sustainability

MDPI

Article

Deep Reinforcement Learning for the Management of Software-Defined Networks and Network Function Virtualization in an Edge-IoT Architecture

Ricardo S. Alonso[1,*], Inés Sittón-Candanedo [1], Roberto Casado-Vara [1], Javier Prieto [1,2] and Juan M. Corchado [1,2,3,4]

1 BISITE Research Group, University of Salamanca, Edificio Multiusos I+D+i, Calle Espejo 2, 37007 Salamanca, Spain; isittonc@usal.es (I.S.-C.); rober@usal.es (R.C.-V.); javierp@usal.es (J.P.); corchado@usal.es (J.M.C.)
2 AIR Institute, Edificio Parque Científico, Módulo 305, Paseo de Belén 11, Campus Miguel Delibes, 47011 Valladolid, Spain
3 Department of Electronics, Information and Communication, Faculty of Engineering, Osaka Institute of Technology, 5-16-1 Omiya, Asahi-ku, Osaka 535-8585, Japan
4 Pusat Komputeran dan Informatik, Universiti Malaysia Kelantan, Bachok 16300, Malaysia
* Correspondence: ralorin@usal.es

Received: 31 May 2020; Accepted: 10 July 2020; Published: 15 July 2020

Abstract: The Internet of Things (IoT) paradigm allows the interconnection of millions of sensor devices gathering information and forwarding to the Cloud, where data is stored and processed to infer knowledge and perform analysis and predictions. Cloud service providers charge users based on the computing and storage resources used in the Cloud. In this regard, Edge Computing can be used to reduce these costs. In Edge Computing scenarios, data is pre-processed and filtered in network edge before being sent to the Cloud, resulting in shorter response times and providing a certain service level even if the link between IoT devices and Cloud is interrupted. Moreover, there is a growing trend to share physical network resources and costs through Network Function Virtualization (NFV) architectures. In this sense, and related to NFV, Software-Defined Networks (SDNs) are used to reconfigure the network dynamically according to the necessities during time. For this purpose, Machine Learning mechanisms, such as Deep Reinforcement Learning techniques, can be employed to manage virtual data flows in networks. In this work, we propose the evolution of an existing Edge-IoT architecture to a new improved version in which SDN/NFV are used over the Edge-IoT capabilities. The proposed new architecture contemplates the use of Deep Reinforcement Learning techniques for the implementation of the SDN controller.

Keywords: industrial internet of things; edge computing; software defined networks; network function virtualization; deep reinforcement learning

1. Introduction

Technologies such as *Internet of Things (IoT)*, *Industrial Internet of Things* (especially robust and fault tolerant IoT devices) and *Cyber-Physical Systems* allow millions of sensor and actuator devices that interact with the context of users [1]. These data is usually managed by the Cloud, where *digital twins* represent physical entities. Data is stored in Big Data repositories and processed to extract knowledge and forecast the state of entities and context in the future. For this purpose, different artificial intelligence techniques are applied in the Cloud, including multi-agent systems [2], or Machine Learning/Deep Learning techniques [3]. IoT, IIoT and CPS paradigms are used in many different applications, such as healthcare, smart energy, smart farming, or industry 4.0, among many others [4].

There are scenarios where a solution consisting of only one IoT layer and one Cloud layer may have certain drawbacks. The dependency on the Cloud as a data store and application provider implies that an interruption in the link between the IoT layer and the Cloud layer also interrupts the service [5]. Moreover, Cloud service providers offer pricing plans based on the amount of resources that customers use during time . In this sense, the *Edge Computing* paradigm arises to reduce the cost associated with the transfer, storage and processing of data in the Cloud. In this way, data is filtered and pre-processed in the edge of the network before being sent to the Cloud, allowing the application of machine learning techniques in the same edge, obtaining shorter response times and maintaining system functionalities even during communication breaks between the IoT layer and the Cloud layer [6].

On the other hand, there is a growing trend to share physical network resources and costs by different user entities through *Network Function Virtualization* (NFV) [7]. To reduce the maintenance costs of these physical resources and make the configuration of the network more flexible over time, *Software-Defined Networks (SDN)* techniques are also used, which are closely related to NFV [8]. Thanks to SDN, it is possible to use general purpose COTS (Commercial Off-The-Self) elements that can be reconfigured over time according to the changing needs of the network, instead of deploying specific elements (e.g., switches, bridges or routers) that have to be replaced when the network grows or changes [9].

In this sense, intelligent mechanisms are needed to efficiently optimize virtual data flows in networks according to the *Quality of service* (QoS) required by each application. *Deep Reinforcement Learning* is one of the most promising trends in recent years in the field of machine learning aimed at control [10]. Among its applications are self-driving cars In this type of application, we will call as "agent" a self-driving car, chess virtual player or an Edge node in an IoT network redirecting network traffic and defending itself from multiple sources of cyber attacks. Nonetheless, the agent cannot store and contemplate all the possible scenarios in which it can be found due to the enormous number of possibilities. Thus, the agent explores its environment knowing the possible states, the probability of changing between those states, the set of actions it can perform in each of those states and the reward or penalty it will get in each case.

In this work, we propose the evolution of an existing Edge-IoT architecture to a new improved version in which SDN/NFV are used over the Edge-IoT capabilities. For this purpose, the *Global Edge Computing Architecture* (GECA) has been taken as a basis, oriented to the implementation of Edge-IoT solutions and presented by Sittón-Candanedo et al. [11]. The Global Edge Computing Architecture has already been applied, in fact, in Smart Energy [12] and Smart Farming [6] scenarios where it was necessary to reduce data transfer between the IoT and the Cloud and, therefore, the costs of computing and storage in the Cloud. However, in its previous version it did not have SDN/NFV capabilities, which have been explored by Alonso et al. [13] for its current incorporation to the architecture. The new proposed architecture contemplates the use of Deep Reinforcement Learning techniques for the implementation of the *SDN controller*. In this sense, it is proposed the application of a Deep Q-Learning model [14] for the management of virtual data flows in SDN/NFV in an Edge-IoT architecture according to the required quality of service.

The rest of this work is structured as follows. The next section (Section 2) describes the problem to be solved along with the state of the art in the field of Internet of Things and Edge Computing solutions, the different approaches when using Network Function Virtualization and Software-Defined Networks to Edge Computing scenarios, as well as the application of Deep Reinforcement Learning for data flow management in networks. After that, Section 3 presents the Global Edge Computing Architecture and the new mechanism proposed to manage data flows in Software Defined Networks based on GECA. Then, Section 4 describes the experimentation and the obtained results after implementing the GECA 2.0 SDN mechanism based on Deep Q-Networks. Finally, conclusions and future work are presented in Section 5.

2. Problem Description and Related Work

In this section we will progressively describe the problem to be solved. In order to do this, we will make an analysis of the related work existing in the field of the Internet of Things and Industrial Internet of Things in Section 2.1. After that, in Section 2.2 we will describe what advantages the Edge Computing paradigm provides over solutions based only on IoT and Cloud layers. On the other hand, Section 2.3 will describe how Software-Defined Networking and Network Function Virtualization can help in Edge-IoT scenarios. Finally, Section 2.4 will present Reinforcement Learning and Deep Reinforcement Learning techniques to manage network virtualization in Edge-IoT approaches.

2.1. Challenges in Internet of Things and Industrial Internet of Things Scenarios

The *Internet of Things* (IoT) is, in part, an evolution of other concepts such as *Cyber-Physical Systems*, *Sensor Networks* and *Wireless Sensor Networks* [15], among others. Wireless Sensor Networks allow us to collect information about different physical measurements of the users' environment and, sometimes, of the users themselves [16]. WSNs have been applied in multiple scenarios such as healthcare and telecare [17], smart buildings [18] or smart farming [6]. As can be seen, over time there have been and are multiple wireless technologies to implement WSNs. In fact, we often find scenarios where several technologies are used at the same time because there are several elements to be monitored (e.g., people, objects, the environment, etc.), so we talk about *Heterogeneous Wireless Sensor Networks* [17]. To accommodate these necessary heterogeneity, different approaches have been employed in terms of frameworks and platforms for cohabitation and information fusion [15]. Nonetheless, the need to connect an increasing number of sensors of different types and monitor and store information in the Cloud, leads us inexorably to the Internet of Things.

However, it was not until 1999 that the term *Internet of Things* was first mentioned by Kevin Ashton [19]. Interest in IoT grew as large companies and governments saw it as a key technology. In this sense, Google began to store data related to the Wi-Fi networks of the users of its services in 2010. That same year, the Chinese government established the IoT as a priority topic within its five-year plan [20]. Over the years the terminology took on its own formal definitions, such as that of Kethareswaran and Ram [21], which defined the Internet of Things as the connection of objects (buildings, vehicles) through a network infrastructure with electronic elements (sensors, actuators, radio frequency identification tags, etc.) to collect and exchange data. The most important IoT application areas include healthcare [17], transport and logistics [22], smart homes [23], smart energy [4], smart cities [24] or Industry 4.0 [25], among many others. On the other hand, the *Industrial Internet of Things* (IIoT) [26] is one of the fundamental aspects of the new *Industry 4.0* [27]. The IIoT emerges as a new milestone on the Internet so that billions of machines in the industry are equipped with various types of sensors, connected to the Internet through heterogeneous networks [28]. Possible applications of the IIoT include abnormal pattern detection [29] or predictive maintenance [25], among many others.

Although the implementation of IoT ingestion layers allows to deal with the problem of heterogeneity, two other problems remain to be solved. One of them is the large amount of data that thousands of devices can send to an IoT platform. To reduce the data traffic between the IoT layer and the cloud, solutions such as the *Edge Computing* paradigm arise. Edge Computing allows reducing congestion by the demand for computing resources, network or storage in the cloud. With this strategy, computational and service infrastructures approach the end user by migrating data filtering, processing or storage from the cloud to the edge of the network [30,31]. The second challenge to be solved is that, regardless of the transmission technology used, there may be multiple different user or applications using a common transmission infrastructure from the IoT to the Cloud. Sharing network resources and infrastructure by various organizations, operators and user groups can be done through Network Function Virtualization and Software-Defined Networks [13].

2.2. Edge Computing and Edge-IoT Platforms

The Edge Computing paradigm allows us to take part of the computational load from the Cloud to the Edge nodes [30]. Shi et al. [32] define *Edge computing* as computer and network resources located between data sources, such as IoT devices, and cloud data centers. In these edge nodes it is possible to filter the information coming from the IoT layer to the Cloud, thus reducing the cost of computing and storage services in the Cloud. Edge nodes can pre-process the information collected from the IoT nodes, thus reducing the information to be processed and stored in the Cloud [11]. In addition, it is possible to execute Machine Learning or even Deep Learning algorithms in the Edge nodes, so that a more direct service can be given to users, reducing response times [6]. It is also possible to continue providing service temporarily during communication breaks with the Cloud [12]. Figure 1 shows the basic scheme of an edge computing architecture where the edge nodes allow user application processes to be executed closer to the data sources [30]. The edge nodes perform compute tasks such as filtering, processing, caching, load balancing by reducing data sent or received from the cloud and requesting services and information [13].

