applied sciences

MDPI

Article

Modeling Dew Computing in DISSECT-CF-Fog

Andras Markus [1,*], Mate Biro [1], Karolj Skala [2], Zorislav Šojat [2] and Attila Kertesz [1]

[1] Department of Software Engineering, University of Szeged, 6720 Szeged, Hungary
[2] Centre for Informatics and Computing, Institut Ruđer Bošković, 10000 Zagreb, Croatia
* Correspondence: markusa@inf.u-szeged.hu

Abstract: Fog computing provides an effective solution to various problems by extending the cloud's functionality to typically more limited computing units closer to user devices. Fog computing can provide a higher level of user experience due to its geographic and network topology location and distribution. IoT services also need to be managed seamlessly to ensure adequate QoS (due to the mobility of devices or the temporary periods without an internet connection). Such domains are combined under the auspices of Dew computing, as in critical cases, extending an IoT service to the end user's device is a feasible task. Such scenarios can hardly be investigated at a large scale due to the lack of dedicated simulation environments. In this paper, we present an extension of the DISSECT-CF-Fog simulator with a Dew computing model, to enable the simulation of IoT-Dew-Fog systems in a cost-effective manner. In particular, we focus on service migration options for mobile devices and cases with temporary internet access limitations. Finally, we performed measurements of real-world use cases with the extended simulator as an evaluation. Our simulation results show that the proposed proactive strategy reduces the processing time of IoT data, exploiting an IoT-Dew-Fog environment.

Keywords: Dew computing; fog computing; Internet of Things; simulation

Citation: Markus, A.; Biro, M.; Skala, K.; Šojat, Z.; Kertesz, A. Modeling Dew Computing in DISSECT-CF-Fog. *Appl. Sci.* **2022**, *12*, 8809. https://doi.org/10.3390/app12178809

Academic Editor: Miguel García-Pineda

Received: 29 July 2022
Accepted: 27 August 2022
Published: 1 September 2022

1. Introduction

The number of smart devices connected to the internet has grown exponentially in the last decade, driven by rapid technological advances and ever-increasing user demand. According to Cisco's annual internet report (visited on 15 May 2022): https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html), by 2023, the number of mobile devices will reach 13.1 billion globally, of which, 1.4 billion devices will have 5G capabilities. The large amounts of data generated by these devices significantly burden traditional cloud computing-based networks; therefore, various distributed and decentralized network paradigms have gained ground. Fog computing complements previous cloud technology by bringing services closer to the users.

Compared to the traditional cloud model, the fog model is more supportive of mobile data processing, as it is more efficient in maintaining minimal latency and predictable response times. Services that process short life-cycle data streams are typically deployed to computing units located at the edge of the network. In the fog model [1], the units are limited in terms of resources, but any device may act as a fog node, which is able to communicate over the network and has storage and computing capacity. This distributed model also provides the possibility to use virtualized units.

Distributed models are intensively researched due to the emergence of the Internet of Things (IoT) and 5G technology. Cloud computing and its complementary models have major roles in the development of IoT applications, especially in storing data and processing related tasks with low latency requirements or deadline constraints.

Although the fog model resolves the problems induced by the volume and diversity of IoT data, mobile data processing is not a trivial task. Data center coverage is limited; due to the long distances covered by the movements of mobile devices, the data to be

processed potentially need several network hops to reach their destinations. Furthermore, increased latencies and unpredictable response times degrade the quality of service (QoS). One possible way to reduce the negative impact, as much as possible, is to dynamically migrate the IoT services to a more optimal (preferably closer) location. Movement-induced service migration is a challenging problem that needs to be addressed from many different angles. For example, the distance between the device and the node (both physical and network distance), the size of the service to be migrated, the available bandwidth, or the current load on the destination node.

Cloud-influenced domains, such as fog, edge, and mobile edge computing [2], are often utilized to outsource cloud computing features and services to the perimeters of mobile networks. Such complex networks strongly rely on (typically the internet) connections; therefore, another novel paradigm was introduced, namely Dew computing, to deal with the lack of internet access. Dew computing compounds the basic capabilities of end devices and the cloud model. In this area, IoT data can be discovered, processed, and stored locally in the case of offline modes. When a network connection is established, the end devices may initiate synchronization with the fog and cloud services. With the Dew concept, various challenges arise, such as resource and network utilization, limited storage, and energy consumption [3].

