network

Article

Formal Algebraic Model of an Edge Data Center with a Redundant Ring Topology

Pedro Juan Roig [1,*], Salvador Alcaraz [1], Katja Gilly [1], Cristina Bernad [1] and Carlos Juiz [2]

[1] Computer Engineering Department, Miguel Hernández University, 03202 Elche, Spain
[2] Mathematics and Computer Science Department, University of the Balearic Islands, 07022 Palma de Mallorca, Spain
[*] Correspondence: proig@umh.es; Tel.: +34-9-6665-8388

Abstract: Data center organization and optimization presents the opportunity to try and design systems with specific characteristics. In this sense, the combination of artificial intelligence methodology and sustainability may lead to achieve optimal topologies with enhanced feature, whilst taking care of the environment by lowering carbon emissions. In this paper, a model for a field monitoring system has been proposed, where an edge data center topology in the form of a redundant ring has been designed for redundancy purposes to join together nodes spread apart. Additionally, a formal algebraic model of such a design has been exposed and verified.

Keywords: data center design; edge AI; Internet of Things; resource migration; toroidal topology

check for updates

Citation: Roig, P.J.; Alcaraz, S.; Gilly, K.; Bernad, C.; Juiz, C. Formal Algebraic Model of an Edge Data Center with a Redundant Ring Topology. *Network* **2023**, *3*, 142–157. https://doi.org/10.3390/network3010007

Academic Editors: Posco Fung Po Tso, Takahiro Shinagawa, Kiho Lim and Alexey Vinel

Received: 22 November 2022
Revised: 12 January 2023
Accepted: 21 January 2023
Published: 30 January 2023

1. Introduction

To start with, artificial Intelligence (AI) could be formally defined as "a cross-disciplinary approach to understanding, modeling, and replicating intelligence and cognitive processes by invoking various computational, mathematical, logical, mechanical, and even biological principles and devices", as stated by Frankish and Ramsey in 2014 [1]. However, it is to be noted that, at this point, there is no widely accepted definition of AI, as well as that AI is still mathematically undefined after some 65 years since its foundation [2].

An interesting definition of an AI algorithm is "a set of rules or processes that aims to solve a problem or a task" [3]. In this sense, two types may be distinguished, such as static ones, where a fixed sequence of actions is performed, or otherwise, dynamic ones, where they can interact with the environment, thus being able to learn and evolve [4]. Those are usually called machine learning (ML) algorithms, which are further divided into supervised, unsupervised and reinforcement learning [5].

Basically, AI methods are just tools to analyze data, whereas ML might be seen as a meeting point between AI and statistics [6], providing ML solutions with the ability to achieve great results in basically any field, such as medicine [7], engineering [8], science [9], accessibility [10], knowledge management [11], general learning [12] and even incidental learning [13].

ML may be seen as the capability to learn out of specific training data to automatically build a model in an analytical way so as to solve related tasks [14]. ML may be divided into shallow learning and deep learning (DL), where the latter usually refers to learning with neural networks with several hidden layers [15], whilst the rest of instances are included in the former [16], even though the boundary between them is a bit blurred as there is no consensus in the literature.

In fact, the development of DL marked a milestone in the ML paradigm [17], as 2010 is considered to be the transition from the pre-DL era to the DL era, where training compute doubled every 20 months approximately in the former, whilst in the latter, it did in about 6 months. Later on, 2015 is considered to be the transition from the DL era to the Large-Scale era, where training compute became 10- to 100-fold higher [18].

AI may be applied to enhance many types of processes by dynamically adjusting them. In this sense, this paper proposes a multipurpose field monitoring system, which may well be fit for agricultural purposes, where moving end devices such as drones, furnished with some sensors, will be flying around so as to check for undesired conditions as plagues, whilst fixed end devices, such as measuring poles, also equipped with some sensors, will be testing for environmental conditions so as to keep track of them.

The data collected by all sensors involved are going to be sent to an edge data center powered with AI in order to obtain faster processing, whereas sustainability is achieved by implementing computing resource migration among hosts in order to minimize the distance between each resource and its associated owner, thus reducing the power consumption, leading to lower the carbon footprint. Additionally, a formal algebraic model will be exposed and further verified.

Hence, the contribution of this paper is related to the establishment of a framework focused on field monitoring system in order to obtain its formal algebraic model from the point of view of distributed networking. First of all, its characteristics and functionality are described, and after that, it is outlined how to enhance its behavior by means of an AI tool. Afterwards, the focus is set on the topology of the data center proposed, and eventually, a formal algebraic model of the whole system is presented, followed by its formal algebraic verification.

The rest of the paper is organized as follows: first, Section 2 looks into the related work, after that, Section 3 introduces the basics of the system proposed; then, Section 4 outlines the design of an AI based algorithm for that system; next, Section 5 presents the redundant ring topology; afterwards, Section 6 exposes a formal algebraic model for the whole system; and Section 7 draws some final conclusions.

2. Related Work

Regarding the literature on modeling distributed systems, there have been many publications from different points of view. For instance, regarding numerical models, Smeliansky et al. proposed a model of a distributed computer system regarding both hardware and software [19], whereas Nenashev et al. presented a model for securing peer-to-peer distributed systems for processing and storage in enterprise networks [20], whilst Delporte et al. exposed a model for distributed network computing considering failures in the computation [21].

On the other hand, with respect to programming distributed models related to big data, Miller et al. presented a model that simplifies failure recovery by design [22], whilst Jiang et al. proposed a model to achieve efficient fault detection and isolation [23], whereas Gyongyosi et al. exposed a model for quantum computers [24].

In addition, Klein et al. developed an abstract mathematical model for information systems [25], whereas Saha et al. proposed a computing model to describe quantum computing [26], whilst Margara et al. presented a unifying model that dissects the core functionalities of data-intensive systems [27].

Centering on modeling distributed communication networks, Martinez et al. presented a model for the cost of a flexible network [28], considering both transient and stationary stages, whereas Irzaev et al. described a model for adapting traffic routing and packet queues policies [29], whilst Doyle et al. talks about the guidelines to undertake models for large scale networks [30].

Focusing on edge computing environments, Zhu et al. talked about resource relocation through a stacked auto-encoder model to predict car sharing demand [31], whereas Fan et al. referred to a scheme to minimize delay when it comes to offloading and resource allocation [32]. Likewise, Matthe et al. develops an approach for proactive self-adaptive systems [33], whilst Krupitzer et al. showed a collection of design patterns for self-adaptive systems to support adaptability in IoT systems [34].

Moreover, Wang et al. provided an edge-cloud computing model which circumvents the problem of dynamic decisions on execution location [35], whilst Amannejad presented an automated method to compare federated solutions with centralized trained models [36], whereas Latif et al. proposed a lightweight trust management model to manage the service level trust along with quality of service [37].

Furthermore, Toczé et al. presented a model of workload specifically for edge offloading tasks [38], whereas Pandian exposed an enhanced edge model for handling big data flows in the applications of Internet of Things [39], whilst Cavalieri d'Oro et al. described a model for critical edge computing systems based on queuing networks [40].

In addition, Allahham et al. exposed a model of reliability in extreme edge computing systems [41], whilst Choi et al. proposed a novel QoS prediction model for edge computing [42], whereas Li et al. described a way to improve robustness and efficiency of edge computing models [43].

Additionally, Berger et al. presented a real-time predictive control with digital twins and edge computing technologies [44], while Jiang et al. carried out a model-based comparison of cloud-edge computing resource allocation policies [45], whereas Sun et al. designed an optimal defense strategy model based on differential game in edge computing [46], whilst Jian et al. proposed an improved chaotic bat swarm scheduling learning model on edge computing [47].