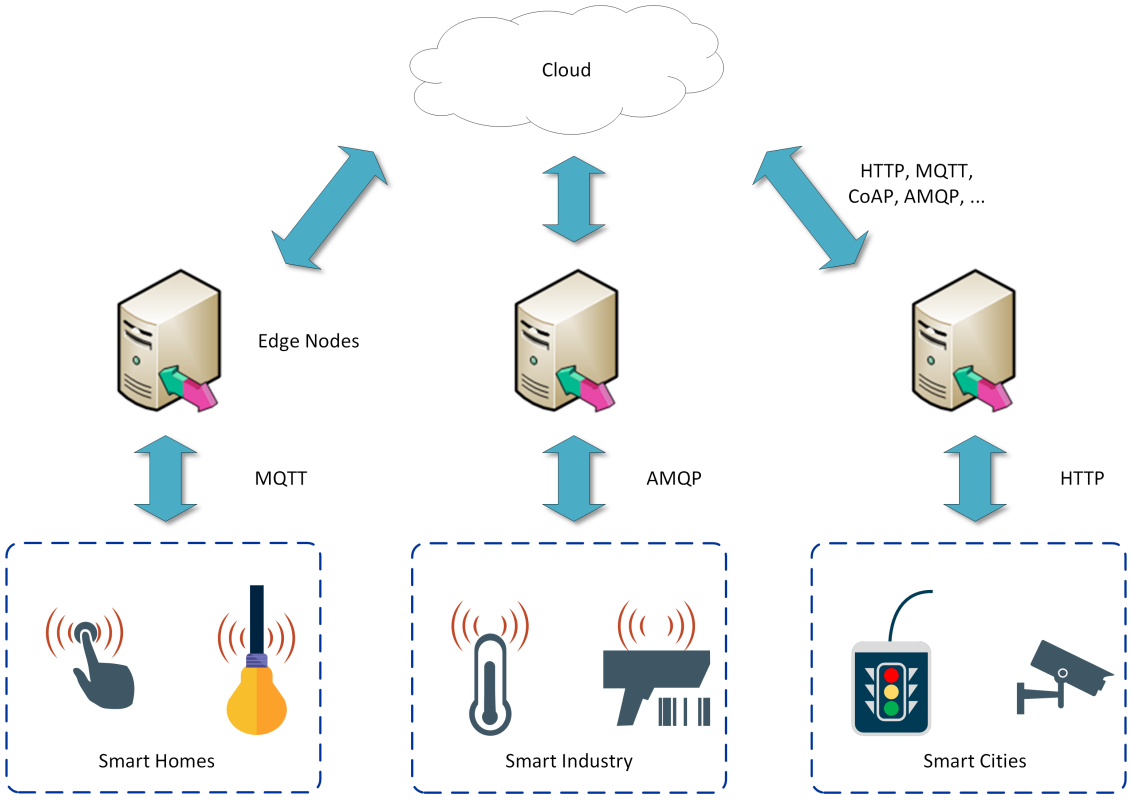

Figure 1. Edge Computing architecture, based on the work of Yu et al. [30].

There is a wide variety of scenarios where solutions based on IoT and Edge Computing are being applied. Among the most relevant applications, we find *Smart Farming* [6], *Smart Energy* [33] or *Industry 4.0* solutions [11], among many others. Moreover, although there are scenarios in which Edge Computing is applied to a single environment as an ad-hoc system, there are also developments aimed at providing Edge functionalities as a platform. In this way, the reproducibility of the solution is increased. Likewise, there are different reference architectures within the scope of Edge Computing applied in industrial environments or *Industry 4.0*. In fact, one of them is the *Global Edge Computing Architecture* [11], on which this work has been based. This architecture, in turn, was the result of analyzing four of the most important reference architectures in the field of Edge-IoT in Industry 4.0. The first of these architectures is *FAR-Edge* [34]. One of the aspects taken by GECA from FAR-Edge is that both incorporate blockchain functionalities. However, in the case of FAR-Edge the blockchain is

incorporated in its intermediate layer, while in GECA it is implemented from the base layer, that is, at the same time that the data are generated by the IoT sensors. Another architecture that has served as a reference for GECA is *INTEL-SAP Reference Architecture* [35]. GECA is inspired by the *SAP Cloud Trust Center* concept, which aims to verify the ownership of devices and register them in the system or platform implemented, assigning a certificate of identity (authenticity) to each owner and keeping an updated list of successfully registered device. In the case of GECA this process is done in the Cloud, before a new device can access a GECA based system or platform. Thirdly, there is the architecture of the *Edge Computing Consortium* [36]. GECA is based on this architecture in terms of the importance of following a standard. Therefore, GECA's design is based on the IEC 61499 standard [37], following a structure of functional blocks, each with its corresponding inputs, processes and outputs. Finally, the architecture of the Industrial Internet Consortium (IIC) proposes the inclusion of an *Enterprise Layer*, on which GECA's *Business Solution Layer* is based [38].

There are two main models when designing Edge Computing solutions [30]. In the *hierarchical model*, the edge architecture is divided into a hierarchy in which functions are defined based on distance and resources. In the hierarchical model, therefore, the different edge and cloudlet servers are deployed at different distances from the end users. Examples of the hierarchical model are the work of Jararweh et al. [39], who propose a model based on the integration of cloudlets and Mobile Edge Computing (MEC) servers. The GECA architecture on which this work is based also followed the hierarchical model before the update proposed in this paper [11]. The second one is the *Software-Defined model* [8], described in the next section.

2.3. Software-Defined Networking and Network Function Virtualization in Edge-IoT Scenarios

In order to provide a more efficient use of resources in IoT networks, new solutions are emerging to virtualize resources [40]. In this sense, concepts such as *Network Function Virtualization* (NFV) arise, oriented to the virtualization of the different components of the network [7]. In fact, ETSI MEC (Mobile Edge Computing) is based on the NFV concept, and its application in IoT scenarios has been explored widely [41]. In a way closely related to the NFV, and often used complementing each other, *Software-Defined Networks* also emerge [8]. Approaches such as Software-Defined Networks and Network Function Virtualization enable cost savings by employing general purpose devices rather than more expensive network-specific devices that may need to be replaced if network configuration needs change over time [9].

Figure 2 shows the typical architecture of *Software-Defined Networks* [42]. In this type of architecture, the network is separated into a *data plane* and a *control plane* [43]. The data plane consists of COTS forwarding nodes with general purpose capabilities, rather than specific purpose hardware (e.g., routers, gateways, etc.) [9]. The remote configuration of the nodes is done from a *controller* (i.e., *SDN Controller*) in the control plane. In this way, the network administrator uses a centralized control console to reconfigure the traffic flow on the network without having to physically modify the network nodes [42]. For this purpose, all these nodes have a common interface for remote configuration, called *southbound interface* [44]. Southbound interfaces allow abstracting the functionality of the COTS forwarding nodes and allow the controller to interact with them. In this regard, *OpenFlow* is the most representative of southbound interfaces [8].

The open-source OpenFlow protocol allows decoupling the *control plane* of the *data plane* of the networks. Thanks to the uncoupling of these two planes, the control and management of the network can be carried out remotely in the cloud (in a centralized or distributed way), while packet forwarding is carried out in the hardware devices that make up the network [8]. The control plane commands hardware devices specifying how to forward these packets between their adjacent nodes. Therefore, it is no longer necessary to build different hardware devices with specific functions (ASIC circuits) [7], making it possible to use cheaper general-purpose hardware, with a lower unit cost and that can evolve over time simply updating its functions and software remotely, reducing replacement and warehousing costs. Furthermore, in the control plane of a SDN, a *Network Operating System (NOS)* is

running over southbound and northbound interfaces [45]. NOS describe the available software-defined networking software tools. Over the specific NOS, is possible to build applications aimed at controlling the network behavior and define high-level network policies (i.e., the *management plane*) [8]. For this purpose, *northbound interfaces* allow these applications to interact with the NOS. A human network administrator uses a centralized control console to reconfigure the traffic flow on the network without having to physically modify the network nodes [42]. However, the administration of the network can be carried out in an automated way by specific software such as a cognitive engine based on intelligent algorithms. This is, in fact, one of the objectives of this work. Examples of this type of approach are the work of Baek et al. [46], or the work of Sampaio et al. [47], as we discuss later in Section 2.4.

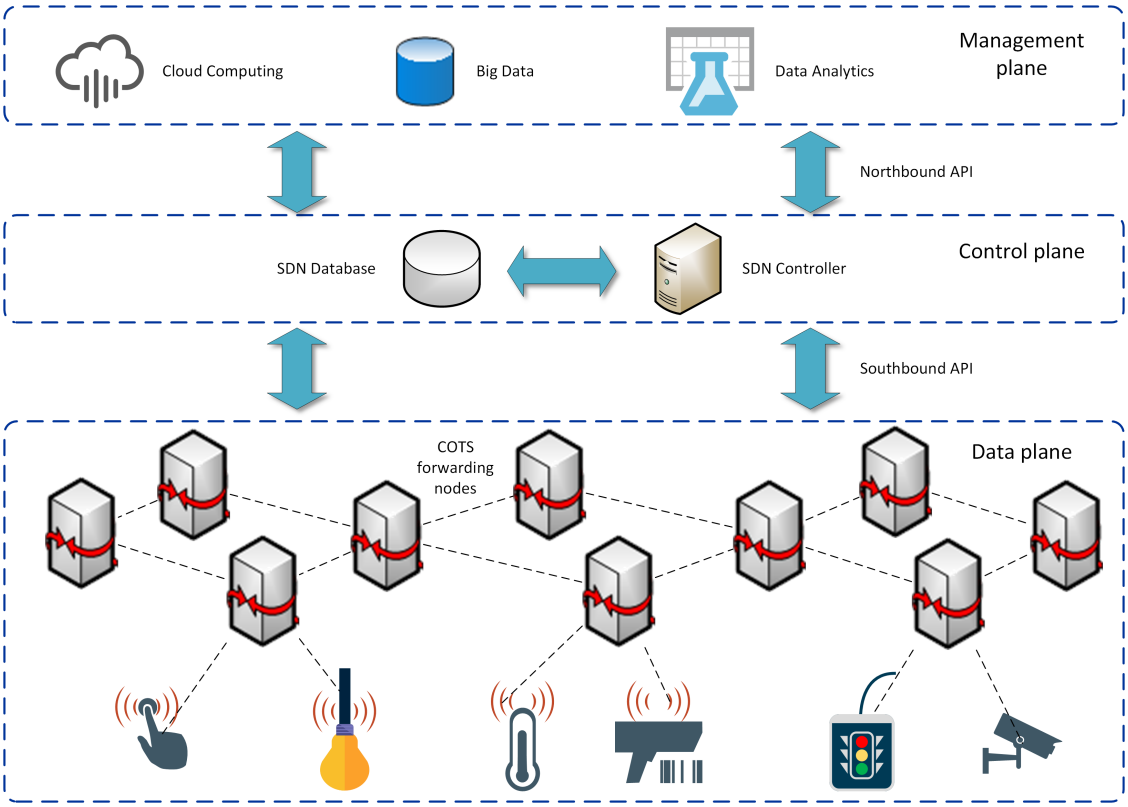

Figure 2. Software Defined Network (SDN) architecture.

The concept of *network virtualization* is defined according to Granelli et al. [7] as the process of combining hardware and software network resources, as well as the network's own functionalities, into a single software-based entity that is called a *virtual network*. Figure 3 shows this approach [42]. As mentioned above, the integration of SDN and NFV is common in the field of scientific and industrial research to take advantage of the benefits of both [43]. Just as there are intelligent algorithms aimed at balancing resources, configuring routing and firewall rules remotely from the management plane using the northbound interface in SDN scenarios, there are also researches that propose intelligent mechanisms aimed at provisioning resources in NFVs, such as the work of Ruiz et al. [48] or Pei et al. [49]. Again, our work is focused on provide with new intelligent mechanisms that allow provisioning virtual network resources in SDN and NFV scenarios.