The concept of Dew computing can be applied to various domains [4], such as air transportation systems, including flying objects (meteorological drones, airplanes, etc.), healthcare systems aimed at real-time monitoring of patients and interactions with doctors, smart manufacturing focusing on Industry 4.0, and traffic control relying on distributed surveillance systems.

The main contributions of the paper are as follows:

- We present an overview of Dew computing and extend the DISSECT-CF-Fog simulator with a Dew model to represent IoT-Dew-Fog systems.
- We introduce realistic, Dew-based IoT use cases exploiting service migration and offline data management.
- We evaluate the use cases in the extended simulator, pinpointing some technical challenges that Dew computing can bring to these complex systems.

The remainder of the paper is as follows. In Section 2, we briefly discuss the Dew computing domain and related simulation approaches, in Section 3, we introduce the simulator and its components, in Section 4, we present two IoT-Dew-Fog use cases for exemplifying its utilization. Finally, in Section 5, we conclude our work.

2. Related Works

2.1. Dew Computing

The relatively recent expansion of the use of processing units in a vast majority of technical products, including home and office control and appliances, industrial equipment, traffic control (earth, sea, and sky), vehicles, lighting, personal data collecting/processing wearables (e.g. health and sports-oriented) etc., including mobile computing, storage, sensing, and communication integrated personal devices (so-called "smartphones") and home computers have led to the necessity to expand the paradigmatic structure of cloud and fog computing with a low-level paradigmatic layer, the Dew computing layer [5–7], enabling seamless inclusion of the mentioned (very heterogeneous) types of processing units into the Rainbow global service ecosystem) [8,9].

The major distinction between the equipment in the cloud/fog/edge areas of the hierarchy and the Dew computing area is that the equipment in the cloud/fog/edge areas are primarily dedicated to general tasks involving computations and communications, and are integral parts of the internet. The Internet of Things (IoT) is a natural extension of the internet and presupposes that "Things" will communicate through (and be a part of) the internet. However, regarding all of the equipment that we mentioned, they have one important thing in common: they have to be self-standing, self-sufficient systems, i.e., they must perform their functions completely independent of any connections with

other equipment or systems. Hereby, we define the self-sufficient system as an independent system of components that can perform its intended functions without any external communication/processing needs (for example a car, a washing machine, a traffic lights system, etc.). All of that equipment is outside of the edge of the internet, including mobile devices and personal computers. However, it can be very beneficial if such equipment could be 'coordinated', and cooperate with cloud and/or fog services whenever an internet connection is established. However, it must be noted that non-internet connectivity is also a major aspect at present and there will be an increasing amount of future equipment (LAN and ad hoc radio/wire/optical connections).

Therefore, at the layer of Dew computing, it is possible to solve the large diversity of communication and information/services. In this paper, we exemplify how to model such tasks in a state-of-the-art simulator called DISSECT-CF-Fog. In the next subsection, we provide an overview of Dew modeling possibilities in the simulation field.

2.2. Simulation Approaches

Investigating and maintaining IoT-Fog-Cloud systems in the real-world could be extremely expensive due to the financial expenses of IoT devices and cloud services. Furthermore, examining various scheduling and offloading algorithms may be time- and energy-consuming. Simulation approaches have become acceptable solutions for such purposes among researchers because they mimic existing systems in realistic manners. They ensure cost-efficient and scalable environments to test and validate new procedures and algorithms, in which results can be used for further modification of the real system.

One of the most well-known simulators dedicated to modeling the cooperation of IoT and fog computing is called iFogSim [10], which extends the functionality of CloudSim toward fog computing. With the latest version, device mobility can be simulated, which attracts the need for application service migration as well, because the positions of the end devices change frequently. This can cause increased response times and the required QoS cannot be guaranteed without migrating services to a more suitable provider. Furthermore, fog node clustering and microservice orchestration of application services are also part of the upgraded system. The literature may refer to this extension as iFogSim2.