Similarly, Aleksandrova et al. describes a machine learning model to obtain updates in edge computing based on optimal stopping theory [48], whilst Song et al. presented a programming model for reliable and efficient edge-based execution under resource variability [49], whereas Song et al. exposed a model-based fleet deployment of edge computing applications [50], while Tawalbeh et al. exhibited an edge enabled IoT system model for secure healthcare [51].

In addition, Pereira et al. proposed analytical models for availability evaluation of edge and fog computing nodes [52], whereas Wang et al. exposed an adaptive offloading scheme for multi-subtasks to multi-servers in edge environments [53], whilst Gadasin et al. exposed the issues of building a cluster model designed for edge computing [54], while Jia et al. proposed a multi-property method to evaluate trust of edge computing based on data driven capsule network [55].

Moreover, Sasaki et al. exposed an edge-cloud computing model for autonomous vehicles using a specific software platform [56], while Ibn-Kheder et al. presented an edge computing assisted autonomous driving model [57], whereas Valocky et al. exhibited an experimental autonomous car model with safety sensor in wireless networks [58], whilst Wei et al. proposed a model for cooperative perception in autonomous driving [59].

Anyway, it is to be said that some models make use of specific branches of mathematics when modeling distributed systems, such as Jaggard et al. focused on game theory to test the convergence of asynchronous dynamics [60], Baranawal et al. centered on graph theory to compose and execute time-dependent graph algoritms [61] or Bagchi et al. referred to topology in order to obtain insights in the field of fault detection and diagnosis [62].

With regards to algebra, some different approaches are taken, such as Makhortov et al. utilized LP structures for optimizing distributed knowledge management [63], Kakkavas et al. used algebraic tools so as to explore network tomography for scaling networks [64] or Fittipaldi et al. applied a discrete-time algebraic model for arbitrary network topology [65].

Likewise, Duarte proposed a characterization in terms of algebraic structures and topological spaces [66], whereas Letychevskyi et al. presents an algebraic approach to verify and test distributed systems [67], whilst Yuan et al. discussed the use of tensor relational algebra for distributed machine learning system design [68].

However, there are few instances of applying process algebras to obtain formal algebraic modeling of distributed networks, such as Gaur et al., which makes a survey about modelling of large distributed systems [69] or Roig et al., which focuses on edge computing domains [70], and that is the motivation of undertaking this current work.

3. Basics of the System Proposed

Let us suppose a square field with a given side length aimed at growing crops, which may easily be divided into four quadrants with their internal boundaries being the co-ordinate axes and their crossing point being the origin of coordinates, thus allowing the assignment of some bidimensional co-ordinates to the location of any given bush or plant. This way, such a field could be assimilated to the real plane with Euclidean distance, where each quadrant would have a unique combination of signs for both axes, the same way as it happens in trigonometry.

In this context, let us suppose four hosts ubicated in each of the corners therein, where every one of those may house diverse remote computing units, such as virtual machines (VMs), which may act as edge computing servers or fog computing servers, depending on their roles, so as to deal with the big data being generated by this whole ecosystem.

Each host, ubicated on each corner, is attached to a pole furnished with a 90-degree sectorial antenna emitting with enough power to completely cover its own quadrant, thus the signal radiated by one of them overlaps around the coordinate axes with those coming from the other ones, as depicted in Figure 1.

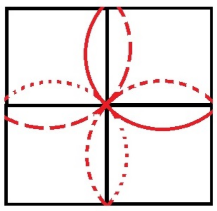

Figure 1. Draft of the square field and the coverage area of the sectorial antennas located in each corner.

This way, communications between any given pair of antennas may be undertaken by means of fiber optical runs, those being either monomode or multimode, depending on the distance among them or the requirements of quality of service, although other middle-range wireless technologies might also be used, such as WiMAX or LoRaWAN, even though diverse long-range wireless solutions may also work, such as cellular deployments or satellite networks. This type of traffic among hosts, or even among VMs, may be identified as control traffic, whereas the sort of traffic among sensors and VMs, or VMs and actuators, may be seen as data traffic.

Focusing on the VMs allocated within each host, which is associated to its particular antenna, there are two types of VMs to be considered herein, such as brokers and orchestrators. On the one hand, the former are dealing with data coming from end devices, being those collecting data from sensors, which could be moving ones, such as drones collecting pictures from diverse locations of the field, or fixed ones, such as environmental posts picking up information about temperature or humidity out of polling stations located around the terrain.

On the other hand, the latter are working with data furnished by the former, which could be seen as a higher hierarchical level entities, whose role is to influence the decisions made by the former. Attending to the origin of the data they operate with and also to their position in the remote computing hierarchy, the former might be considered as edge servers and the latter, as fog servers. In addition, those fog devices might make use of cloud servers as their backup for storaging data or offloading their processing tasks.

Hence, three levels within remote computing hierarchy are being established herein with different roles each, such as edge servers, also known as brokers, occupying the lower level, then the fog servers are above the edge ones, taking the middle level, and in turn, the cloud servers are above the fog ones, thus located on some ISP facilities and being placed on the top level.

Therefore, the four hosts located on each of the corners of the square field may allocate both the VMs acting as brokers and those playing the part of orchestrators. Within the brokers, the ones dealing with environmental data will be always standing in the same host, where basically there will be one situated on each host at all times so as to process the data being received from fixed sensors situated on its own quadrant.

On the contrary, brokers operating with drones will need to be migrated along to the quadrant where a drone is located at a given time so as to be the closest to it, thus reducing the power necessary for the wireless transmissions between the drone and its corresponding VM, and also lowering the latency, thus speeding up the processing times, as well as the energy expenses. Additionally, the orchestrator will also be allocated within any of the hosts, where at least a pair of them would need to be used to obtain two synchronized copies of that VM allocated in different hosts for redundancy purposes.

With respect to the operativity, let us suppose a drone acting as an edge device, whose role is to fly around the square field in order to take pictures of the crops being planted over there and their corresponding environment in order to spot plagues or traces of other harmful situations. This way, the drone would act as an end device, thus capturing the pictures obtained by the camera acting as a sensor, and in turn, forwarding those images to the proper broker, meaning the corresponding VM, in order for it to process them.

The messages between the drone, which acts as a publisher of data, and the edge server, playing the role of a broker, needs to be forwarded through a wireless technology, whose type will depend on the maximum distances within a quadrant, which occurs to be its diagonal, whose distance is given by the square root of 2 multiplied by half of the side length of the whole square. Hence, such a distance would establish the necessary wireless technology to be used, that being either short-range such as Wi-Fi or ZigBee, or middle-range such as WiMAX, or cellular technology for longer distances, or even satellite technology to access any type of environment.

Otherwise, the rest of the sensors may be located in fixed locations in order to measure some key environmental indicators, such as temperature or humidity, in certain parts of the field. In this case, wireless technologies may also be employed to communicate with the edge servers, although a better option would be to employ wired technologies such as fiber optical runs, which might be deployed so as to speed up such communications, even though some civil engineering infrastructure would need to be implemented. This way, the best solution is to evaluate the balance between the costs incurred and the benefits obtained in terms of throughput.

In summary, data obtained by a fixed sensor are forwarded to the corresponding VM acting as a broker, allocated in the host within the quadrant where such a sensor is ubicated, which will in turn process the data, and then, it will send a response to the appropriate actuator to do a certain action to alleviate that condition, or otherwise, it will not send any response whatsoever if there is no need to. Moreover, another VM acting as an orchestrator will supervise the operation and it will forward some extra information so as to influence the decision-making process of the broker.