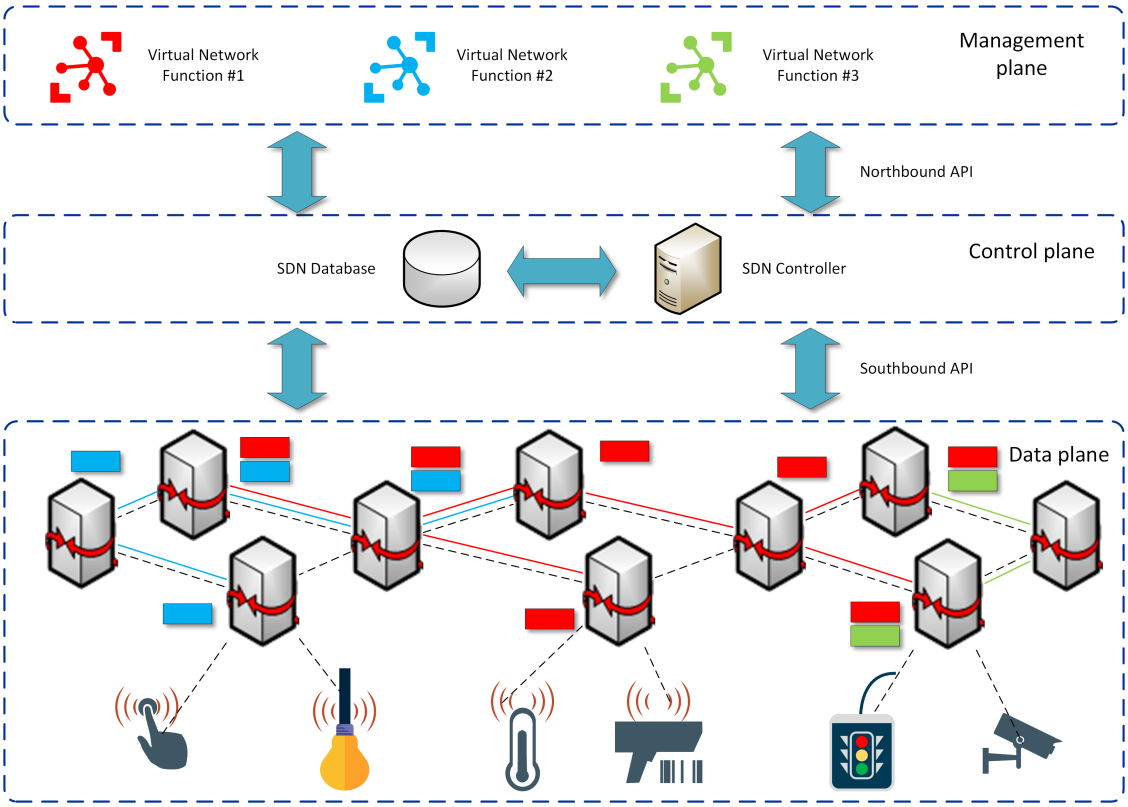

Figure 3. Network Function Virtualization (NFV) over Software-Defined Networks.

There are studies in Software-Defined Networks [50], Wireless Software-Defined Networks [51] and Network Function Virtualization [52] as complementary technologies that could work together with Edge Computing architectures. Jararweh et al. [53] propose a software-defined model aimed at the integration of *Mobile Edge Computing (MEC)* and SDN. Salman et al. [54] go one step further, proposing the integration of MEC, SDN and NFV, achieving a better MEC performance in mobile networks and that can be further extended enabling IoT-wide deployment scenarios. More specifically, there are different solutions aimed at combining both paradigms to further optimize resources in IoT networks [55]. Within the different approaches, we find *HomeCloud* [56], a framework that combines the use of SDN and NFV with the aim of allowing efficient orchestration and delivery of applications from the servers that are deployed on the Edge itself. Caraguay et al. [40] propose a SDN/NFV architecture specifically for IoT networks and focused on modifying QoE/QoS flows in real-time by means of the controller capabilities. Likewise, Monfared et al. [57] propose a two-tier cloud architecture. In Monfared et al. [57] architecture, on the one hand, there are data servers in the cloud and, on the other, there are edge devices to provide data closer to users. For the control and management of the architecture, a network infrastructure defined by the software is proposed.

Nonetheless, it is necessary to find innovative solutions that design and implement intelligent algorithms that allow automated tasks such as resource balancing, network reconfiguration or cyberattack prevention in scenarios where IoT, Edge Computing and SDN/NFV are combined. In this sense, there are some solutions like the proposal of Ruiz et al. [48], who proposes a genetic algorithm for VNF provisioning in NFV-Enabled Cloud/MEC RAN architectures. However, among the intelligent algorithms one of the most promising fields in recent years is *Reinforcement Learning*. This is the basis of approaches such as that of He et al. [58], which proposes a deep reinforcement learning approach in SDN with MEC. In the next section we will briefly analyze the state of the art of the use of Deep Reinforcement Learning in SDN/NFV scenarios.

2.4. Reinforcement Learning and Deep Reinforcement Learning in SDN/NFV Scenarios

Reinforcement learning (and *Deep Reinforcement Learning*) is one of the most promising trends in recent years in the field of machine learning [10]. Among its applications are self-driving cars [59], robot vacuums [60], automated trading [61], enterprise resource management [62,63] or games. In this sense, in 2017 was released *AlphaGo Zero*, an algorithm that was not trained by human competitor data. AlphaGo Zero was only able to sense positions on the board, rather than having previous data. AlphaGo Zero also ran on a single machine with 4 TPUs and beat previous AlphaGo Lee [64] in three days by 100 games to 0. AlphaGo Zero's neural network was previously trained using TensorFlow with 64 GPU workers and 19 CPU parameter servers [65]. As can be seen, the potential for Deep Reinforcement Learning in scenarios where the capacity for initial training dataset is not available is enormous. This allows envisioning solutions in which to build devices that learn from scratch the best way to balance data flows in a software defined network taking into account the different QoS (Quality of Service) and optimizing the use of resources [58], or optimize the use of energy in edge scheduling [66]. Likewise, they represent algorithms with a great potential to detect new cyberattacks not yet known without the need to train new models as new threats arise [67].

The *Q-learning* algorithm [68] is one of the best known model-free techniques in reinforcement learning, and has numerous evolutions and variants of the same [69]. For any *Finite Markov Decision Processes (FMDP)*, Q-learning (Q comes from *Quality*) finds a policy that is optimal in the sense that it maximizes the expected value of the total reward over any and all successive steps, starting from the current state [70]. Thus, we have a *value function* $Q(s, a)$ that takes as input the current state s and the current action a and returns the expected reward for that action and all subsequent actions. Initially, Q returns the same arbitrary value. As the agent explores the environment, Q returns an increasingly better approximation of the value of an a action, given a s state. That is, the function Q is progressively updated. So, the new value $Q^{new}(s_t, a_t)$ is updated in each iteration from the old value $Q(s_t, a_t)$ with a *learning rate* α, since the *learned value* $(r_t + \gamma \cdot \max_a Q(s_{t+1}, a))$ is known, where r_t is the *reward*, γ the *discount factor* and $\max_a Q(s_{t+1}, a)$ the estimate of the future optimal value:

$$Q^{new}(s_t, a_t) \leftarrow (1 - \alpha)Q(s_t, a_t) + \alpha \left(r_t + \gamma \cdot \max_a Q(s_{t+1}, a) \right). \quad (1)$$

In this way, the agent has an estimated value for each state-action pair, and whose knowledge increases with time. With this information, the agent can select which action to carry out in each moment according to its *action-selection strategy*, which can take into account an epsilon-greedy strategy, for example, to increase its knowledge of the environment.

Deep learning has accelerated progress in Reinforcement Learning, with the use of deep learning algorithms within the RL that define the field of *Deep Reinforcement Learning* [69]. Deep Learning allows Reinforcement Learning to scale up to previously intractable decision-making problems, that is, environments with high dimensional states and action spaces. There are several Deep Reinforcement Learning algorithms [69], including the variants of Deep Q-Learning (DQN), such as double DQN [66], dueling DQN, Deep Recurrent Q-Networks (DQRN) and multi-step DQN, among others, and those based on policy gradient, such as REINFORCE, Advantage Actor-Critic (A2C), Natural Policy Gradient (NPG), among many others.

Thus, deep reinforcement learning techniques provide great potential in IoT, edge and SDN scenarios. In the work by Amiri et al. [71], reinforcement learning techniques are used for energy management in heterogeneous networks based on the QoS required by each service in the network. Liu et al. [72] focused on designing an IoT-based energy management system using an Edge Computing infrastructure with deep reinforcement learning. Ferdowsi and Saad [67] proposes the use of deep reinforcement learning using LSTM (*Long and Short-Term Memories*) [73] blocks for signal authentication and security in massive IoT systems.

One of the most relevant works in this sense is the approach by Mu et al. [74], which proposes a SDN flow entry management mechanism based on Deep Q-Networks and compared with classic

Q-learning reinforcement learning using Mininet emulator [75]. In our work, Deep Q-Networks is proposed to manage the Network Function Virtualization in Edge-IoT scenarios. Unlike the work of Mu et al. [74], whose methodology serves as a basis for the experimentation stage of our work, our proposal is oriented towards a specific SDN management solution in a reference architecture aimed to the design, implementation and deployment of Edge-IoT solutions, in which the flows between nodes are almost entirely *mice flows* (that carries small amounts of data).

3. Management of SDN Flow Entries in the Global Edge Computing Architecture by means of Deep Reinforcement Learning

In this section we will describe the contribution of this work. First, Section 3.1 will depict the GECA architecture in its previous state before introducing SDN and NFV management. After that, Section 3.2 will describe the new components introduced in the architecture in order to optimize the management of data flows. In this sense, a flow management mechanism will be introduced in the Cloud layer (*Business Solution* Layer) based on Deep Reinforcement Learning based on rewards indicated by the Edge nodes based on their *satisfaction* with the observed QoS. The mechanism will be based on Deep Q-Networks, and its concept design will be explained in Section 3.3.

3.1. The Global Edge Computing Architecture 1.0

This section depicts the *Global Edge Computing Architecture* (GECA), an architecture designed and presented by Sittón-Candanedo et al. [11] in a modular and tiered manner, based on the concept of functional blocks. GECA is structured in three layers: *IoT*, *Edge* and *Business Solution* layers. The main objective of the architecture is to incorporate low-cost but high-capacity computing resources in its edge layer to allow pre-processing and filtering of the volume of data sent in a traditional IoT environment by the set of connected sensors or devices. The principal features and components of the layers of the architecture, shown in Figure 4 are described in the following order:

- *IoT Layer*: in GECA, this layer is made up of the set of elements that define an environment like IoT, that is, objects or connected things that constantly generate data. Among them are sensors, actuators, controllers or IoT gateways. The data transmission process in this layer is done through the most used communication standards, such as: Wi-Fi, ZigBee, LoRa or SigFox. In addition to the IoT devices, this layer includes the components that provide security to the architecture. The incorporation of a basic blockchain scheme in which the smart contracts through oracles interact with the physical components of the layer, allows the data to be transferred safely and following the predefined terms in each contract.
- *Edge Layer*: This layer proposes the use of low-cost and high-capacity boards such as the Raspberry Pi, whose characteristics allow it to function as an edge node to process and filter the data collected and sent by the devices deployed in the IoT layer. The components of these boards: I/O ports, 32 or 64 bit Linux operating system, RAM memory up to 4GB, USB 2.0 to 3.0 ports or SD cards, allow the installation of libraries such as TensorFlow Lite or similar used to apply Machine Learning techniques for data management [76]. Machine Learning techniques contribute to the deployment of the Data Analytics in the Edge layer, making it easier for users to obtain valuable data and responses with lower latency, reducing the costs associated with Cloud Computing such as shipping, processing, data storage as well as bandwidth consumption.
- *Business Solution Layer*: The services and applications associated with Business Intelligence, which are in a classic Cloud Computing architecture, are included in this layer of GECA. At this level the architecture facilitates the deployment of public or private services, also allowing the inclusion of components of: analysis (case based reasoning, data analysis and visualization algorithms), authentication (to ensure security) knowledge base (using virtual agent organizations or decision support systems) and APIs (so that services are available in any standard web browser).

