidding nodes are created where each node has its own available resources. Bidding nodes connect to the main deployment node with a communication link named the dispatcher. The responsibility of the policy module can be summarized in mapping the objective of a specific sub-task with offers provided by bidding nodes, while keeping the end-to-end latency as short as possible in choosing the best offers. Additionally, the bidding policy module works to choose the best candidate offer based on the privacy requirements built by the node itself. Their approach is tested on a montage graph from the real word and run on a single core Intel i5 2.3 GHz, while changing the range of divided sub-tasks from 24 to 50 and the range of resource constraints between 1 and 10. Testing results show efficiency at finding the best candidate nodes that minimize end-to-end latency and meet the privacy requirement of application objectives.

6. Edge Computing over 5G/6G Networks

In [24], the authors attempt to show the possibility of integrating IoT networks, 5G, and cloud computing by conducting a research test-bed, and then they evaluate the system performance by running a mission-critical time-sensitive application on an edge cloud platform. A 5G wireless network is one of the most important components of the system; it was implemented using a LuMaMi test-bed, while many computing nodes were connected and joined by this network. Four different types of nodes were used to carry out the computation tasks in this network: a Raspberry Pi 3Bs for representing the plant (mechanical device which is continuously performing a specific task), an Intel Core i7 desktop for representing edge nodes, an Intel Core i7 VM for representing ERDC, and an Intel Xeon VM for representing AWS. The two VMs of AWS and ERDC were connected to the subnet over a VPN, which facilitated the direct access between computing nodes. However, Calvin was used to represent the cloud platform. For evaluating the system performance, the ball and beam was the control process of the test-bed research, and it was considered as a time-sensitive mission-critical application. They tested system characteristics, overall performance, and the controller’s ability to continuously work successfully. Evaluation results show that the native control loops which have been applied on the cloud platform can work successfully on the edge cloud infrastructure with the possibility of changing the controller location without losing the benefit provided by edge or cloud nodes or without affecting system stability. Some improvements related to scheduling, synchronizations, and passing messages will have a notable increase in the targeted applications by having high adaptive, user-friendly, low jitter, and low latency applications.

In [25], an innovative architecture aimed at providing on-demand services in terms of communication, computation, and caching in 3D space anytime and anywhere is proposed, relying on tight integration between conventional ground base stations and flying nodes. This is particularly relevant for 6G, since the sixth generation standard will exploit terrestrial and non-terrestrial (e.g., satellite and aerial) platforms jointly, with the aim of improving radio access capability and unlocking the capability of offering cloud services in 3D space on demand, namely through the incorporation of mobile edge computing (MEC) functions on non-terrestrial platforms (e.g., aerial vehicles and low-orbit satellites). This will allow the extension of MEC support to devices and network elements in the sky, thus forging a space-borne MEC that enables artificial-intelligence-driven, customized, and distributed on-demand services. In such a way, the end users eventually experience the impression of being surrounded by a distributed computer, capable of fulfilling all requests with nearly zero latency. For this to happen, it is of paramount importance that resources are jointly orchestrated by relying on artificial-intelligence-based algorithms, as well as by exploiting virtualized network functions that are dynamically deployed in a distributed way across terrestrial and non-terrestrial nodes.

References

1. Bilal, K.; Khalid, O.; Erbad, A.; Khan, S.U. Potentials, trends, and prospects in edge technologies: Fog, cloudlet, mobile edge, and micro data centers. Comput. Netw. 2018, 130, 94–120.
2. Li, Y.; Wang, S. An energy-aware edge server placement algorithm in mobile edge computing. In Proceedings of the 2018 IEEE International Conference on Edge Computing (EDGE), San Francisco, CA, USA, 2–7 July 2018; pp. 66–73.