Otherwise, data obtained by a moving sensor, which in this case are the pictures taken by the drone, are also forwarded to its corresponding VM acting as a broker, but its location will be into the quadrant where the drone is flying over. The processing of such data matches that exposed for the fixed sensor case, ending up with an appropriate response to the actuators so as to cope with the potential issue being detected, or otherwise, no response is expected in case no relevant event is detected.

However, the moving sensor case requires a handover action of the VM associated to the drone, which successfully transfers that VM from one host to another. The point where the handover needs to be made gets indicated by the fact of crossing the coordinate axes at any point, in a way that when a drone traverses any of those, then it happens that the distance to the host within the new quadrant is shorter than that to the old one, thus handover needs to be accomplished to revert that situation.

In order to facilitate the handoff procedure, the signals coming from the antenna located on each corner covers their semiaxes acting as a boundary to separate a given quadrant to the other ones, hence there is an overlap of signals along each section of the coordinate axis. Hence, when a drone passes through a coordinate axis, which the drone knows because the power received from the antenna where its associated VM is currently allocated is lower than the power received from another antenna, it means that the drone has entered into a new quadrant. This very fact makes the drone send a control signal to its associated VM with the identification of the new quadrant, and upon receipt, the VM gets migrated from the current host, that being located within the current quadrant, to the new host, that being situated in the host within the new quadrant, thus getting the migration process completed.

In order to clarify the sequence of events of the handoff process, the following list is given, whose scheme is exhibited in Figure 2:

1. The drone sends a message indicating the number of quadrant it got into to the current host;
2. The current host asks for resources to the new host to perform the VM migration;
3. The new host informs positively about having enough resources (supposing it has enough room left);
4. The current host proceeds to migrate the associate VM of the drone to the new host;
5. The new host acknowledges the successful migration to the current host;
6. The current host acknowledge the successful migration to the drone;
7. The drone sends a hello message to the new host, where its VM is now located
8. The new host acknowledges that message.

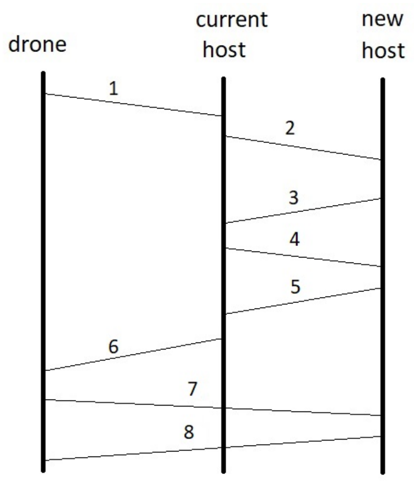

Figure 2. Migration diagram.

Regarding the application layer protocol, MQTT may be a convenient solution due to its lightweigthness and the fact of allowing subjects to label the different messages exchanged so as to quickly distinguish among pictures, temperature or humidity, although CoAP may also be an interesting solution [71].

4. Implementing AI on the System Proposed

AI may be implemented in any of the three kinds of remote computing levels exposed, such as on edge servers, fog servers and cloud servers. As explained above, the systems proposed only use the cloud premises for backup and offload services, thus the application of AI techniques at that level makes improve the backup competences and the offload capabilities, hence getting better performance for the overall system. In addition, AI may export some proper data from the cloud level to the lower levels in order to influence their performance.

For instance, there may be a coefficient c_{c1} associated with the storage backup process, whose value may be *nil* if such a process in the cloud premises are being undertaken at full extent, whereas such a value might obtain a non-zero value if some kind of congestion is detected, where that value might grow as the level of congestion does. Analogously, another coefficient c_{c2} may become associated with the offloading capabilities, whose functionality may follow the same pattern of behavior as the former. It is to be remarked that non-zero values of any of both coefficients may penalize the VM migration process among fog servers, as it is a high-resource consumption operation and congestion on the cloud level might affect the backup and offloading tasks right after the migration is conducted.

Centering on the fog level, the application of AI techniques may induce some coefficients about the VM migration processes referred to orchestrators, such as c_{f1i} gets associated with the occupation rate of resources in host $i = \{1 \cdots 4\}$, such as their values may stand to *nil* in cases of low occupation, and it may grow as such a rate increases.

In addition, another coefficient c_{f2} may be associated to the fact that some interhost link may be down, thus increasing the traffic flows going through the rest of the links, leading to a higher probability of saturation of those links. Similarly, another coefficient c_{f3} may be defined for the case of having some host down, thus rising the traffic being cursed for the rest of hosts.

Focusing on the edge level, applying AI techniques may incorporate some coefficients regarding VM migration processes referred to brokers, resulting in some of them being analogous as those in the fog case, such as c_{e1i} being related to the occupation rate of host $i = \{1 \cdots 4\}$, c_{e2} being attached to some interhost link down, and c_{e3} being associated with having some host down. Additionally, c_{e4} being related to a high error bit rate in the air interface due to interferences, or c_{e5} being associated with some error in any sensor or actuator.

All of those coefficients are related to the system itself, even though other external parameters might also be taken into account, such as current network load, stated by c_n, historical data, given by c_h, variation over the baseline, cited by c_b, or network intelligence data, quoted by c_d.

As a summary, here they are the list of coefficients conveniently grouped.

- Edge: c_{c1}, c_{c2}
- Fog: c_{f11}, c_{f12}, c_{f13}, c_{f14}, c_{f2}, c_{f3}
- Cloud: c_{e11}, c_{e12}, c_{e13}, c_{e14}, c_{e2}, c_{e3}, c_{e4}, c_{e5}
- Others: c_n, c_h, c_b, c_d

Eventually, the target is to craft an objective function, whose aim is to search for the goal proposed, whilst applying some penalties referred to all of the aforesaid coefficients c_j, each one being multiplied by a certain particular value λ_j, established by an AI tool after the appropriate training, as exhibited in (1), where the goal is obtained to the fullest if no penalization exist, although the goal gets reduced in a given percentage related to the weighted sum of the value of the coefficients not being zero.

$$objectiveFunction \;=\; (1 - \sum_j \lambda_j c_j) \cdot goal \tag{1}$$

The initial training stage may be used to tune up the AI with a specific dataset, considering that coefficients have a value within the interval $[0, 1]$, where the importance of each error influences the value of its related coefficient, and that value will be dynamically adjusted by the AI tool being employed. The target is obviously to have the system duly adjusted, where in such a case, all coefficients will be worth 0, and so will the sum of all penalties, thus the goal will be achieved at full extent, that being 100%. Otherwise, it will accumulate the weighted sum of all non-zero coefficients, thus reducing the goal accordingly.

Furthermore, regularization may be applied to the weighted sum of coefficients by means of a loss function, also known as cost function. This could be conducted in different ways, such as Lasso Regression (2), or otherwise, Ridge Regression (3), both being widely used in supervised ML for output prediction.

$$\textstyle\sum_{i=1}^{n}(y_i - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} |\beta_j| \tag{2}$$

$$\sum_{i=1}^{n}(y_i - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 \quad (3)$$

Looking at the former, if lambda is worth zero, the result matches the ordinary least square (OLS) method, which applies linear regression to fit a straight line through the data available, whilst if it is a large value, it makes coefficients zero, thus causing under-fit. On the other hand, taking the latter, if lambda is worth zero, OLS applies as in the former, whereas if it is a large value, it adds to much weight, leading to under-fit. Both methods focus on reducing model complexity and preventing over-fitting, which might arise out of applying simple linear regression.