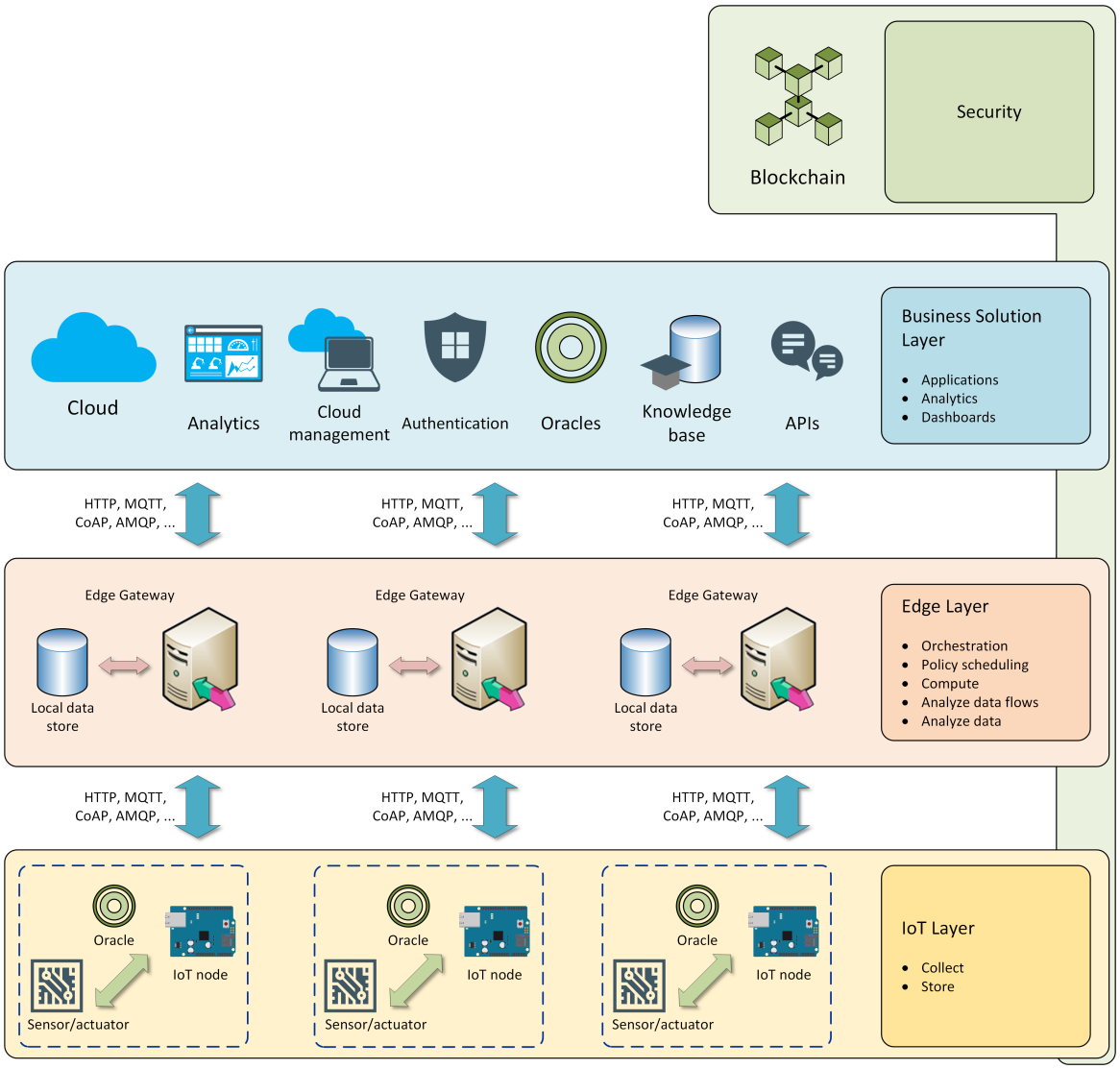

Figure 4. The schema of the Global Edge Computing Architecture.

This architecture was designed from the design phase to support the deployment of solutions that required IoT, Edge and Cloud layers, and securing the information through a blockchain from the moment the data was entered from the IoT nodes. As demonstrated through its application in different scenarios, this architecture offers a scalable and secure environment for users to obtain their information in real time, remotely accessing the applications that are deployed in the Business solution layer [6,12]. Nevertheless, the initial design did not consider the possibility of using software defined networks or network function virtualization to optimize the use of resources in the Edge-IoT networks. In this work, this architecture is updated taking into account these new functionalities.

3.2. SDN and NFV in the New Global Edge Computing Architecture 2.0

In order to support SDN and NFV capabilities, some of the layers in the Global Edge Computing Architecture were updated, resulting in the Global Edge Computing Architecture 2.0. The new architecture is shown in Figure 5. Thanks to the modular and scalable design of the architecture, the introduction of the new sub-layers and components was a straightforward process.

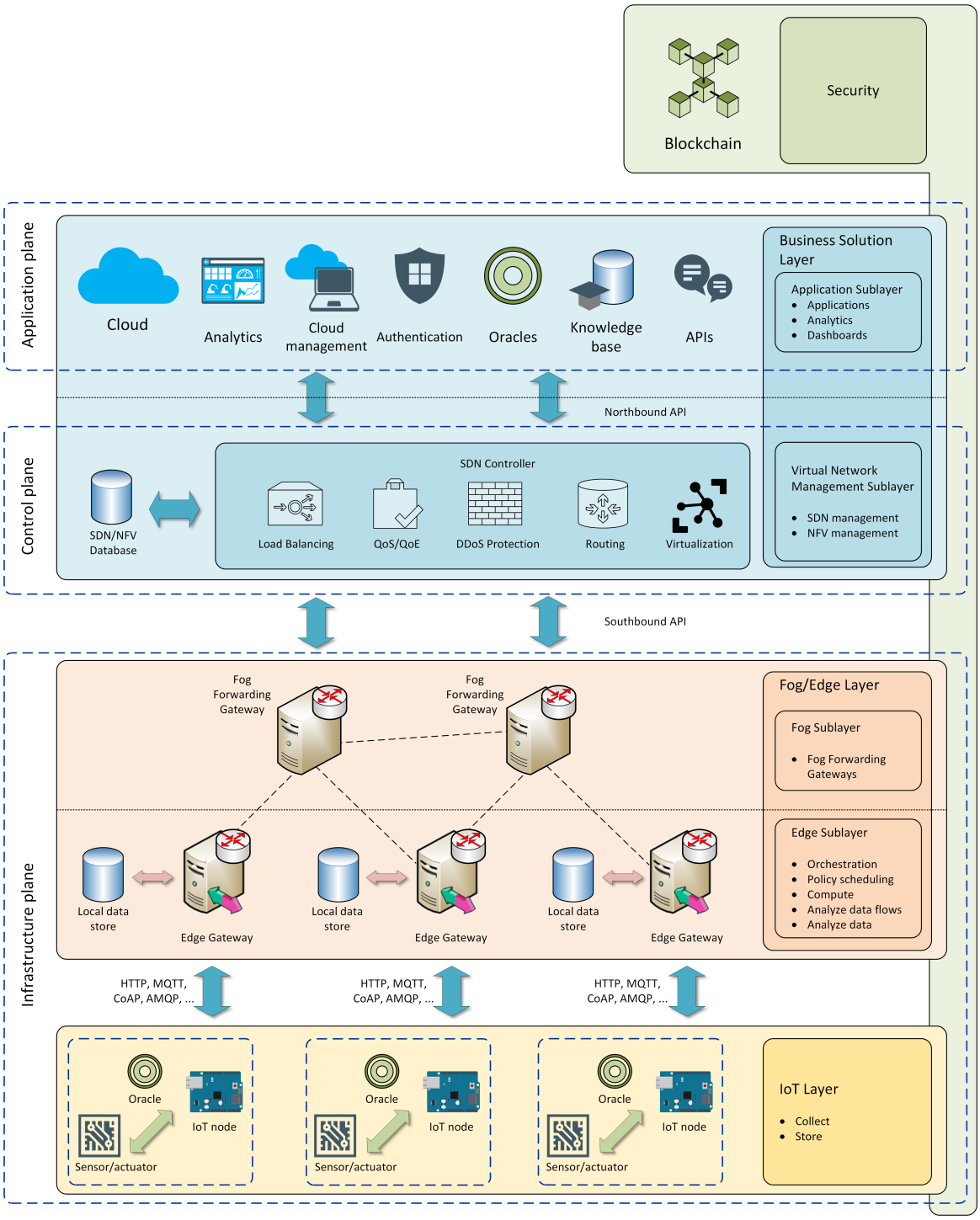

Figure 5. The schema of the Global Edge Computing Architecture 2.0.

Therefore, the architecture was updated according to the following modifications:

- *IoT Layer*: This layer and its components remain unchanged in the new version of the architecture. In this sense, it is not necessary to modify any of its components, since in this version only the interaction with the Edge nodes (and the new Fog nodes) from the *SDN controller* has been considered. The IoT Layer is the lowest layer within the *Infrastructure plane* (called *data plane* in other SDN architectures).

- *Fog/Edge Layer*: The previous *Edge Layer* in GECA 1.0 is now called *Fog/Edge Layer*. Moreover, the *Fog/Edge Layer* is the topmost layer within the *Infrastructure plane*. This layer is now subdivided into two sublayers: the *Edge Sublayer* and the *Fog Sublayer*.

  - *Edge Sublayer*: This sublayer inherits the existing components of the previous GECA 1.0 Edge Layer. That is, it is formed by the Edge nodes or *Edge Gateways*. Thus, these nodes act as a gateway between the IoT devices and the applications in the Cloud (*Business Solution Layer*). Like in GECA 1.0, the Edge Gateways have computing and local storage capacities to preprocess the data sent from the *IoT Layer* to the *Business Solution layer*. Besides, it is possible to apply machine learning techniques in the same *Edge* in order to detect anomalous patterns. Among others, it is possible to apply k-NN (*k-Nearest Neighbors*) algorithms to detect anomalous service requests that may mean cyberattacks [77]. Nonetheless, in the new version of the GECA 2.0 architecture, the *Edge Gateways* include in their local data stores routing tables with information about what to do with each type of packet (discard, forward to a certain communication interface or send to the *SDN Controller*), as well as counters with the number of packets received of each type. The packet types are actually filtering patterns based on the packet headers. The entries (rules) in the tables of each *Edge Gateway* are configured and consulted remotely from the *SDN Controller* in the Cloud (i.e., in the *Business Solution Layer*) by means of the *Southbound API*. In this sense, this nodes incorporate capabilities similar to those of the *COTS forwarding nodes* in Figure 2. Each designer can follow the OpenFlow standard or implement his/her own protocol. The architecture is flexible in this sense, in order to be open and dynamic in time.
  - *Fog Sublayer*: This new sublayer is composed of *Fog Forwading Gateways*. Like the *Edge Gateways* in the *Edge Sublayer*, the *Fog Forwarding Gateways* include packet routing tables that can be configured remotely from the *SDN Controller* via the *Southbound API*. However, these types of nodes are not connected directly to a subnet of devices in the *IoT Layer* and do not need to be able to apply machine learning techniques on the edge. Thus, their function is quite similar to that of the *COTS forwarding nodes* in Figure 2. Thus, they are not provided with computational or storage resources that would raise the costs of the network infrastructure (whether physical or virtual).

- *Business Solution Layer*: The *Business Solution Layer* maintains its name in the new GECA 2.0. However, the *Business Solution Layer* is now subdivided into two sublayers: the *Virtual Network Management Sublayer* and the *Application Sublayer*.