MobFogSim [11] is derived from the original version of iFogSim, supporting user (i.e., device) migration to minimize access delay by triggering the handover of user devices between computing nodes. Mobility may require migration solutions for VMs and containers to mitigate increasing latency and improve QoS. IoTSim-Edge [12], similar to iFogSim, is built upon CloudSim, and it extends its capabilities with IoT-related behavior, including IoT data generation, battery drainage simulation, local and remote IoT data processing, as well as device mobility. EdgeCloudSim [13] relies on CloudSim as well; therefore, it inherited the functionalities of the core simulator. However, it has a new CPU utilization model, device mobility, and edge orchestrator.

To the best of our knowledge, DewSim [14] is currently the only simulator that models a Dew computing environment directly. It mainly focuses on simulating mobile device clusters, of which, members are considered the primary computing resource providers. Trace-based battery simulation is also available in the simulator in order to model battery drainage realistically. With different job allocation strategies, jobs are only distributed among battery-dependent devices; moreover, non-battery-dependent devices and detailed physical infrastructure management are not considered at all. P. Sanabria et al. [15] introduced an extended version of DewSim, which supports simulating hybrid Dew/edge environments, including non-battery-dependent devices; however, the management of the computing infrastructure is still missing.

DISSECT-CF-Fog [16] deals with a detailed IaaS model, including physical and virtual machines, storage, and data center network properties, among others. In general, its components are split into two main parts, physical and virtual layers. The physical parameters describe the physical capabilities of the fog and cloud nodes. The fog extension also provides a comprehensive IoT layer representation, where IoT physical layer components

(sensors and actuators) can be represented, and smart devices with different properties can be modeled. In the virtual layer, IoT applications running on the computing node are responsible for processing data. This layer also considers the energy consumption of the entities, IoT device mobility, and pricing schemes of real providers.

We present the simulators in Table 1, where we denote the domains (IoT, edge/Dew computing and fog/cloud computing) and the functionalities available in the tools. According to the preliminary analysis and previous studies [16,17] (which showed that DISSECT-CF-Fog is more scalable, reliable, and faster than iFogSim), in this paper, we chose DISSECT-CF-Fog to extend towards IoT-Dew-Fog systems by modeling Dew computing.

Table 1. Summary of the related simulation approaches (√ means the simulator contains that functionality).

Simulators	Internet of Things			Edge/Dew Computing	Fog/Cloud Computing	
-	IoT Devices	Battery/Energy Consumption	Mobility	-	IaaS	Migration
iFogSim	√	√	√	X	√	√
MobFogSim	√	√	√	X	√	√
IoTSim-Edge	√	√	√	√	√	X
EdgeCloudSim	√	√	√	√	√	X
DewSim	√	√	X	√	X	X
DISSECT-CF-Fog	√	√	√	√	√	√

3. The Proposed Model for Dew Computing

High distances between the IoT devices and fog/cloud nodes can cause critical increases in service response times. Furthermore, the lack of internet access also requires effective solutions to provide continuous service. If a smart device has sufficient computing and storage capacity, the service can be migrated to the associated smart device. In most cases, the capacities of end user devices, which can also be local PCs or tablets, are more limited than the fog nodes serving them; therefore, computational tasks may take more time to complete. The requested service should only be installed on a smart device if the increased response time due to distance is expected to be less than the response time of the application running on the device. Such migration and further CPU-related tasks can affect the battery life of the device and may increase its energy consumption dramatically, or slow down the performance of other tasks.

DISSECT-CF-Fog has a generic IoT device representation, which transmits the information collected by the sensors to the processing units. To model Dew-related use cases realistically, this representation was extended to provide various behaviors, including service migration to mobile devices and temporary internet access limitation. To be as realistic as possible, the simulator also supports several types of device mobilities: (i) *random* motion moves at random speeds between the minimum and maximum values in a random direction in a circle of radius r for a period t, and then at the $t + 1$ moment, the speed and direction may change. When the device reaches the edge of its moving environment, the device is turned back according to the current degree of motion. (ii) *Deterministic* motion is capable of moving at a given speed based on predefined geographic positions $l_1, l_2, \ldots, l_n$ in the specified order. When the device reaches position l_i, the next destination, if any, is l_{i+1}. If there are no more positions, the device stops. Finally, (iii) *GPS* movement deals with real GPS data. The coordinates are read from the files in a corresponding timestamp. Besides these movements, devices can be placed in static positions; in this case, the initial geographic coordinates of the device did not change during the simulation.