5. Redundant Ring Topology

In view of that there are one physical host located in each of the corners of the field, the easiest way to link them through a wired infrastructure is by deploying a physical ring passing along each of the nodes. This way, every one of the n nodes being part of the ring will have just a predecessor node and a successor node, both being only one hop away but in different directions, whilst the remaining node will be two hops away, which may be seen as its opposite node, regardless the path going through any of its two neighbors, thus accounting for a pair of redundant paths to go from node i to its opposite one.

The more straightforward way to label those n nodes is by using modular arithmetic, where the n nodes are assigned a single value within the interval $[0 \cdots n-1]$, where one of them is selected to be node 0 and the rest of nodes go in a sequential order along the same direction. Hence, according to modular arithmetic, a given node i will have node $(i-1)_{|n}$ as its predecessor neighbor, whereas it will have node $(i+1)_{|n}$ as its successor neighbor, whilst the opposite node will be labeled as $(i+2)_{|n}$.

One of the advantages of implementing a ring is that there are always two redundant paths to go from a source node to a destination node, which basically avoids the existence of single points of failure. However, if one of the links go down, then the ring shape is converted into a bus shape, where its ends are just the nodes sharing the broken link. Hence, as those nodes do not share a direct link any more, such that the path between them needs to be conducted by going all around the ring, thus passing through the rest of the nodes, it might induce latency issues when dealing with critical infrastructures.

Therefore, a common solution in production environments is to deploy redundant rings, thus having a couple of links between a particular node and its clockwise neighbor and another couple towards its counterclockwise neighbor. Figure 3 depicts a redundant ring topology with n items, where it may be appreciated its shape of a toroidal array, as the elements on both ends are linked together, which may also be compared with the shape of a bead collar where the diverse nodes may be considered as its beads.

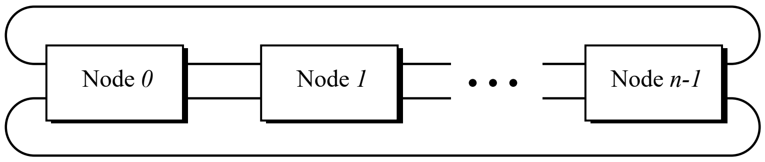

Figure 3. Redundant ring topology with n elements.

Regarding the port nomenclature, Figure 4 exposes its layout, where all ports are organized in a clockwise manner. This way, ports 0 and 1 are pointing to the successor node, whilst ports 2 and 3 are heading for the predecessor node.

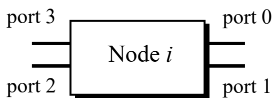

Figure 4. Port layout of a redundant ring node.

6. Formal Algebraic Model

In order to obtain a formal algebraic model of both the processing hierarchy and the resource migration between a couple of physical hosts, no matter whether the virtual resource is a broker or an orchestrator, it is to be mentioned that an abstract process algebra called Algebra of Communicating Processes (ACP) may be seen as a convenient solution, as it allows to just focus on relationships among objects, thus leaving aside the real nature of them [72]. It is to be said that ACP employs two atomic actions, such as send a message and receive a message, both having a generic content d, so as to express the behavior of objects, which is exhibited by means of algebraic expressions [73].

It is said that a bunch of atomic actions may become interrelated among them by means of operators [74]. One of the most commonly used is the sequential, which is a binary operator and is described as $\cdot$, whose meaning is that the first action is run, and in turn, the second action. Another common operator is the alternate, which is also a binary operator and is denoted as $+$, meaning the execution of either the first or the second action.

In addition, the merge operator admits a string of actions and is exposed as $||$, meaning the concurrent execution of all related actions. Moreover, the conditional operator presents a condition between two triangles pointing outwards, where the action on the left must be run if the condition is true, or otherwise, the action on the right must be executed in case it is false, thus following the scheme (True $\triangleleft$ condition $\triangleright$ False).

Those four operators are usually enough to algebraically describe the behavior of each object involved [75]. Furthermore, the encapsulation operator over a set H containing all internal atomic actions, described as ∂_H, brings those internal atomic actions to deadlock, whilst allowing internal communications to occur. After the application of this unary operator, a sequence of events is represented where all objects involved play their part.

At that point, the abstractor operator over a set I containing all internal communications, denoted as τ_I, masks internal communications, thus leaving only external atomic actions to happen. After applying that unary operator, the external behavior of the model gets unveiled, as the model is now presented as a black box, where only entries and exits are viewed.

Then, the external behavior of the real system is obtained, and afterwards, both external behaviors are confronted in order to check whether those share the same string of actions and the same branching structure, which leads to consider the model as rooted branching bisimilar to the real system, such that a rooted branching bisimulation equivalence relation between them both is established, which is a sufficient condition to obtain a verified model [76].

Therefore, the first step to obtain the intended formal algebraic model for the processing hierarchy is to quote the behavior of each of the objects involved in the system, as shown in Table 1. In order to keep the scenario as simple as possible, let us suppose just one fixed pole within each of the quadrants and only a drone moving around, where all those types of end devices simply have one sensor and one actuator, such as the former reads raw data from the environment and passes them on to an end device and the latter receives processed data from an end device and act in the environment according to those data.

When an end device receives raw data, it sends those to its broker, which either may process them and forward them to the proper actuator, or otherwise, pass them on to the orchestrator, which gets raw data transformed into processed data after undertaking the duly processing, and at that point, such data are sent back to the appropriate broker, which in turn, it sends them back to the proper end device. However, some tasks might require extra processing capabilities, and in such a case, the orchestrator must send raw data up to the cloud server for it to have the further processing conducted and send them back again, as in this model, the cloud is supposed to deal with all types of requests related to raw data.

Table 1. Objects taking part in the model proposed.

Object	Type	Location	Migrate
pole 1	fixed end device	quadrant 1	No
pole 2	fixed end device	quadrant 2	No
pole 3	fixed end device	quadrant 3	No
pole 4	fixed end device	quadrant 4	No
drone 1	moving end device	move around	No
broker 1	edge server	physical host	Yes
orchestrator 1	fog server	physical host	Yes
cloud 1	cloud server	cloud host	No

With all those pieces of information, here they are the models for the objects involved in the processing hierarchy. To start with, poles are identified as P_i, where $i = \{1 \cdots 4\}$, such that $AllPoles = 4$, and its model is exposed in (4), as it may just read raw data d from the environment $'env'$ and send it upwards to a broker u, or otherwise, read processed data e from a broker and send it to the environment so as to act on it. Hence, d goes upwards and e does downwards. Furthermore, channels between the environment and a pole, as well as between a pole and a broker are both bidirectional, where directions of the traffic flows involved are stated through an arrow, which applies to all models presented.

$$P_i = \sum_{i=1}^{AllPoles} \big(r_{env \to i}(d) \cdot s_{i \to u}(d) + r_{u \to i}(e) \cdot s_{i \to env}(e) \big) \cdot P_i \tag{4}$$

The drone is known as D_j, where j would account for further drones involved, even though just 1 is considered now, such that $AllDrones = 1$, whilst its model is expressed in (5), where its actions are those related to the pole.

$$D_j = \sum_{j=1}^{AllDrones} \big(r_{env \to j}(d) \cdot s_{j \to u}(d) + r_{u \to j}(e) \cdot s_{j \to env}(e) \big) \cdot D_j \tag{5}$$

The broker is called B_u, where u would make for future brokers involved, although only 1 is considered now, such that $AllBrokers = 1$, whereas its model is shown in (6). Basically, it may receive raw data d from any end device (that being either a pole i or a drone j), and in turn, it may either process them with the help of an AI tool named AI_B and send back the processed data e to that end device, or otherwise, send the raw data d up to an orchestrator v if such processing requires more computing power.