  - *Virtual Network Management Sublayer*: This sublayer forms the *control plane* of the SDN architecture and it contains the *SDN Controller*. Also, in this sublayer would be carried out the capabilities of Network Function Virtualization when managing flows. The same *SDN Controller* carries in its SDN/NFV database the management of virtual flows and the dynamic reconfiguration of remote nodes. In this way, from the first moment of the GECA 2.0 design, it is possible to use SDN and NFV functionalities indistinctly or combined if the network designer requires it. The *SDN Controller* makes use of the *Southbound API* to remotely reconfigure both *Fog Forwarding Gateways* and *Edge Gateways* in the *Fog/Edge Layer*. In addition, it offers elements in the *Application Sublayer* the possibility of accessing the SDN configuration and the data exchange with the IoT-Edge layers via the *Northbound API*. A proposal for network resource allocation and flow control to be implemented as part of the *SDN Controller* will be described in the Section 3.3.
  - *Application Sublayer*: This sublayer includes all the functionalities that previously existed in the *Business Solution Layer* in GECA 1.0. The difference with GECA 1.0 is that now the different user applications and the different management components included in the *Business Solution Layer* communicate with the network through the *Northbound API* offered

by the *SDN Controller*. Thus, the application layer corresponds to the *application plane* (or *management plane* in other SDN architectures). Also, thanks to the functionalities of the *SDN Controller*, different tenants or different applications can share the network infrastructure transparently, without seeing the data of other tenants or applications that are in another virtual network sharing the same infrastructure.

GECA was inspired in its initial design by other reference architectures oriented to Industry 4.0 applications, so security is an essential aspect to take into account. Thus, as in the first version of GECA, in GECA 2.0 there is a transversal layer for the securization of all the data exchanged. This includes the management of the authentication of the IoT, Edge or Fog devices in the network before they become part of it, as well as the management by design of distributed ledger technologies, such as blockchain, in order to save all the transactions carried out by the IoT devices [11]. The IoT layer contemplates the use of hardware encryption devices to facilitate the incorporation of secure transaction blocks from the very moment the data is collected by the sensors, behaving like blockchain *oracles*, as they are information exchange paths between the blockchain and the real physical world [78]. There are also oracles in the *Business Solution Sublayer* in order to exchange dice with the virtual world outside the blockchain.

3.3. Adaptive Assignment of Network Resources by Means of Deep Q-Learning Techniques

This section will describe the method that is proposed to be developed in GECA 2.0 to dynamically establish the routing tables in the different *Fog Forwarding Gateways* and the *Edge Gateways* from the *SDN Controller*. To do this, first we will describe in more detail the Q-Learning algorithm and the Deep Q-Networks.

As introduced in Section 2.4, *Q-Learning* is a reinforcement learning algorithm aimed at learning a *policy* [68]. This policy or strategy is the core of the agent and it is what dictates how the agent behaves with the environment. The policy determines what action the agent will take given a given state. For example, which resources will be reserved in each moment by a SDN controller in a software-defined network [62].

Mathematically, the policy in the Q-learning algorithm is modeled as an *action-value function* $Q(s, a)$. That is, $Q(s, a)$ determines the policy of the agent, resulting in the *Q-value* or *quality* level that it obtains when taking the action a given that it is in the state s. The more $Q(s, a)$ approaches the best policy, the better it will have been its learning. The objective of the agent is, therefore, to learn the best $Q(s, a)$, which is known as *TD-target* or $Q^*(s, a)$.

In supervised learning, a *cost function* or *loss function* measures the quality of the $h(x)$ function used as a *hypothesis function* to predict the value of a given label for a new input. In reinforcement learning there is a similar concept of loss function, known as *TD-error*. Thus, the cost function or loss function is:

$$Loss = Q^*(s_t, a_t) - Q(s_t, a_t). \tag{2}$$

However, in reinforcement learning the objective function *TD-target* is always unknown. In this sense, the *Bellman equation* is used to estimate $Q^*(s, a)$ [68]:

$$Q^*(s_t, a_t) = r_{t+1} + \gamma \max Q(s_{t+1}, a). \tag{3}$$

This equation means the following. At the moment t, the agent is in the state s_t, and, because of that, it decides to take the action a_t. Consequently, the agent gets a reward r_{t+1} and its environment observation indicates that it is in the new state s_{t+1}. According to $Q(s, a)$, the agent calculates, based on its knowledge until that moment, the maximum Q-value that it can get given the state s_{t+1} considering all the possible actions that it can take. The *discount factor* γ is a hyper-parameter between 0 and 1 that allows to weigh the importance of the maximum Q-value in the future state.

Thus, the Equation (2) results in:

$$Loss = r_{t+1} + \gamma \max Q(s_{t+1}, a) - Q(s_t, a_t). \tag{4}$$

One way of minimizing the loss function is by the *gradient descent* method [79]. In this method, the $Q(s, a)$ policy is updated based on the current reward and the maximum value of expected future rewards:

$$Q^{new}(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left(r_{t+1} + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t) \right). \tag{5}$$

The *learning rate* α is a hyper-parameter between 0 and 1 that allows to control the speed of convergence of the procedure.

The Q-Learning algorithm can be implemented with a table known as *Q-matrix* (or *Q-table*) of dimensions $S \times A$, where S is the number of possible states and A is the number of actions that the agent can perform in each state. The element i, j in the table, or the element s, a, stores the value $Q(s, a)$.

Initially, the agent has no value (or zero) in each element in the table, since it has not explored the environment yet. The agent, then, starts the first training episode. To do so, it starts moving through the states s of the environment performing actions a and getting rewards r for it. This allows the agent to update the table learning the $Q(s, a)$ policy. Once the first episode is finished, it starts a second episode in which it no longer starts from an empty *Q-matrix*, but it takes into account the policy $Q(s, a)$ that it learned during the first episode. It repeats, in succession, a series of N episodes, each with a maximum number of M actions (or transitions between states) pre-established.

During the learning process, the hyper-parameters explained previously are taken into account: the *discount rate* γ and the *learning rate* α. However, there is another hyper-parameter to be taken into account, the *exploration rate* ϵ, or the percentage of time the agent spends on exploration instead of exploitation. We recall that ϵ can be progressively reduced depending on the knowledge acquired by the agent. Usually, in fact, it starts at 1 and ends at 0, being reduced in each episode by an *exploration decay*, usually $1/N$. During the learning stage, in each state in each episode, the agent will decide to take the optimal policy (*exploitation*) with a probability $P_O = (1 - \epsilon)$ or to follow a random policy (*exploration*), doing a random action, with a probability $P_R = \epsilon$. Once the training of the agent is finished, it passes to the testing stage, and it will always take the optimal policy, unless it continues with its learning also in production.

The problem with the classical method based on *Q-matrix* is that it requires a very high number of memory spaces. Problems are common where the number of states or even the number of actions is very high, so that the memory space $S \times A$ would be unwieldy or directly unmanageable. In these situations the *Q-table* can be replaced by a neural network. Thus, we move from a Q-Learning algorithm to a *Deep Q-Learning algorithm* [80].

In this way, we can use a neural network with an input layer with S units (the number of possible states) and an output layer of A units (the number of possible actions in each state). The number of hidden layers and units for each of the hidden layers will be determined by the design of the network and the testing of the different models. We train the neural network, for example, using an optimal method to minimize loss function, such as gradient descent. In the testing phase, the neural network will take at its input the state s in which it is in the agent. At the output we will obtain the A values $Q(s, a)$ for such state s.

Depending on the instructions previously received via OpenFlow protocol, *flow tables* are established on the switches in a Software-Defined Network implemented following the GECA 2.0 architecture. Each flow table includes *match fields*, an *action* to be performed and *statistics*. Thus, when a switch receives an incoming packet, it checks its header and the match fields in its flow tables. If a *match* occurs, the action is carried out and the counter in the statistics is increased. The actions can be to send the packet to a certain port, send the packet to the controller, or drop the packet.

In this sense, and following partly the methodology of Mu et al. [74], based on Mnih et al. [81], it is necessary to define the different *states*, *actions* and *reward* mechanisms that will feed the Deep Q-Networks.

Let's start with *rewards* to better understand the overall concept. In this regard, an interface is defined so that each IoT node informs periodically to the *SDN Controller* about the response time (i.e., the time between the source node sends a packet to a destination node, and the time the source node receives the acknowledgment from the destination node) in each flow that this IoT node maintains with another IoT node in the network.

Thus, a short response time will imply a greater reward than a long response time. A failure to respond will be considered a penalty. Therefore, the reward related to the flow between the source IoT node i and the destination IoT node j will be based on previous response time following the Equation (6):

$$r_{i,j,t} = \begin{cases} +1 & \text{if } response\_time_{i,j,t} < response\_time_{i,j,current\_best} \\ 0 & \text{if } response\_time_{i,j,t} = response\_time_{i,j,current\_best} \\ -1 & \text{if } response\_time_{i,j,t} > response\_time_{i,j,current\_best} \\ -Timeout\_penalty & \text{if } timeout \end{cases}, \tag{6}$$

where $Timeout\_penalty$ is a positive value to penalize appropriately if the current forwarding tables are wrong, but considering that a transient network interruption or an overload in the destination IoT node is possible even though with a right table configuration.

Then, the total reward in each cycle will be given by the Equation (7):

$$r_t = \sum_{i,j} r_{i,j,t}. \tag{7}$$

Regarding *states*, in this first release of the GECA 2.0 SDN mechanism, the *SDN Controller* knows and establishes all the different flow entries in the flow tables in all the *Edge Gateways* and *Fog Forwarding Gateways* registered in the network in a centralized way, as described in Equation (8):

$$s_t = \left\{ flow\_entry_{i,j} \right\}, \tag{8}$$

where $flow\_entry_{i,j}$ represents the j-th entry in the table in the i-th *Edge Gateway* or *Fog Forwarding Gateway* node.

The different possible *actions* are defined by the possible actions that can be performed from the SDN Controller to the *Edge Gateways* and *Fog Forwarding Gateways* through the *Southbound API*. That is, keep an entry in a table, update it, delete it or add a new entry in the table. Thus, for each node in the *Fog/Edge layer* there will be at least 4 possible actions in the output layer of the Q-Network, which determines the actions to be taken by the mechanism for a certain $flow\_entry_{i,j}$, as given by Equation (9):

$$a_{i,j} = \left\{ keep_{i,j}, update_{i,j}, add_{i,j}, delete_{i,j} \right\}. \tag{9}$$

Therefore, the complete set of possible actions in each iteration will be given by Equation (10):

$$a_t = \bigcup_{i,j} a_{i,j,t}. \tag{10}$$

4. Experimentation and Results

For the implementation of the Deep Q-Network, we utilize TensorFlow and Keras libraries under Python. As depicted in Figure 6, in our Q-Network we use as input layer the current states of the system (given by (8)) and the current performance of the IoT network, given by the last obtained reward

(Equation (7)). For the output layer we consider an output per each possible action (Equation (10)). For the intermediate (hidden) layers, we have considered different network architectures (distinct combination of layers and neurons per layer) for the tests to compare their results, but using always fully-connected inner product layers.