3.1. Service Migration

Moving IoT devices may switch service providers at any time in order to ensure the best QoS, which is called mobility-induced service migration. The main issue is how to seamlessly redirect users to other computing notes without interrupting the service,

especially in the case of time-sensitive applications. Service migration can be distinguished based on timing [18]. In case of proactive migration (i), the migration of the requested service must be done before the device starts using it. With a proactive strategy, near-continuous service is provided; however, it requires a preliminary decision. In case of an incorrect decision, unnecessary migration can be costly and degrade QoS. With the reactive strategy (ii), unnecessary migration can be avoided; however, partial outage of the service is likely because the migration process is induced after the handover of the IoT devices. Whichever strategy is used, the network properties, the physical distance, and the size of the service influence the length of the process.

We focused on proactive migration mechanisms because service interruption is unacceptable for today's real-time applications. As a result, the service is already ready by the time the user's device is received by the new node. In order to make such decisions, the strategy has to decide when and where the service should be migrated. Typically it is initialized when the response time or the latency increase due to the distance between the node and the IoT device. To decide which other node (i.e., where) the service should be migrated, the actual load, network delay, and physical position can be considered.

In this paper, a Markov chain was used for proactive migration prediction, because it is a popular method for mobility estimation due to its efficiency and intuitiveness [19]. The Markov model looks at a set of data and attempts to establish rules between the different directions of movement. The next estimate is based only on the measurements that preceded it. Hence, the k-order Markov model examines the $k-1$ of data preceding the estimate.

The Markov chain defines a so-called Markov space, which can be written as a vector of finite size. Let MS denote the Markov space, and $MS_t = x_t$ denote the probability that the model is in the x_t state at time t ($t = 0, 1, \dots$), formally:

$$\mathcal{P}(MS_{t+1} = x | MS_1 = x_1, \dots, MS_t = x_t = \mathcal{P}(MS_{t+1} = x | MS_t = x_t) \quad (1)$$

The first step in a predictive procedure based on the Markov model is to define a state transition matrix (denoted by P), where each element represents the probability of the transition between two possible states, formally:

$$P_{i,j} = \mathcal{P}(MS_{t+1} = x_j | MS_t = x_i) \quad (2)$$

Of course, the number of transitions can be increased if we want to take into account earlier states in time, formally:

$$P_{i,j}^{(k)} = \mathcal{P}(MS_k = x_j | MS_0 = x_i) \quad (3)$$

The transition matrix stores the probability between two directions measured in degrees. This means that the size of the matrix is $N \times N$, where $N = 360$. For example, the 10th row and 250th column of the matrix give the probability that currently the device is moving at an angle of 10 degrees, and in the next step, it will turn to the angle of 250 degrees.

In this model, we deal with the following assumption: *The transition probability between two closer angles is proportionally higher than further angles*. For instance, the probability that the device will turn from 90 degrees to 92 degrees is much higher than turning from 90 degrees to 180 degrees. Furthermore, the chances of turning from 90 degrees to 0 degrees are the same as turning from 90 degrees to 180 degrees.

As we initially had no information about the moving behavior of the devices, we used the assumption mentioned earlier to initialize the matrix P, as follows:

1. With an appropriate function, which strictly monotonically increases, 0 to 179 degrees are determined. In this paper, the following function is used, but it can be easily changed in the simulator:
$$\left(\frac{e}{\pi}\right)^{-0.2x} \tag{4}$$
2. From 180 to 359 degrees, the previously defined values are mirrored and added to the original vector in reverse order. The resulting values are shown in Figure 1.
3. The sum of the vector elements must be 1; therefore, each element is normalized in the vector.
4. Each row of the matrix is obtained by shifting the elements of the vector constructed before by the corresponding row of the matrix so that the highest values lie on the diagonal of the matrix. The left neighbor of element 0 is element 360, and the right neighbor of element 360 is element 0.