$$\begin{aligned} B_u = \textstyle\sum_{k=1}^{AllBrokers} \Big(\big(r_{i \to u}(d) + r_{j \to u}(d) \big) \cdot \\ \Big(\big(s_{u \to i}(e) + s_{u \to j}(e) \big) \triangleleft AI_B \triangleright s_{u \to v}(d) \Big) \Big) \cdot B_u \end{aligned} \tag{6}$$

The orchestrator is named O_v, where v would be kept for future orchestrators put in place, even though just 1 is accounted, such that $AllOrchs = 1$, whilst its model is depicted in (7). Its task is to receive raw data d from a broker, and then, it may process them by using an AI tool called AI_O, that being more powerful that the one being shown for the brokers, and in turn, processed data e are forwarded to the proper broker. In addition, that processed data might also be sent up to a cloud server w just in case the processing power needed is not enough.

$$O_v = \sum_{v=1}^{AllOrchs} \Big(r_{u \to v}(d) \cdot \big(s_{v \to w}(e) \triangleleft AI_O \triangleright s_{v \to u}(d) \big) \Big) \cdot O_v \tag{7}$$

The cloud is addressed as C_w, where n would be used if more than one cloud server were employed, although only 1 is considered now, in a way that $AllClouds = 1$, whereas its model is exhibited in (8). Its responsability is to process the raw data d being sent from an orchestrator and process them by means of an AI tool branded AI_C, being more powerful than those shown so far, which results in processed data e being sent back to the appropriate orchestrator.

$$C_w = \sum_{w=1}^{AllClouds} \big(r_{v \to w}(d) \cdot AI_C \cdot s_{w \to v}(e) \big) \cdot C_w \tag{8}$$

Once all objects involved in the scenario have been modeled, then it is time to put them all in a concurrent manner, and then, apply the encapsulation operator so as to achieve the sequence of events, such as exposed in (9). It is to be noted that the encapsulation operator transforms internal atomic actions in the same channel into communications, whereas it brings the rest of internal atomic actions to deadlock, thus being irrelevant for the model. Furthermore, the symbol $\emptyset$ represents that the model does nothing in one of the possible actions related to a conditional operator.

$$\begin{gathered}\textstyle\sum_{i=1}^{AllPoles} \sum_{j=1}^{AllDrones} \sum_{u=1}^{AllBrokers} \sum_{v=1}^{AllOrchs} \sum_{w=1}^{AllClouds} \\ \Big(\partial_H \big(P_i \mid\mid D_j \mid\mid B_u \mid\mid O_v \mid\mid C_w \big) \Big) = \\ \Big(\big(r_{env \to i}(d) \cdot c_{i \to u}(d) + r_{env \to j}(d) \cdot c_{j \to u}(d) \big) \cdot \\ \Big(\emptyset \lhd AI_B \rhd c_{u \to v}(d) \cdot \big(\emptyset \lhd AI_O \rhd c_{v \to w}(d) \cdot AI_C \cdot c_{w \to v}(e) \big) \cdot c_{v \to u}(e) \Big) \cdot \\ \big(c_{u \to i}(e) \cdot s_{i \to env}(e) + c_{u \to j}(e) \cdot s_{j \to env}(e) \big) \Big) \cdot \\ \Big(\partial_H \big(P_i \mid\mid D_j \mid\mid B_u \mid\mid O_v \mid\mid C_w \big) \Big)\end{gathered} \tag{9}$$

At this point, the abstraction operator may be applied in order to mask all internal communications, thus revealing its external behavior, as seen in (10).

$$\begin{gathered}\textstyle\sum_{i=1}^{AllPoles} \sum_{j=1}^{AllDrones} \sum_{u=1}^{AllBrokers} \sum_{v=1}^{AllOrchs} \sum_{w=1}^{AllClouds} \\ \Big(\tau_I \big(\partial_H \big(P_i \mid\mid D_j \mid\mid B_u \mid\mid O_v \mid\mid C_w \big) \big) \Big) = \\ \Big(\big(r_{env \to i}(d) + r_{env \to j}(d) \big) \cdot \big(s_{i \to env}(e) + s_{j \to env}(e) \big) \Big) \cdot \\ \Big(\tau_I \big(\partial_H \big(P_i \mid\mid D_j \mid\mid B_u \mid\mid O_v \mid\mid C_w \big) \big) \Big)\end{gathered} \tag{10}$$

On the other hand, the external behavior of the real model is shown in (11).

$$X = \Big(\big(r_{env \to i}(d) + r_{env \to j}(d) \big) \cdot \big(s_{i \to env}(e) + s_{j \to env}(e) \big) \Big) \cdot X \tag{11}$$

Eventually, it seems clear that the external behavior of the model and that of the real system share the same string of actions and the same branching structure. Hence, they both may be considered to be rooted branching bisimilar, and therefore, that is a sufficient condition to verify a model, as shown in (12).

$$\textstyle\sum_{i=1}^{AllPoles} \sum_{j=1}^{AllDrones} \sum_{u=1}^{AllBrokers} \sum_{v=1}^{AllOrchs} \sum_{w=1}^{AllClouds} \Big(\tau_I \big(\partial_H \big(P_i \mid\mid D_j \mid\mid B_u \mid\mid O_v \mid\mid C_w \big) \big) \Big) \Leftrightarrow X \tag{12}$$

With respect to the formal algebraic model of the migration of computing resources, it is to be reminded that there are $n = 4$ nodes, going from 0 to 3, whereas each node has $p = 4$ ports. In case of the source node, ports 0 and 1 send traffic flows to the successor node, whilst ports 2 and 3 do to the predecessor node. Otherwise, in case of the destination node, ports 0 and 1 are waiting to receive traffic from the predecessor node, whereas ports 2 and 3 do from the successor node. It is to be noted that, in this model, atomic actions show their node identifiers first, and afterwards, their port identifiers, separated by a comma for clarification purposes.

Supposing a given node x, such as V_x, that being the source node of a migration, then the traffic flow associated to such an action may head for its successor if the node destination is $y = (x+1)_{|n}$, whereas it may lead to its predecessor if the node destination is $y = (x-1)_{|n}$, or otherwise, it may go any way if the node destination is $y = (x+2)_{|n}$, as both paths, namely clockwise and counterclockwise, are redundant with equal cost.

Hence, the formal algebraic model for the resource migration take all three possibilities, as exhibited in (13) for the node source V_x and in (14) for the node destination V_y. Regarding verification, it is obvious that send and read action clearly match, thus verification of this model is analogous to the previous one.

$$\begin{aligned} V_x = \textstyle\sum_{x=0}^{n-1} \Big(\textstyle\sum_{p=0}^{1} s_{x,p}(d) \triangleleft y = (x+1)_{|n} \triangleright \\ \big(\textstyle\sum_{p=2}^{3} s_{x,p}(d) \triangleleft y = (x-1)_{|n} \triangleright \textstyle\sum_{p=0}^{3} s_{x,p}(d) \big) \Big) \end{aligned} \tag{13}$$

$$\begin{aligned} V_y = \textstyle\sum_{y=0}^{n-1} \Big(\textstyle\sum_{p=2}^{3} r_{y,p}(d) \triangleleft y = (x+1)_{|n} \triangleright \\ \big(\textstyle\sum_{p=0}^{1} r_{y,p}(d) \triangleleft y = (x-1)_{|n} \triangleright \textstyle\sum_{p=0}^{3} r_{y,p}(d) \big) \Big) \end{aligned} \tag{14}$$

Additionally, some theorems could be derived from the aforementioned expressions in order to assure that the model works as expected.