As in Reference [74], we use ReLU (*Rectifier Lineal Unit*) functions as activation functions, as provided by Equation (11).

$$f(x) = \begin{cases} 0 & \text{if } x < 0 \\ x & \text{if } x \geq 0 \end{cases}. \tag{11}$$

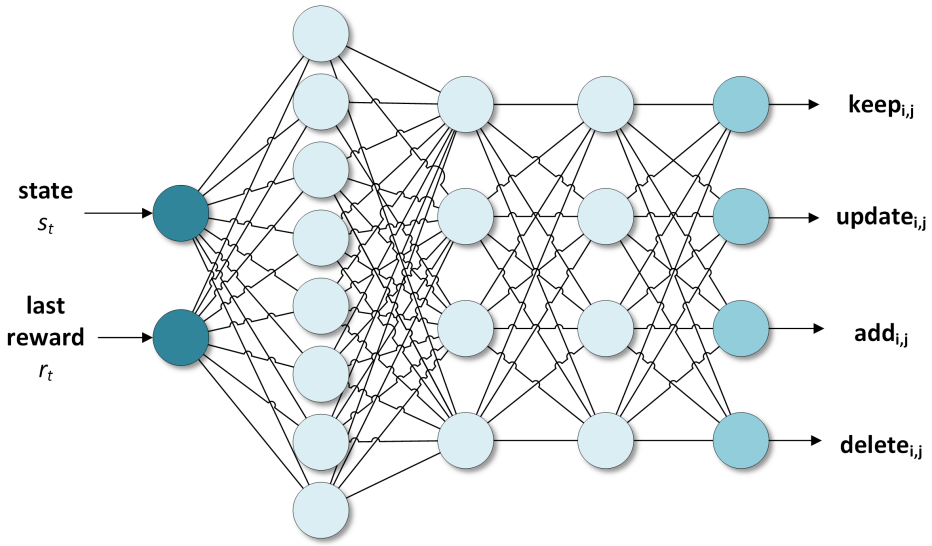

Figure 6. Q-Network used in Global Edge Computing Architecture (GECA) 2.0 SDN mechanism.

For the experimentation a Mininet 2.2.2 virtual machine on an Ubuntu 20.4 LTS running directly on Oracle Cloud with a 1 vCPU and 1GB of RAM has been used. In this base experiment, Python 3.8.2 has been used to implement the new mechanism of the Deep Q-Network acting in the SDN controller to dynamically configure the tables in the remote nodes, as this allows to use POX as *Network Operating System* to implement the SDN [8]. Python has also been used to emulate the remote nodes sending periodic response times to the SDN Controller.

In the experiment, we have emulated 1 central SDN controller, 4 fully-connected *Fog Forwarding Gateways*, 8 *Edge Gateways* (2 per each *Fog Forwarding Gateway*) and 16 IoT nodes (2 per each *Edge Gateway*), as depicted in Figure 7. We have only considered *mice flows* (that carries small amounts of data) of just using Mininet's built-in *iperf* tool to generate random traffic patterns, specifically Poisson traffic at an average rate of 128 kbps, considering the work of Mu et al. [74] as design guide. We have not considered *elephant flows* in this experiment as they are not relevant in the majority of IoT environments, specially in communications among IoT nodes.

Two different tests were performed with this IoT network, using two different neural network architectures in each case. In Test 1, three intermediate (hidden) layers were used, with 8, 4 and 4 neurons respectively. In Test 2, three intermediate layers were also used, but this time with 8, 8 and 4 neurons, respectively. In neither of the two scenarios were the networks trained previously with some samples. That is, the initial values of the weights were chosen using random values.

For each of the two cases a maximum number of 1000 episodes was used. In both cases the same hyper-parameters were used, using typical values in Deep Q-Learning—*discount factor* $\gamma = 0.95$, *learning rate* $\alpha = 0.1$ and an initial *exploration rate* $\epsilon_{t=0} = 1.00$, decreasing to a minimum exploration rate $\epsilon_{t=1000} = 0.01$ in the last episode. In order to calculate the loss value, the *Stochastic Gradient Descent* (SGD) method is used.

In both cases the goal to achieve was to increase by 30% the performance of the IoT network, as measured by Equation (7), that is, to reduce by 30% the sum of total response times with respect to the starting situation in the episode $t = 0$.

Each of the tests was run 10 times in order to average the results obtained. The summary of the configuration used and the results obtained is shown in the Table 1.

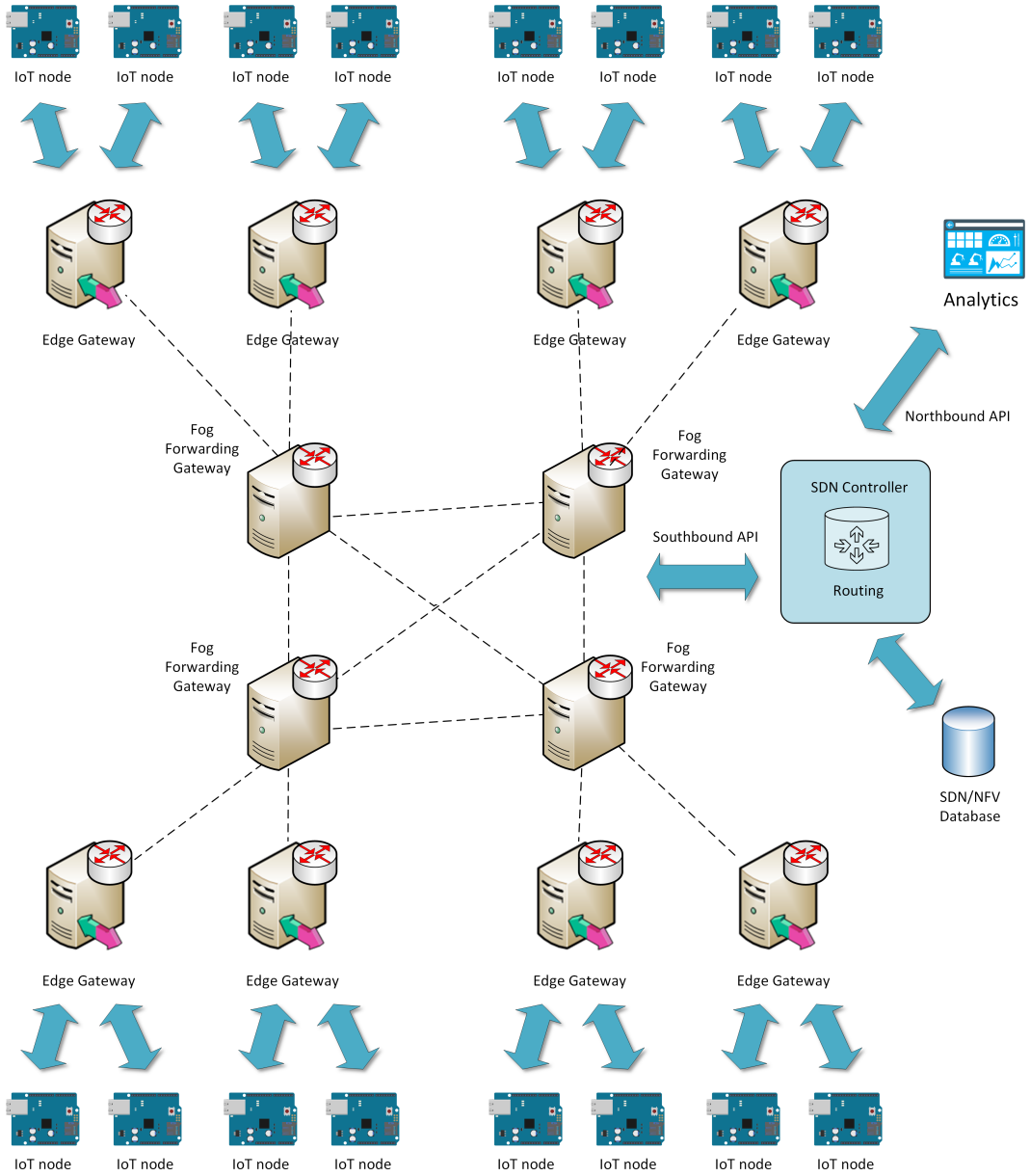

Figure 7. Emulated Fog-Edge-Internet of Things (IoT) network used for the experiment.

As can be seen, the Test 2 neural network achieves better results than the Test 1 network for the configuration used in the experiment. However, the computational load is significantly higher and the tests are executed notably slower in the case of Test 2. It is necessary to perform more tests with different configurations (different IoT network topologies, different traffic flows between nodes and different hyper-parameters) and to incorporate quantitative measures of the computational load required to estimate the cost-benefit balance of using a denser neural network.

Table 1. Comparison of configuration and results obtained with the different Q-Networks.

Parameter	Description	Test 1	Test 2
Neural network architecture	Neurons per layer	4:8:4:4:4	4:8:8:4:4
γ	Discount factor	0.95	0.95
α	Learning rate	0.1	0.1
N	Max episodes	1000	1000
$\epsilon_{t=0}$	Initial exploration rate	1.00	1.00
$\epsilon_{t=1000}$	Minimum exploration rate	0.01	0.01
Goal	Decrease in total response times	30%	30%
Episode in which goal is met	Maximum	198	142
	Average	183	102
	Minimum	163	92

5. Conclusions and Future Work

SDNs ad NFV make easier to deploy and distribute applications by dramatically reducing infrastructure overhead and costs. SDNs enable cloud architectures through automated and scalable application distribution and mobility. Moreover, VFN increase flexibility and resource utilization on SDNs by means of data center virtualization. Thanks to the application of SDNs on IoT scenarios, it is possible to separate the data plane from the network control plane and introduce a logically centralized control plane, called a controller, to abstract control functions from networking.

Programmable control mechanisms of software-defined networks make them an alternative for reducing the complexity of Edge Computing (EC) architectures by enabling more efficient use of available computing resources. By using SDNs the data traffic originating from Edge servers can be dynamically routed freeing Edge devices from the execution of complex network activities such as service detection, orchestration, and QoS (performance-delay) requirements.

However, these advantages are accompanied by new challenges in terms of managing virtual resources. In this sense, it is necessary to develop intelligent mechanisms that allow the automated and dynamic management of the virtual communications established in the SDNs by the different user nodes. There are different proposals in this regard based on machine learning, such as genetic algorithms or deep learning. Recently, new approaches based on Deep Reinforcement Learning have demonstrated to offer great potential in solving this challenge, especially due to the advantages of not needing previous training data.

Future work includes the implementation and validation of the Global Edge Computing Architecture (GECA) 2.0, with the possibility of implementing Software-Defined Networks, as well as Network Function Virtualization. Thanks to the modular and scalable design of the architecture, the introduction of the new sub-layers and components is a straightforward process. The Deep Q-Networks will be implemented, compared and tested in the laboratory. After that, an Industry 4.0 platform will be deployed using the new version of the architecture in a real scenario (mixed dairy farm) where GECA has been formerly applied [6] to validate the new intelligent mechanism.

Also, in this first version of GECA 2.0 has been considered a solution in which the *SDN Controller* is centralized in the Cloud. However, as future work we will investigate solutions in which the training of the Q-Networks is carried out in the Cloud, but GECA 2.0 is provided with a mechanism to transfer the models to the *Fog Forwarding Gateways* and *Edge Gateways*. This will allow the reconfiguration of the routing tables in the nodes to be carried out in a distributed way and with less delay, being more appropriate in scenarios where reconfiguration speed and response times are critical. Moreover, future work also includes the research and implementation of new components within the GECA 2.0 reference architecture, including the development of its NFV features, such as the *Management and Orchestration* (MANO), as well as intelligent mechanisms based on Edge Computing and Machine Learning to implement the provisioning and configuration of the VNFs and the Edge layer.

Author Contributions: Conceptualization, R.S.A., I.S.-C. and R.C.-V.; Investigation, R.S.A., I.S.-C. and R.C.-V.; Methodology, J.P. and J.M.C.; Writing—original draft, R.S.A., I.S.-C. and R.C.-V.; Writing—review and editing, J.P. and J.M.C. All authors have read and agreed to the published version of the manuscript.