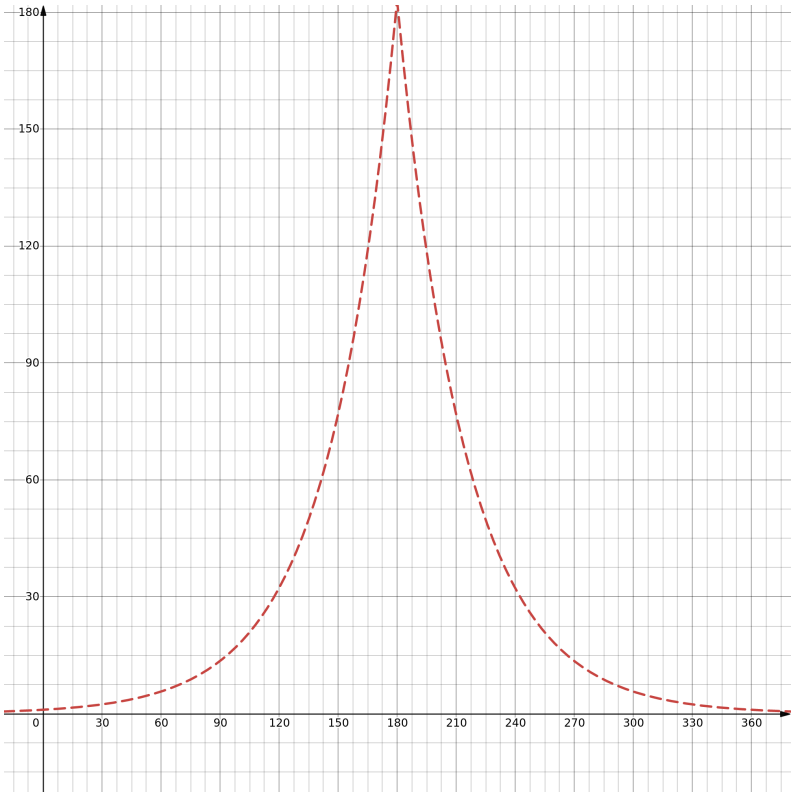

Figure 1. Vector values of 360 degrees.

As the simulator runs, we update the transition matrix of a device based on its movement. The element of the matrix (i, j) is incremented by a constant $(\frac{0.025}{\sum_{j=1}^{360} P_{i,j}})$ if there is a transition from i to j.

A weighted Markov model takes into account the partial results of previous steps with different weights. It can be assumed that older movements have less effect on the outcomes of the next move than recent movements. Thus, the weight coefficients decrease from 1 to k for a k-order model. We define the weights as follows [20]:

$$\omega_i = \frac{k-i+1}{\sum_{m=1}^{k} m}, i = 1, 2, \ldots, k. \tag{5}$$

As the last step to determine the estimated direction, we calculate the probability of each possible direction (360). Assume that the current direction is dir_1, the previous k directions are $dir_2, dir_3, \ldots, dir_k$, and the possible direction is dir_n:

$$dir_1 \rightarrow dir_n := \sum_{j=1}^{k} \omega_j P^{(k)}_{dir_j, dir_n} \tag{6}$$

The resulting values are compared, and the maximum value is chosen to obtain the estimated value.

3.2. Limited Internet Access

The two main features of Dew computing are (i) *independence* when the functionality of the cloud service is still available even with a limited internet connection, and (ii) *collaboration*, when the information sampled and collected is exchanged automatically with the cloud service. It can happen continuously or when internet access recovers. Dew computing extends the cloud functionalities to the end users' gadgets, such as laptops, smartphones, and so on, which is often called a Dew server.

The different positions of the local Dew servers are depicted in Figure 2. Since distance can have a critical impact on service response times, in this work, the movements of the devices are also considered ($device_a$, $device_b$, $device_c$). The increasing latency of the computing nodes ($node_i$, $node_j$) is depicted by the ranges ($range(node_i)$, $range(node_j)$), the closer a device is to the boundary of the range, the higher the latency it deals with. These limitations are considered as follows: $device_a$ is located inside two ranges; therefore, it can decide which node will be advantageously utilized by the device. Various properties can be involved in such decisions (utilization cost, latency, workload). Moreover, $device_b$ device is limited to communicating with the center node of the range. Finally, $device_c$ is too far from any center nodes, which means it is considered a Dew server in a non-internet zone.

However, if it is reached in the future by its movement (any range covered by at least one node), the device would be able to begin communicating with a more beneficial fog node. Typically, the nodes with fewer computing tasks and shorter response times are preferred by the devices. On the contrary, if any device leaves a range (i.e., the response time/latency becomes intolerable), it will act as a Dew server.