Theorem 1. *All incoming messages from the sensors which are processed by the brokers are redirected out of the system and applied to the actuators.*

Proof of Theorem 1. Taking Equation (9), which exhibits the sequence of events of the model, it may be seen that information collected from the sensors in the environment (env) are taken by either the pole i $(r_{env \to i})$ or the drone j $(r_{env \to j})$, which in turn sends a raw message (d) to broker u (either $c_{i \to u}$ or otherwise $c_{j \to u}$). At that point, if the broker is able to process the raw message (AI_B condition is true), then the raw message (d) is transformed into the processed message (e), meaning that such a message is not forwarded to the orchestrator v, which is expressed by $\emptyset$. Afterwards, the broker b sends the processed message (e) to either the pole i $(c_{u \to i})$ or the drone j $(c_{u \to j})$, which in turn, is sent to the actuator so as to act on the environment (env) as $(s_{i \to env})$ or $(s_{j \to env})$, respectively. □

Theorem 2. *All incoming messages from the sensors which are processed by the orchestrators are redirected out of the system and applied to the actuators.*

Proof of Theorem 2. Keeping the focus on Equation (9), and following the explanation given in the previous theorem, if the broker u is not able to process the raw message (AI_B condition is false), then it is forwarded to the orchestrator v $(c_{u \to v})$. At that point, if the broker is able to process the raw message ((AI_O condition is true), then the raw message (d) is transformed into the processed message (e), meaning that such a message is not forwarded to the cloud w, which is expressed by $\emptyset$. After that, the orchestrator v sends the processed message (e) to the broker u $(c_{v \to u})$, and from that point on, it follows the explanation given in the previous theorem. □

Theorem 3. *All incoming messages from the sensors which are processed by the clouds are redirected out of the system and applied to the actuators.*

Proof of Theorem 3. Sticking to Equation (9), and following the explanation given in the last theorem, if the orchestrator v is not able to process the raw message (AI_O condition is false), then it is forwarded to the cloud w $(c_{v \to w})$. At that point, the cloud is able to process the raw message (AI_C poses no condition, as it is always able to deal with raw data), then the raw message (d) is transformed into the processed message (e). Then, the cloud w sends the processed message (e) to the orchestrator v $(c_{w \to v})$, and from that point on, it follows the explanation given in the last theorem. □

7. Conclusions

In this paper, AI and sustainability have been introduced so as to be used in the construction of a formal algebraic model for an edge monitoring system, dedicated to agricultural purposes.

To start with, a brief description has been exposed about what AI is and what their main challenges are, such as ethics and its implications. After that, a brief overview of sustainability has been exposed and their main issues have been remarked, such as how data centers may become sustainable.

Afterwards, a multipurpose field monitoring system has been proposed, making use of virtualized computing resources, which may need to be migrated among different physical hosts in order to enable them to become as close as possible to their associated users, thus lowering carbon emissions, where an AI may play its part.

Then, a formal algebraic model has been proposed for such an edge system, where two instances have been carried out, such as the processing hierarchy and the migration process, where both have been verified. Additionally, three theorems have proposed and proved regarding the formal algebraic model presented.

Eventually, this model represents an interesting design for data center organization and optimization, where nodes are spread through a given area and redundant paths are required.

Author Contributions: Conceptualization, P.J.R.; Formal analysis, P.J.R.; supervision, P.J.R., S.A., K.G., C.B. and C.J.; validation, P.J.R. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Data sharing is not applicable to this article as no new data were created or analyzed in this study.

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

Abbreviations

The following abbreviations are used in this manuscript:

ACP	Algebra of Communicating Processes
AI	Artificial Intelligence
CPS	Cyber-Physical Systems
DL	Deep Learning
DS	Data Science
FDT	Formal Description Techniques
IoT	Internet of Things
IP	Internet Protocol
IT	Information Technology
LAN	Local Area Network
MEC	Multi-Access Edge Computing
ML	Machine Learning
WAN	Wide Area Network

References

1. Frankish, K.; Ramsey, W.M. *The Cambridge Handbook of Artificial Intelligence*; Cambridge University Press: Cambridge, UK, 2014.
2. Kutyniok, G. The Mathematics of Artificial Intelligence. *arXiv* **2022**, arXiv:2203.08890v1.
3. Kazim, E.; Koshiyama, A. A High-Level Overview of AI Ethics. *Soc. Sci. Res. Netw. (SSRN)* **2020**, 3609292 , 1–18. [CrossRef]
4. Han, T.A.; Pereira, L.M.; Lenaerts, T.; Santos, F.C. Mediating artificial intelligence developments through negative and positive incentives. *PLoS ONE* **2021**, *16*, e0244592. [CrossRef] [PubMed]