Funding: This work has been partially supported by the European Regional Development Fund (ERDF) through the Interreg Spain-Portugal V-A Program (POCTEP) under grant 0677_DISRUPTIVE_2_E (Intensifying the activity of Digital Innovation Hubs within the PocTep region to boost the development of disruptive and last generation ICTs through cross-border cooperation). Inés Sittón-Candanedo has been supported by scholarship program: IFARHU-SENACYT (Government of Panama).

Acknowledgments: Some icons in Figures by Copyright (c) 2018 Sandro Pereira (under MIT License).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Radanliev, P.; De Roure, D.C.; Nicolescu, R.; Huth, M.; Montalvo, R.M.; Cannady, S.; Burnap, P. Future developments in cyber risk assessment for the internet of things. *Comput. Ind.* **2018**, *102*, 14–22. [CrossRef]
2. De la Prieta, F.; Rodríguez-González, S.; Chamoso, P.; Corchado, J.M.; Bajo, J. Survey of agent-based cloud computing applications. *Future Gener. Comput. Syst.* **2019**, *100*, 223–236. [CrossRef]
3. Mohammadi, M.; Al-Fuqaha, A.; Sorour, S.; Guizani, M. Deep learning for IoT big data and streaming analytics: A survey. *IEEE Commun. Surv. Tutor.* **2018**, *20*, 2923–2960. [CrossRef]
4. García, O.; Alonso, R.S.; Prieto, J.; Corchado, J.M. Energy Efficiency in Public Buildings through Context-Aware Social Computing. *Sensors* **2017**, *17*, 826. [CrossRef]
5. Shi, W.; Schahram, D. The promise of Edge Computing. *Computer* **2016**, *49*, 78–81. [CrossRef]
6. Alonso, R.S.; Sittón-Candanedo, I.; García, Ó.; Prieto, J.; Rodríguez-González, S. An intelligent Edge-IoT platform for monitoring livestock and crops in a dairy farming scenario. *Ad Hoc Netw.* **2020**, *98*, 102047. [CrossRef]
7. Granelli, F.; Gebremariam, A.A.; Usman, M.; Cugini, F.; Stamati, V.; Alitska, M.; Chatzimisios, P. Software defined and virtualized wireless access in future wireless networks: Scenarios and standards. *IEEE Commun. Mag.* **2015**, *53*, 26–34. [CrossRef]
8. Puente Fernández, J.A.; García Villalba, L.J.; Kim, T.H. Software Defined Networks in Wireless Sensor Architectures. *Entropy* **2018**, *20*, 225. [CrossRef]
9. Alenezi, M.; Almustafa, K.; Meerja, K.A. Cloud based SDN and NFV architectures for IoT infrastructure. *Egypt. Inform. J.* **2019**, *20*, 1–10. [CrossRef]
10. Leike, J.; Krueger, D.; Everitt, T.; Martic, M.; Maini, V.; Legg, S. Scalable agent alignment via reward modeling: A research direction. *arXiv* **2018**, arXiv:1811.07871.
11. Sittón-Candanedo, I.; Alonso, R.S.; Corchado, J.M.; Rodríguez-González, S.; Casado-Vara, R. A review of edge computing reference architectures and a new global edge proposal. *Future Gener. Comput. Syst.* **2019**, *99*, 278–294. [CrossRef]
12. Sittón-Candanedo, I.; Alonso, R.S.; García, Ó.; Muñoz, L.; Rodríguez-González, S. Edge computing, iot and social computing in smart energy scenarios. *Sensors* **2019**, *19*, 3353. [CrossRef] [PubMed]
13. Alonso, R.S.; Sittón-Candanedo, I.; Rodríguez-González, S.; García, Ó.; Prieto, J. A Survey on Software-Defined Networks and Edge Computing over IoT. In *Highlights of Practical Applications of Survivable Agents and Multi-Agent Systems*; The PAAMS Collection; Communications in Computer and Information Science; De La Prieta, F., González-Briones, A., Pawleski, P., Calvaresi, D., Del Val, E., Lopes, F., Julian, V., Osaba, E., Sánchez-Iborra, R., Eds.; Springer: Cham, Switzerland, 2019; pp. 289–301. [CrossRef]
14. Van Hasselt, H.; Guez, A.; Silver, D. Deep reinforcement learning with double q-learning. In Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, Phoenix, AZ, USA, 12–17 February 2016.
15. Alonso, R.S.; Tapia, D.I.; Bajo, J.; García, Ó.; de Paz, J.F.; Corchado, J.M. Implementing a hardware-embedded reactive agents platform based on a service-oriented architecture over heterogeneous wireless sensor networks. *Ad Hoc Netw.* **2013**, *11*, 151–166. [CrossRef]
16. Ko, H.; Bae, K.; Marreiros, G.; Kim, H.; Yoe, H.; Ramos, C. A Study on the Key Management Strategy for Wireless Sensor Networks. *ADCAIJ Adv. Distrib. Comput. Artif. Intell. J.* **2015**, *3*, 43–53. [CrossRef]
17. Alonso, R.S.; García, Ó.; Saavedra, A.; Tapia, D.I.; de Paz, J.F.; Corchado, J.M. Heterogeneous wireless sensor networks in a tele-monitoring system for homecare. In Proceedings of the International Work-Conference

on Artificial Neural Networks, Limassol, Cyprus, 14–17 September 2009; Springer: Berlin, Germany, 2009; pp. 663–670.
18. García, Ó.; Alonso, R.S.; Tapia, D.I.; Corchado, J.M. Electrical power consumption monitoring in hotels using the n-core platform. In Proceedings of the 2016 Clemson University Power Systems Conference (PSC), Clemson, CA, USA, 8–11 March 2016; IEEE: Piscataway, NJ, USA, 2016; pp. 1–6.
19. Ashton, K. That 'internet of things' thing. *RFID J.* **2009**, *22*, 97–114.
20. Srinidhi, N.N.; Dilip Kumar, S.M.; Venugopal, K.R. Network optimizations in the Internet of Things: A review. *Eng. Sci. Technol. Int. J.* **2018**, *22*, 1–21. [CrossRef]
21. Kethareswaran, V.; Ram, C.S. An Indian Perspective on the adverse impact of Internet of Things (IoT). *ADCAIJ Adv. Distrib. Comput. Artif. Intell. J.* **2017**, *6*, 35–40. [CrossRef]
22. Chamoso, P.; Prieta, F.D.L. Swarm-Based Smart City Platform: A Traffic Application. *ADCAIJ Adv. Distrib. Comput. Artif. Intell. J.* **2015**, *4*, 89–98. [CrossRef]
23. González-Briones, A.; De La Prieta, F.; Mohamad, M.; Omatu, S.; Corchado, J. Multi-agent systems applications in energy optimization problems: A state-of-the-art review. *Energies* **2018**, *11*, 1928. [CrossRef]
24. Chamoso, P.; González-Briones, A.; Rodríguez, S.; Corchado, J.M. Tendencies of Technologies and Platforms in Smart Cities: A State-of-the-Art Review. *Wirel. Commun. Mob. Comput.* **2018.** [CrossRef]
25. Sittón-Candanedo, I.; Hernández-Nieves, E.; Rodríguez-González, S.; Santos-Martín, M.T.; González-Briones, A. Machine learning predictive model for industry 4.0. In Proceedings of the International Conference on Knowledge Management in Organizations, Žilina, Slovakia, 6–10 August 2018; Springer: Berlin, Germany, 2018; pp. 501–510.
26. Liao, Y.; Loures, E.d.F.R.; Deschamps, F. Industrial Internet of Things: A systematic literature review and insights. *IEEE Internet Things J.* **2018**, *5*, 4515–4525. [CrossRef]
27. Trappey, A.J.; Trappey, C.V.; Govindarajan, U.H.; Chuang, A.C.; Sun, J.J. A review of essential standards and patent landscapes for the Internet of Things: A key enabler for Industry 4.0. *Adv. Eng. Inform.* **2017**, *33*, 208–229. [CrossRef]
28. Rodríguez, S.; De Paz, J.F.; Villarrubia, G.; Zato, C.; Bajo, J.; Corchado, J.M. Multi-agent information fusion system to manage data from a WSN in a residential home. *Inf. Fusion* **2015**, *23*, 43–57. [CrossRef]
29. Sittón, I.; Rodríguez, S. Pattern extraction for the design of predictive models in industry 4.0. In Proceedings of the International Conference on Practical Applications of Agents and Multi-Agent Systems, Porto, Portugal, 21–23 June 2017; Springer: Berlin, Germany, 2017; pp. 258–261.
30. Yu, W.; Liang, F.; He, X.; Hatcher, W.G.; Lu, C.; Lin, J.; Yang, X. A Survey on the Edge Computing for the Internet of Things. *IEEE Access* **2017**, *6*, 6900–6919. [CrossRef]
31. Sanchez-Iborra, R.; Sanchez-Gomez, J.; Skarmeta, A. Evolving IoT networks by the confluence of MEC and LP-WAN paradigms. *Future Gener. Comput. Syst.* **2018**, *88*, 199–208. [CrossRef]
32. Shi, W.; Cao, J.; Zhang, Q.; Li, Y.; Xu, L. Edge Computing: Vision and Challenges. *IEEE Internet Things J.* **2016**, *3*, 637–646. [CrossRef]
33. Sittón-Candanedo, I.; Alonso, R.S.; García, Ó.; Gil, A.B.; Rodríguez-González, S. A Review on Edge Computing in Smart Energy by means of a Systematic Mapping Study. *Electronics* **2020**, *9*, 48. [CrossRef]
34. FAR-EDGE-P. *H2020 FAR-EDGE Projec*t; Factory Automation Edge Computing Operating System Reference Implementation (FAR-EDGE): Rome, Italy, 2017.
35. INTEL-SAP. IoT Joint Reference Architecture from Intel and SAP. Available online: https://www.intel.com/content/dam/www/public/us/en/documents/reference-architectures/sap-iot-reference-architecture.pdf (accessed on 15 May 2020).
36. Edge Computing Consortium; Alliance of Industrial Internet. *Edge Computing Reference Architecture 2.0*; Technical Report; Edge Computing Consortium and Alliance of Industrial Internet: Beijing, China, 2017.
37. Malik, A.; Roop, P.S.; Allen, N.; Steger, T. Emulation of cyber-physical systems using IEC-61499. *IEEE Trans. Ind. Inform.* **2018**, *14*, 380–389. [CrossRef]
38. Tseng, M.; Canaran, T.E.; Canaran, L. *Introduction to Edge Computing in IIoT*; Technical Report; Industrial Internet Consortium: Needham, MA, USA, 2018.
39. Jararweh, Y.; Doulat, A.; AlQudah, O.; Ahmed, E.; Al-Ayyoub, M.; Benkhelifa, E. The future of mobile cloud computing: integrating cloudlets and mobile edge computing. In Proceedings of the 2016 23rd International Conference on Telecommunications (ICT), Thessaloniki, Greece, 16–18 May 2016; IEEE: Piscataway, NJ, USA, 2016; pp. 1–5.