3. Abbas, N.; Zhang, Y.; Taherkordi, A.; Skeie, T. Mobile edge computing: A survey. IEEE Internet Things J. 2017, 5, 450–465.
4. Maia, A.M.; Ghamri-Doudane, Y.; Vieira, D.; de Castro, M.F. Optimized placement of scalable iot services in edge computing. In Proceedings of the 2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM), Washington, DC, USA, 8–12 April 2019; pp. 189–197.
5. Xiao, K.; Gao, Z.; Wang, Q.; Yang, Y. A heuristic algorithm based on resource requirements forecasting for server placement in edge computing. In Proceedings of the 2018 IEEE/ACM Symposium on Edge Computing (SEC), Bellevue, WA, USA, 25–27 October 2018; pp. 354–355.
6. Personè, V.D.N.; Grassi, V. Architectural issues for self-adaptive service migration management in mobile edge computing scenarios. In Proceedings of the 2019 IEEE International Conference on Edge Computing (EDGE), Milan, Italy, 8–13 July 2019; pp. 27–29.
7. Fan, K.; Pan, Q.; Wang, J.; Liu, T.; Li, H.; Yang, Y. Cross-domain based data sharing scheme in cooperative edge computing. In Proceedings of the 2018 IEEE International Conference on Edge Computing (EDGE), San Francisco, CA, USA, 2–7 July 2018; pp. 87–92.
8. Caprolu, M.; Di Pietro, R.; Lombardi, F.; Raponi, S. Edge computing perspectives: Architectures, technologies, and open security issues. In Proceedings of the 2019 IEEE International Conference on Edge Computing (EDGE), Milan, Italy, 8–13 July 2019; pp. 116–123.
9. Alrowaily, M.; Lu, Z. Secure edge computing in iot systems: Review and case studies. In Proceedings of the 2018 IEEE/ACM Symposium on Edge Computing (SEC), Bellevue, WA, USA, 25–27 October 2018; pp. 440–444.
10. Giang, N.K.; Lea, R.; Blackstock, M.; Leung, V.C. Fog at the edge: Experiences building an edge computing platform. In Proceedings of the 2018 IEEE International Conference on Edge Computing (EDGE), San Francisco, CA, USA, 2–7 July 2018; pp. 9–16.
11. Loghin, D.; Ramapantulu, L.; Teo, Y.M. Towards analyzing the performance of hybrid edge-cloud processing. In Proceedings of the 2019 IEEE International Conference on Edge Computing (EDGE), Milan, Italy, 8–13 July 2019; pp. 87–94.
12. Ozcan, M.O.; Odaci, F.; Ari, I. Remote Debugging for Containerized Applications in Edge Computing Environments. In Proceedings of the 2019 IEEE International Conference on Edge Computing (EDGE), Milan, Italy, 8–13 July 2019; pp. 30–32.
13. Li, X.; Ding, R.; Liu, X.; Yan, W.; Xu, J.; Gao, H.; Zheng, X. Comec: Computation offloading for video-based heart rate detection app in mobile edge computing. In Proceedings of the 2018 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Ubiquitous Computing & Communications, Big Data & Cloud Computing, Social Computing & Networking, Sustainable Computing & Communications (ISPA/IUCC/BDCloud/SocialCom/SustainCom), Melbourne, Australia, 11–13 December 2018; pp. 1038–1039. Available online: https://ieeexplore.ieee.org/document/8672279 (accessed on 10 August 2021).
14. Hochstetler, J.; Padidela, R.; Chen, Q.; Yang, Q.; Fu, S. Embedded deep learning for vehicular edge computing. In Proceedings of the 2018 IEEE/ACM Symposium on Edge Computing (SEC), Bellevue, WA, USA, 25–27 October 2018; pp. 341–343.
15. Daga, H.; Yoon, H.; Bhardwaj, K.; Gupta, H.; Gavrilovska, A. From Back-of-the-envelope to Informed Estimation of Edge Computing Benefits in Minutes Using Castnet. In Proceedings of the 2019 IEEE International Conference on Fog Computing (ICFC), Prague, Czech Republic, 24–26 June 2019; pp. 165–174.
16. Alsenani, Y.; Crosby, G.; Velasco, T. SaRa: A stochastic model to estimate reliability of edge resources in volunteer cloud. In Proceedings of the 2018 IEEE International Conference on Edge Computing (EDGE), San Francisco, CA, USA, 2–7 July 2018; pp. 121–124.
17. Wang, X.; Yeo, C.S.; Buyya, R.; Su, J. Optimizing the makespan and reliability for workflow applications with reputation and a look-ahead genetic algorithm. Future Gener. Comput. Syst. 2011, 27, 1124–1134.
18. Celestini, A.; Lafuente, A.L.; Mayer, P.; Sebastio, S.; Tiezzi, F. Reputation-based cooperation in the clouds. In International Conference on Trust Management; Springer: Berlin/Heidelberg, Germany, 2014; pp. 213–220.
19. Hu, P.; Chen, W. Software-defined edge computing (SDEC): Principles, open system architecture and challenges. In Proceedings of the 2019 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI), Leicester, UK, 19–23 August 2019; pp. 8–16. Available online: https://ieeexplore.ieee.org/document/9060313 (accessed on 12 August 2021).
20. Qian, J.; Gochhayat, S.P.; Hansen, L.K. Distributed active learning strategies on edge computing. In Proceedings of the 2019 6th IEEE International Conference on Cyber Security and Cloud Computing (CSCloud)/2019 5th IEEE

International Conference on Edge Computing and Scalable Cloud (EdgeCom), Paris, France, 21–23 June 2019; pp. 221–226. Available online: https://ieeexplore.ieee.org/abstract/document/8854053 (accessed on 15 July 2021).

21. Ren, P.; Qiao, X.; Chen, J.; Dustdar, S. Mobile edge computing—A booster for the practical provisioning approach of web-based augmented reality. In Proceedings of the 2018 IEEE/ACM Symposium on Edge Computing (SEC), Bellevue, WA, USA, 25–27 October 2018; pp. 349–350.
22. Song, Z.; Cheng, J.; Chauhan, A.; Tilevich, E. Pushing Participatory Sensing Further to the Edge. In Proceedings of the 2019 IEEE International Conference on Edge Computing (EDGE), Milan, Italy, 8–13 July 2019; pp. 24–26.
23. Avasalcai, C.; Tsigkanos, C.; Dustdar, S. Decentralized resource auctioning for latency-sensitive edge computing. In Proceedings of the 2019 IEEE International Conference on Edge Computing (EDGE), Milan, Italy, 8–13 July 2019; pp. 72–76.
24. Skarin, P.; Tärneberg, W.; Årzen, K.E.; Kihl, M. Towards mission-critical control at the edge and over 5G. In Proceedings of the 2018 IEEE International Conference on edge Computing (EDGE), San Francisco, CA, USA, 2–7 July 2018; pp. 50–57.
25. Strinati, E.C.; Barbarossa, S.; Choi, T.; Pietrabissa, A.; Giuseppi, A.; De Santis, E.; Kim, I. 6G in the sky: On-demand intelligence at the edge of 3D networks. arXiv 2020, arXiv:2010.09463.