5. Lotfi, I.; El Bouhadi, A. Artificial Intelligence Methods: Toward a New Decision Making Tool. *Appl. Artif. Intell.* **2021**, *36*, 1992141. [CrossRef]
6. Colosimo, B.A.; del Castillo, E.; Jones-Farmer, L.A.; Paynabar, K. Artificial intelligence and statistics for quality technology: An introduction to the special issue. *J. Qual. Technol.* **2021**, *53*, 443–453. [CrossRef]
7. Faes, L.; Liu, X.; Wagner, S.K.; Fu, D.J.; Balaskas, K.; Sim, D.A.; Bachmann, L.M.; Keane, P.A.; Denniston, A.K. A Clinician's Guide to Artificial Intelligence: How to Critically Appraise Machine Learning Studies. *Transl. Vis. Sci. Technol.* **2020**, *9*, 7. [CrossRef]
8. Lelli, F. On Exploring the Possibilities and the Limits of AI for an Interoperable and Empowering Industry 4.0. In Proceedings of the Workshop of I-ESA'22, Valencia, Spain, 23–24 March 2022; pp. 1–6.
9. Boukabara, S.A.; Krasnopolsky, V.; Penny, S.G.; Stewart, J.Q.; McGovern, A.; Hall, D.; Hoeve, J.E.T.; Hickey, J.; Huang, H.-L.A.; Williams, J.K.; et al. Outlook for Exploiting Artificial Intelligence in the Earth and Environmental Sciences. *Bull. Am. Meteorol. Soc.* **2021**, *102*, 1016–1032. [CrossRef]
10. Lee, S.; Yu, R.; Xie, J.; Billah, S.M.; Carrol, J.M. Opportunities for Human-AI Collaboration in Remote Sighted Assistance. In Proceedings of the 27th International Conference on Intelligent User Interfaces (IUI'22), Helsinki, Finland, 22–25 March 2022; pp. 63–78.
11. Pai, R.Y.; Shetty, A.; Shetty, A.D.; Bhandary, R.; Shetty, J.; Nayak, S.; Dinesh, T.K.; D'souza, K.J. Integrating artificial intelligence for knowledge management systems–synergy among people and technology: A systematic review of the evidence. *Econ. Res. Ekon. Istraz.* **2022**, *35*, 7043–7065. [CrossRef]
12. Niemi, H. AI in learning: Preparing grounds for future learning. *J. Pac. Rim Psychol.* **2021**, *15*, 1–12. [CrossRef]
13. Gajos, K.Z.; Mamykina, L. Do People Engage Cognitively with AI? Impact of AI Assistance on Incidental Learning. In Proceedings of the 27th International Conference on Intelligent User Interfaces (IUI'22), Helsinki, Finland, 22–25 March 2022; pp. 794–806.
14. Janiesch, C.; Zschech, P.; Heinrich, K. Machine learning and deep learning. *Electron Mark.* **2021**, *31*, 685–695. [CrossRef]
15. Halina, M. Insightful artificial intelligence. *Mind Lang.* **2021**, *36*, 315–329. [CrossRef]
16. Chauhan, S.; Vig, L.; De Grazia, M.F.; Corbetta, M.; Ahmad, S.; Zorzi, M. A Comparison of Shallow and Deep Learning Methods for Predicting Cognitive Performance of Stroke Patients From MRI Lesion Images. *Front. Neuroinform.* **2019**, *13*, 53. [CrossRef]
17. Coiera, E. The Last Mile: Where Artificial Intelligence Meets Reality. *J. Med. Internet Res.* **2019**, *21*, e16323. [CrossRef] [PubMed]
18. Sevilla, J.; Heim, L.; Ho, A.; Besiroglu, T.; Hobbhahn, M.; Villalobo, P. Compute Trends Across Three Eras of Machine Learning. In Proceedings of the International Joint Conference on Neural Networks (IJCNN), Padua, Italy, 18–21 July 2022; pp. 1–8.
19. Smeliansky, R.L. Model of Distributed Computing System Operation with Time. *Program. Comput. Softw.* **2013**, *39*, 233–241. [CrossRef]
20. Nenashev, A.V.; Tolsteko, A.Y.; Oleshko, R.S. Model of the peer-to-peer distributed system for securable information storage and processing without traffic prioritization (The OoL project). In Proceedings of the III International Workshop on Modeling, Information Processing and Computing (MIP: Computing-2021), Krasnoyarsk, Russia, 28 May 2021; pp. 141–150.
21. Delporte-Gallet, C.; Fauconnier, H.; Fraigniaud, P.; Rabie, M. Distributed Computing in the Asynchronous LOCAL model. *arXiv* **2019**, arXiv:1904.07664.
22. Miller, H.; Haller, P.; Müller, N.; Boullier, J. Function passing: A model for typed, distributed functional programming. In Proceedings of the ACM International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software (Onward! 2016), Amsterdam, The Netherlands, 2–4 November 2016; pp. 82–97.
23. Jiang, Q.; Yan, S.; Cheng, H.; Yan, X. Local–Global Modeling and Distributed Computing Framework for Nonlinear Plant-Wide Process Monitoring With Industrial Big Data. *IEEE Trans. Neural Netw. Learn. Syst.* **2021**, *32*, 3355–3365. [CrossRef] [PubMed]
24. Gyongyosi, L.; Imre, S. Scalable distributed gate-model quantum computers. *Sci. Rep.* **2021**, *11*, 5172. [CrossRef]
25. Klein, C.; Rumpe, B.; Broy, M. A stream-based mathematical model for distributed information processing systems. *arXiv* **2014**, arXiv:1409.7236.
26. Saha, S.; Guha, T.; Bhattacharya, S.S.; Banik, M. Distributed Computing Model: Classical vs. Quantum vs. Post-Quantum. *arXiv* **2020**, arXiv:2012.05781.
27. Margara, A.; Cugola, G.; Felicioni, N.; Cilloni, S. A Model and Survey of Distributed Data-Intensive Systems. *arXiv* **2020**, arXiv:2203.10836.
28. Martínez-Alba, A.; Babarczi, P.; Blenk, A.; He, M.; Kalmbach, P.; Zerwas, J.; Kellerer, W. Modeling the Cost of Flexibility in Communication Networks. In Proceedings of the IEEE Conference on Computer Communications (IEEE INFOCOM 2021), Virtual Conference, 10–13 May 2021; pp. 1–10.
29. Irzaev, G. A logical model of the distributed corporate network of an insurance company. *J. Phys. Conf. Ser.* **2020**, *1515*, 042088. [CrossRef]
30. Doyle, A.; Roy, R. Communication Network Models. In *Network Models and Optimization. Decision Engineering*; Springer: London, UK, 2008; pp. 229–295.
31. Zhu, X.; Li, J.; Liu, Z.; Yang, F. Location deployment of depots and resource relocation for connected car-sharing systems through mobile edge computing. *Int. J. Distrib. Sens. Netw.* **2017**, *13*. [CrossRef]
32. Fan, W.; Liu. J.; Hua, M.; Wu, F.; Liu, Y. Joint Task Offloading and Resource Allocation for Multi-Access Edge Computing Assisted by Parked and Moving Vehicles. *IEEE Trans. Veh. Technol.* **2022**, *71*, 5314–5330. [CrossRef]
33. Matthé, M. Applying reconfiguration cost and control pattern modeling to self-adaptive systems. In Proceedings of the ACM/IEEE 44th International Conference on Software Engineering (ICSE'22), Pittsburgh, PA, USA, 22–27 May 2022; pp. 248–250.