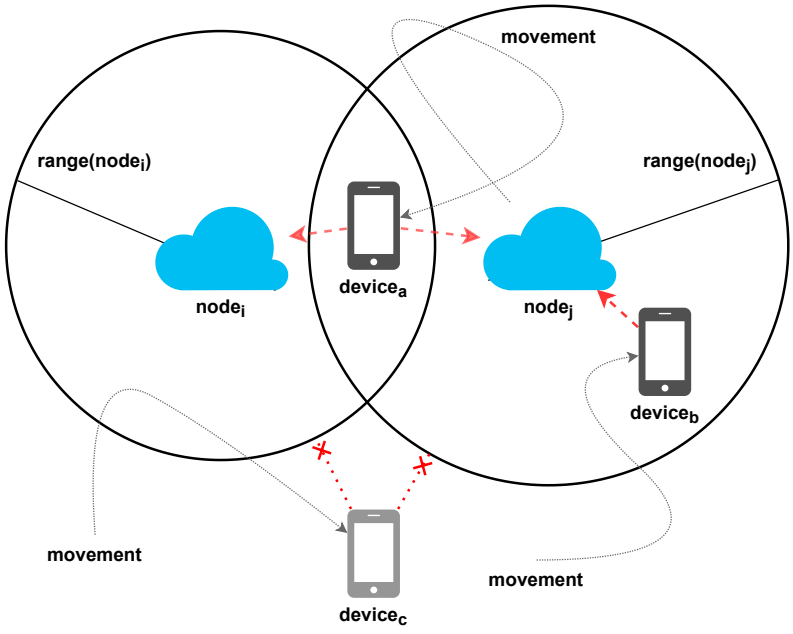

Figure 2. Limited internet access to smart devices.

For Dew computing, only smart devices are considered that have sufficient networking, storing, and processing abilities; thus, the service can be migrated to the associated smart device. However, this should be handled with care, as it can greatly affect the battery life of the device or slow down other tasks. In most cases, the capacity of the end users' devices is more limited than fog and cloud nodes, so computational tasks may take more time to complete.

Nevertheless, today's smart devices, such as laptops, tablets, personal computers, and phones, fulfil the requirements of Dew computing to serve as Dew servers.

To model a smart/Dew device as similar as possible to a fog/cloud node, virtual layers should be provided. As most Dew devices have general purpose memories and CPUs, this is an easy task. In the virtualized Dew device architecture, we assume that there is only one VM at a time, which is vertically scalable, (i.e., more resources can be allocated to a given VM if needed). On the other hand, a standard fog server is also horizontally scalable (i.e., it can run multiple VMs at the same time).

As a result, the device behaving as a Dew server (running a Dew service) will be different form a full-fledged fog node, as it will not be able to serve other devices, but will only respond to data from its own domain, which is processed and evaluated locally.

4. Evaluation

4.1. Weather Forecasting Scenario

In this subsection, we present an IoT-assisted weather forecasting scenario utilizing both fog nodes and Dew services running on Dew devices. As can be sen in Figure 3, in the central European area, drones fly and collect information about temperature, humidity, and wind in the atmosphere (Droneblog.com (visited on 15 June 2022): https://www.droneblog.com/how-drones-are-helping-with-weather-forecasting/), the size of the whole area is around 850×230 km^2. At the beginning, the devices are located at the edge of the map (uniformly distributed) and are heading to opposite sides. Such devices are equipped with computing and storing units (8 CPU cores, 8 GB RAMs, 16 SD-card), and fly at an average speed of 70 km/h.

In the case of smart device data transmission, the communication delays are calculated based on average 4G latency (50 ms) weighted with the physical distance to the node utilized by the device.

To be as realistic as possible, we applied the cloud schema of the ELKH cloud of ELKH SZTAKI (ELKH cloud of ELKH SZTAKI (visited on 20 May 2022): https://science-cloud.hu/en) to determine the CPU processing power, network operations, and storage capacities of physical machines for the experiments. Figure 3 depicts five fog nodes located in Lille, Cologne, Frankfurt, Nuremberg, and Prague, each of them dealing with 32 CPU cores and 64 GB memory. The simulator can also calculate resource utilization costs, so we set VM prices according to the pricing scheme of Amazon Web Services (Amazon Web Service (visited on 15 June 2022): https://aws.amazon.com/ec2/pricing/on-demand/). Each VM that runs on fog nodes has 4 CPU cores and 8 GB memory, which costs USD 0.116 hourly (*"a1.xlarge"*).