40. Caraguay, Á.L.V.; González, P.L.; Tandazo, R.T.; López, L.I.B. SDN/NFV Architecture for IoT Networks. In Proceedings of the 14th International Conference on Web Information Systems and Technologies (WEBIST 2018), Seville, Spain, 18–20 September 2018; pp. 425–429.
41. Sabella, D.; Vaillant, A.; Kuure, P.; Rauschenbach, U.; Giust, F. Mobile-edge computing architecture: The role of MEC in the Internet of Things. *IEEE Consum. Electron. Mag.* **2016**, *5*, 84–91. [CrossRef]
42. Yang, M.; Li, Y.; Jin, D.; Zeng, L.; Wu, X.; Vasilakos, A.V. Software-Defined and Virtualized Future Mobile and Wireless Networks: A Survey. *Mob. Netw. Appl.* **2015**, *20*, 4–18. [CrossRef]
43. Jammal, M.; Singh, T.; Shami, A.; Asal, R.; Li, Y. Software defined networking: State of the art and research challenges. *Comput. Netw.* **2014**, *72*, 74–98. [CrossRef]
44. Sezer, S.; Scott-Hayward, S.; Chouhan, P.K.; Fraser, B.; Lake, D.; Finnegan, J.; Viljoen, N.; Miller, M.; Rao, N. Are we ready for SDN? Implementation challenges for software-defined networks. *IEEE Commun. Mag.* **2013**, *51*, 36–43. [CrossRef]
45. Kim, H.; Feamster, N. Improving network management with software defined networking. *IEEE Commun. Mag.* **2013**, *51*, 114–119. [CrossRef]
46. Baek, J.Y.; Kaddoum, G.; Garg, S.; Kaur, K.; Gravel, V. Managing fog networks using reinforcement learning based load balancing algorithm. In Proceedings of the 2019 IEEE Wireless Communications and Networking Conference (WCNC), Marrakech, Morocco, 15–18 April 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 1–7.
47. Sampaio, L.S.; Faustini, P.H.; Silva, A.S.; Granville, L.Z.; Schaeffer-Filho, A. Using NFV and reinforcement learning for anomalies detection and mitigation in SDN. In Proceedings of the 2018 IEEE Symposium on Computers and Communications (ISCC), Natal, Brazil, 25–28 June 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 00432–00437.
48. Ruiz, L.; Durán, R.J.; De Miguel, I.; Khodashenas, P.S.; Pedreno-Manresa, J.J.; Merayo, N.; Aguado, J.C.; Pavon-Marino, P.; Siddiqui, S.; Mata, J.; et al. A genetic algorithm for VNF provisioning in NFV-Enabled Cloud/MEC RAN architectures. *Appl. Sci.* **2018**, *8*, 2614. [CrossRef]
49. Pei, J.; Hong, P.; Li, D. Virtual network function selection and chaining based on deep learning in SDN and NFV-enabled networks. In Proceedings of the 2018 IEEE International Conference on Communications Workshops (ICC Workshops), Kansas City, MO, USA, 20–24 May 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 1–6.
50. Hu, F.; Hao, Q.; Bao, K. A Survey on Software-Defined Network and OpenFlow: From Concept to Implementation. *IEEE Commun. Surv. Tutor.* **2014**, *16*, 2181–2206. [CrossRef]
51. Jagadeesan, N.A.; Krishnamachari, B. Software-Defined Networking Paradigms in Wireless Networks: A Survey. *ACM Comput. Surv.* **2014**, *47*, 27:1–27:11. [CrossRef]
52. Mijumbi, R.; Serrat, J.; Gorricho, J.; Bouten, N.; Turck, F.D.; Boutaba, R. Network Function Virtualization: State-of-the-Art and Research Challenges. *IEEE Commun. Surv. Tutor.* **2016**, *18*, 236–262. [CrossRef]
53. Jararweh, Y.; Doulat, A.; Darabseh, A.; Alsmirat, M.; Al-Ayyoub, M.; Benkhelifa, E. SDMEC: Software defined system for mobile edge computing. In Proceedings of the 2016 IEEE International Conference on Cloud Engineering Workshop (IC2EW), Berlin, Germany, 4–8 April 2016; IEEE: Piscataway, NJ, USA, 2016; pp. 88–93.
54. Salman, O.; Elhajj, I.; Kayssi, A.; Chehab, A. Edge computing enabling the Internet of Things. In Proceedings of the 2015 IEEE 2nd World Forum on Internet of Things (WF-IoT), Milan, Italy, 14–16 December 2015; IEEE: Piscataway, NJ, USA, 2015; pp. 603–608.
55. Baktir, A.C.; Ozgovde, A.; Ersoy, C. How Can Edge Computing Benefit from Software-Defined Networking: A Survey, Use Cases, and Future Directions. *IEEE Commun. Surv. Tutor.* **2017**, *19*, 2359–2391. [CrossRef]
56. Pang, Z.; Sun, L.; Wang, Z.; Tian, E.; Yang, S. A Survey of Cloudlet Based Mobile Computing. In Proceedings of the 2015 International Conference on Cloud Computing and Big Data (CCBD), Shanghai, China, 4–6 November 2015; pp. 268–275. [CrossRef]
57. Monfared, S.; Bannazadeh, H.; Leon-Garcia, A. Software defined wireless access for a two-tier cloud system. In Proceedings of the 2015 IFIP/IEEE International Symposium on Integrated Network Management (IM), Ottawa, ON, USA, 11–15 May 2015; pp. 566–571. [CrossRef]
58. He, Y.; Yu, F.R.; Zhao, N.; Leung, V.C.; Yin, H. Software-defined networks with mobile edge computing and caching for smart cities: A big data deep reinforcement learning approach. *IEEE Commun. Mag.* **2017**, *55*, 31–37. [CrossRef]

59. Sallab, A.E.; Abdou, M.; Perot, E.; Yogamani, S. Deep reinforcement learning framework for autonomous driving. *Electron. Imaging* **2017**, *2017*, 70–76. [CrossRef]
60. Suerich, D.; Young, T. Machine Learning for Optimized Scheduling in Complex Semiconductor Equipment. In Proceedings of the 2019 30th Annual SEMI Advanced Semiconductor Manufacturing Conference (ASMC), Saratoga Springs, NY, USA, 6–9 May 2019; IEEE: Piscataway, NJ, USA, 2019, pp. 1–4.
61. Deng, Y.; Bao, F.; Kong, Y.; Ren, Z.; Dai, Q. Deep direct reinforcement learning for financial signal representation and trading. *IEEE Trans. Neural Netw. Learn. Syst.* **2016**, *28*, 653–664. [CrossRef] [PubMed]
62. Ma, L.; Zhang, Z.; Ko, B.; Srivatsa, M.; Leung, K.K. Resource management in distributed SDN using reinforcement learning. In *Ground/Air Multisensor Interoperability, Integration, and Networking for Persistent ISR IX*; International Society for Optics and Photonics: Bellingham, WA, USA, 2018; Volume 10635, p. 106350M.
63. Amiri, R.; Almasi, M.A.; Andrews, J.G.; Mehrpouyan, H. Reinforcement learning for self organization and power control of two-tier heterogeneous networks. *IEEE Trans. Wirel. Commun.* **2019**, *18*, 3933–3947. [CrossRef]
64. Silver, D.; Huang, A.; Maddison, C.J.; Guez, A.; Sifre, L.; Van Den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; et al. Mastering the game of Go with deep neural networks and tree search. *Nature* **2016**, *529*, 484. [CrossRef] [PubMed]
65. Silver, D.; Schrittwieser, J.; Simonyan, K.; Antonoglou, I.; Huang, A.; Guez, A.; Hubert, T.; Baker, L.; Lai, M.; Bolton, A.; et al. Mastering the game of go without human knowledge. *Nature* **2017**, *550*, 354–359. [CrossRef] [PubMed]
66. Zhang, Q.; Lin, M.; Yang, L.T.; Chen, Z.; Khan, S.U.; Li, P. A double deep Q-learning model for energy-efficient edge scheduling. *IEEE Trans. Serv. Comput.* **2018**, *12*, 739–749. [CrossRef]
67. Ferdowsi, A.; Saad, W. Deep learning for signal authentication and security in massive internet-of-things systems. *IEEE Trans. Commun.* **2018**, *67*, 1371–1387. [CrossRef]
68. Watkins, C.J.; Dayan, P. Q-learning. *Mach. Learn.* **1992**, *8*, 279–292. [CrossRef]
69. Arulkumaran, K.; Deisenroth, M.P.; Brundage, M.; Bharath, A.A. Deep reinforcement learning: A brief survey. *IEEE Signal Process. Mag.* **2017**, *34*, 26–38. [CrossRef]
70. Melo, F.S. *Convergence of Q-Learning: A Simple Proof*; Institute for Systems and Robotics, Instituto Superior Técnico: Lisboa, Portugal, 2001.
71. Amiri, R.; Mehrpouyan, H.; Fridman, L.; Mallik, R.K.; Nallanathan, A.; Matolak, D. A machine learning approach for power allocation in HetNets considering QoS. In Proceedings of the 2018 IEEE International Conference on Communications (ICC), Kansas City, MO, USA, 20–24 May 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 1–7.
72. Liu, Y.; Yang, C.; Jiang, L.; Xie, S.; Zhang, Y. Intelligent Edge Computing for IoT-Based Energy Management in Smart Cities. *IEEE Netw.* **2019**, *33*, 111–117. [CrossRef]
73. Kim, J.; Kim, J.; Lee, S.; Park, J.; Hahn, M. Vowel based voice activity detection with LSTM recurrent neural network. In Proceedings of the 8th International Conference on Signal Processing Systems, Urumqi, China, 20–22 July 2019; ACM: New York, NY, USA, 2016; pp. 134–137.
74. Mu, T.Y.; Al-Fuqaha, A.; Shuaib, K.; Sallabi, F.M.; Qadir, J. SDN flow entry management using reinforcement learning. *ACM Trans. Auton. Adapt. Syst. (TAAS)* **2018**, *13*, 1–23. [CrossRef]
75. De Oliveira, R.L.S.; Schweitzer, C.M.; Shinoda, A.A.; Prete, L.R. Using mininet for emulation and prototyping software-defined networks. In Proceedings of the 2014 IEEE Colombian Conference on Communications and Computing (COLCOM), Bogota, Colombia, 4–6 June 2014; IEEE: Piscataway, NJ, USA, 2014; pp. 1–6.
76. Tang, J. *Intelligent Mobile Projects with TensorFlow: Build 10+ Artificial Intelligence Apps Using TensorFlow Mobile and Lite for IOS, Android, and Raspberry Pi*; Packt Publishing Ltd.: Birmingham, UK, 2018.
77. AlEroud, A.; Alsmadi, I. Identifying cyber-attacks on software defined networks: An inference-based intrusion detection approach. *J. Netw. Comput. Appl.* **2017**, *80*, 152–164. [CrossRef]
78. Casado-Vara, R.; González-Briones, A.; Prieto, J.; Corchado, J.M. Smart Contract for Monitoring and Control of Logistics Activities: Pharmaceutical Utilities Case Study. In *Advances in Intelligent Systems and Computing Proceedings of the International Joint Conference SOCO'18-CISIS'18-ICEUTE'18, San Sebastián, Spain, 6–8 June 2018*; Graña, M., López-Guede, J.M., Etxaniz, O., Herrero, Á., Sáez, J.A., Quintián, H., Corchado, E., Eds.; Springer: Berlin, Germany, 2019; pp. 509–517.
79. Hasselt, H.V. Double Q-learning. In *Advances in Neural Information Processing Systems 23, Proceedings of the 23rd International Conference on Neural Information Processing Systems, Vancouver, Canada, 6-11 December 2010*;

Lafferty, J.D., Williams, C.K.I., Shawe-Taylor, J., Zemel, R.S., Culotta, A., Eds.; Neural Information Processing Systems: San Diego, CA, USA, 2010; pp. 2613–2621.

80. Shoeibi, N.; Shoeibi, N. Future of Smart Parking: Automated Valet Parking Using Deep Q-Learning. In Proceedings of the International Symposium on Distributed Computing and Artificial Intelligence, Avila, Spain, 26–28 June 2019; Springer: Berlin, Germany, 2019; pp. 177–182.
81. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.A.; Veness, J.; Bellemare, M.G.; Graves, A.; Riedmiller, M.; Fidjeland, A.K.; Ostrovski, G.; et al. Human-level control through deep reinforcement learning. *Nature* **2015**, *518*, 529. [CrossRef] [PubMed]