34. Krupitzer, C.; Prantl, T.; Raibulet, C.Adaptive Systems in the Context of the Internet of Things. *IEEE Access* **2020**, *8*, 187384–187399. [CrossRef]
35. Wang, Y.; Xia, Y.; Zhang, Y.; Melissourgos, D.; Odegbile, O.; Chen, S. A Full Mirror Computation Model for Edge-Cloud Computing. In Proceedings of the 13th International Conference on Contemporary Computing (IC3-2021), Noida, India, 5–7 August 2021; pp. 132–139.
36. Amannejad, Y. Building and Evaluating Federated Models for Edge Computing. In Proceedings of the 16th International Conference on Network and Service Management (CNSM), Izmir, Turkey, 2–6 November 2020.
37. Latif, R.; Ahmed, M.U.; Tahir, S.; Latif, S.; Igbal, W.; Ahmad, A. A novel trust management model for edge computing. *Complex Intell. Syst.* **2022**, *8*, 3747–3763. [CrossRef]
38. Toczé, K.; Lindqvist, J.; Nadjm-Tehrani, S. Characterization and modeling of an edge computing mixed reality workload. *J. Cloud Comput.* **2020**, *9*, 46. [CrossRef]
39. Pandian, A.P. Enhanced edge model for big data in the Internet of Things based applications. *J. Trends Comput. Sci. Smart Technol.* **2019**, *1*, 57–67.
40. Cavalieri d'Oro, E.; Colombo, S.; Gribaudo, M.; Iacono, M.; Manca, D.; Piazzolla, P. Modeling and evaluating a complex edge computing based systems: An emergency management support system case study. *Internet Things* **2019**, *6*, 100054. [CrossRef]
41. Allahham, M.S.; Mohamed, A.; Erbad, A.; Hassanein, H. On the Modeling of Reliability in Extreme Edge Computing Systems. *arXiv* **2022**, arXiv:2208.05817.
42. Choi, J.; Lee, J.; Ryu, D.; Kim. S.; Baik, J. GAIN-QoS: A Novel QoS Prediction Model for Edge Computing. *J. Web Eng.* **2022**, *21*, 27–52. [CrossRef]
43. Li, Y.; Lu, Y.; Cui, H.; Velipasalar, S. Improving robustness and efficiency of edge computing models. *Wirel. Netw.* **2023**, *29*, 27–52. [CrossRef]
44. Berger, M.; Bernardello, F.; Barry, C.; Badjoonauth, P.; Balaji, S.; Lakhdar, M. Real-time Model Predictive Control with Digital Twins and Edge Computing Technologies. In Proceedings of the 14th REHVA HVAC World Congress, Rotterdam, The Netherlands, 22–25 May 2022.
45. Jiang, L.; Chang, X.; Yang, R.; Misic, J.; Misic, V.B. Model-Based Comparison of Cloud-Edge Computing Resource Allocation Policies. *Comput. J.* **2020**, *63*, 1564–1583. [CrossRef]
46. Sun, Y.; Li, Y.; Xuehong, C.; Li, J. Optimal defense strategy model based on differential game in edge computing. *J. Intell. Fuzzy Syst.* **2020**, *39*, 1449–1459. [CrossRef]
47. Jian, C.; Chen, J.; Ping, J.; Zhang, M. An Improved Chaotic Bat Swarm Scheduling Learning Model on Edge Computing. *IEEE Access* **2019**, *7*, 58602–58610. [CrossRef]
48. Aleksandrova, E.; Anagnostopoulos, C.; Kolomvatsos, K. Machine Learning Model Updates in Edge Computing: An Optimal Stopping Theory Approach. In Proceedings of the 18th International Symposium on Parallel and Distributed Computing (ISPDC), Amsterdam, The Netherlands, 5–7 June 2019.
49. Song, Z.; Tilevich, E. A Programming Model for Reliable and Efficient Edge-Based Execution under Resource Variability. In Proceedings of the IEEE International Conference on Edge Computing (EDGE), Milan, Italy, 8–13 July 2019; pp. 64–71.
50. Song, H.; Dautov, R.; Ferry, N.; Solberg, A.; Fleurey, F. Model-based fleet deployment of edge computing applications. In Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems (MODELS 2020), New York, NY, USA, 16–23 October 2020; pp. 132–142.
51. Tawalbeh, L.; Muheidat, F.; Tawalbeh, M.; Quwaider, M.; Abd El-Latif, A.A. Edge enabled IoT system model for secure healthcare. *Measurement* **2022**, *191*, 110792. [CrossRef]
52. Pereira, P.; Araujo, J.; Melo, C.; Santos, V.; Maciel, P. Analytical models for availability evaluation of edge and fog computing nodes. *J. Supercomput.* **2021**, *77*, 9905–9933. [CrossRef]
53. Wang, J.; Wu, W.; Liao, Z.; Sangaiah, A.K.; Sherratt, R.S. An Energy-Efficient Off-Loading Scheme for Low Latency in Collaborative Edge Computing. *IEEE Access* **2019**, *7*, 149182–149190. [CrossRef]
54. Gadasin, D.V.; Shvedov, A.V.; Koltsova, A.V. Cluster Model for Edge Computing. In Proceedings of the International Conference on Engineering Management of Communication and Technology (EMCTECH), Vienna, Austria, 20–22 October 2020.
55. Jia, C.; Lin, K.; Deng, J. A Multi-property Method to Evaluate Trust of Edge Computing Based on Data Driven Capsule Network. In Proceedings of the IEEE INFOCOM 2020 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), Toronto, ON, Canada, 6–9 July 2020.
56. Sasaki, Y.; Sato, T.; Chishiro, H.; Ishigooka, T.; Otsuka, S.; Yoshimura, K.; Kato, S. An Edge-Cloud Computing Model for Autonomous Vehicles. In Proceedings of the 11th Workshop on Planning, Perception and Navigation for Intelligent Vehicles, Macau, China, 4 November 2019.
57. Ibn-Khedher, H.; Laroui, M.; Ben Mabrouk, M.; Moungla, H.; Afifi, H.; Oleari, A.N.; Kamal, A.E. Edge Computing Assisted Autonomous Driving Using Artificial Intelligence. In Proceedings of the International Wireless Communications and Mobile Computing (IWCMC), Harbin, China, 28 June–2 July 2021.
58. Valocky, F.; Orgon, M.; Fujdiak, I. Experimental Autonomous Car Model with safety sensor in Wireless Network. In Proceedings of the 16th IFAC Conference on Programmable Devices and Embedded Systems (PDES), High Tatras, Slovakia, 29–31 October 2019.
59. Wei, Y.; Zhang, J. A Vehicular Edge Computing-Based Architecture and Task Scheduling Scheme for Cooperative Perception in Autonomous Driving. *Mathematics* **2020**, *10*, 3328. [CrossRef]

60. Jaggard, A.D.; Lutz, N.; Schapira, M.; Wright, R.N. Dynamics at the Boundary of Game Theory and Distributed Computing. *ACM Trans. Econ. Comput.* **2017**, *5*, 1–20. [CrossRef]
61. Baranawal, A.; Simmhan, Y. Optimizing the interval-centric distributed computing model for temporal graph algorithms. In Proceedings of the Seventeenth European Conference on Computer Systems (EuroSys'22), Rennes, France, 5–8 April 2022; pp. 541–558.
62. Bagchi, S. Computational modeling of consistent observation of asynchronous distributed computation on N-manifold. *Cogent Eng.* **2018**, *5*, 1528029. [CrossRef]
63. Makhortov, S.D.; Bolotova, S.Y. An algebraic model of the production type distributed intelligent system. *J. Phys. Conf. Ser.* **2019**, *1203*, 012045. [CrossRef]
64. Kakkavas, G.; Gkatzioura, D.; Karyotis, V.; Papavassiliou, S. A Review of Advanced Algebraic Approaches Enabling Network Tomography for Future Network Infrastructures. *Future Internet* **2020**, *12*, 20. [CrossRef]
65. Fittipaldi, P.; Giovanidis, A.; Grosshans, F. A Linear Algebraic Framework for Quantum Internet Dynamic Scheduling. In Proceeding of the IEEE International Conference on Quantum Computing and Engineering (QCE), Broomfield, CO, USA, 18–23 September 2022; pp. 447–453.
66. Duarte, C.H.C. Mathematical Models of Object-Based Distributed Systems. *Lect. Notes Comput. Sci.* **2011**, *7000*, 57–73.
67. Letychevskyi, O.; Peschanenko, V.; Radchenko, V.; Orlovsky, M.; Sobol, A. Algebraic Approach to Verification and Testing of Distributed Applications. In Proceeding of the 1st International Electronics Communication Conference (IECC 2019), Okinawa, Japan, 7–9 July 2019.
68. Yuan, B.; Jankov, D.; Zou, J.; Tang, Y.; Bourgeois, D.; Jermine. C. Tensor Relational Algebra for Distributed Machine Learning System Design. *Proc. VLDB Endow.* **2021**, *14*, 1338–1350. [CrossRef]
69. Gaur, M.; Kant, R. A Survey on Process Algebraic Stochastic Modelling of Large Distributed Systems for Its Performance Analysis. In Proceedings of the 3rd International Conference on Eco-friendly Computing and Communication Systems, Mangalore, India, 18–21 December 2014; pp. 206–211.
70. Roig, P.J.; Alcaraz, S.; Gilly, K.; Bernad, C.; Juiz, C. Modeling an Edge Computing Arithmetic Framework for IoT Environments. *Sensors* **2022**, *22*, 1084. [CrossRef] [PubMed]
71. Seoane, V.; García-Rubio, C.; Almenares, F.; Campo, C. Performance evaluation of CoAP and MQTT with security support for IoT environments. *Comput. Netw.* **2021**, *197*, 108338. [CrossRef]
72. Fokkink, W. *Modelling Distributed Systems*, 2nd ed.; Springer: Berlin/Heidelberg, Germany, 2017.
73. Roig, P.J.; Alcaraz, S.; Gilly, K.; Bernad, C.; Juiz, C. Modeling of a Generic Edge Computing Application Design. *Sensors* **2021**, *21*, 7276. [CrossRef]
74. Groote, J.F.; Mousavi, M.R. *Modeling and Analysis of Communicating Systems*, 1st ed.; MIT Press: Cambridge, MA, USA, 2014.
75. Bergstra, J.A.; Middleburg, C.A. Using Hoare Logic in a Process Algebra Setting. *Fundam. Informaticae* **2021**, *179*, 321–344. [CrossRef]
76. Fokkink, W. *Introduction to Process Algebra*, 2nd ed.; Springer: Berlin/Heidelberg, Germany, 2007.