To evaluate the proposed proactive service migration strategy, the following use cases are defined: (i) without motion prediction and (ii) with motion prediction using the two-order Markov model. In the first case, this means that service migration only happens when the current position of a device is not covered by any node (i.e., reactive migration); therefore, VMs are not initialized in advance. Our experiments considered a fixed number of drones (i.e., 5000), but with different sizes of node ranges (50 and 100 km), to measure the capacities and limitations of the defined architecture. The devices were moving around for 12 h, and the data sampling period was set to 1 min. A measurement is equivalent to 100 bytes of data generation.

For this scenario, we used a PC with Intel Core i5-4460 3.2GHz, 16GB RAM, and a 64-bit Windows 10 operating system to run the simulations; due to the random generator of the simulator, we counted the average values of the experiments, repeated five times.

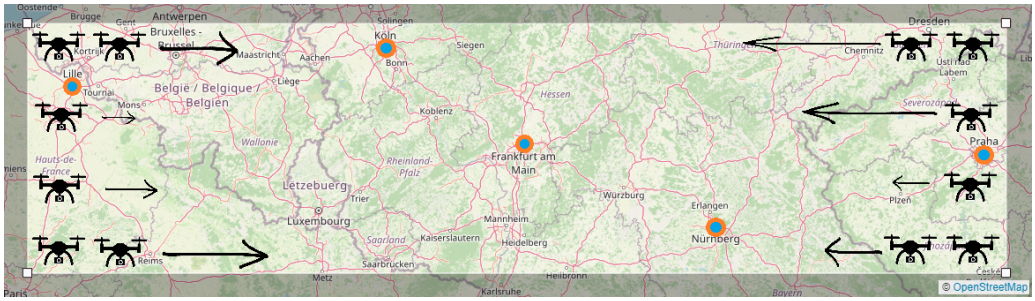

Figure 3. The area of movements and the position of the fog nodes

To compare the results of the scenario runs defined in the previous section, we measured the following metrics in the simulator:

- *No. VMs*: Total number of VMs running on computing nodes.
- *Node cost* (*USD*): Total cost of the VMs with application of the AWS pricing.
- *Node energy* (*kWh*): Energy consumption of the computing nodes based on the LPDS cloud.
- *Timeout* (*minute*): The time interval between the last data-unit procured and processed, the shorter this interval is, the more real-time the simulation is.
- *Data generated* (*MB*): The total sizes of the generated data by smart devices.
- *Data processed by device* (*MB*): The total sizes of the data that were processed on the device.
- *Device processing time* (*minutes*): The average processing time of the devices.
- *No. initialized migration*: The total number of events when a service was initialized to migrate from a node to a device or vice versa.
- *Simulation runtime* (*second*): The necessary time to execute a simulation run in the simulator.

The results of the evaluation scenario are presented in Table 2. Generally speaking, when nodes overlap each other, which can only happen in the case of Cologne, Frankfurt, Nuremberg, with the 100 km range, a device has multiple choices, it can process the data locally or remotely. The 100 km long range ensures effective data offloading and device handover; however, it also means the highest cost, USD 411.63. However, this effectiveness also provides the fastest data processing, with a 1.35 min timeout.

As can be seen in Table 2, due to the proactive service migration strategy, more VMs are used (35 in the case of the 100 km long range). Due to the uncertainty of the proactive strategy (i.e., it can initialize launching VMs in advance, but eventually no migration occurs due to the movement of the device), it deals with the highest energy consumption (25.89 kWh) and initialized migrations (22,487).

It can also be observed that, in the first case, without motion prediction, around 60% (203 MB) of the data—and in the second case, with motion prediction, around 50% (121 MB) of the data—are processed by Dew devices. These ratios can also be observed when taking a look at how much time the devices spent processing data besides generating data (405.68 and 243.7 min).

Finally, considering the time needed to run the simulation, applying the Markov-model, the average simulation time increased (30 s), which can be explained by matrix multiplication, and since DISSECT-CF-Fog is an event-driven simulator, by the handling of extra events caused by migration.

Table 2. Results of the scenario, simulating drone movements

	Without Motion Prediction		With Motion Prediction		Difference	
Node Range (km)	50	100	50	100	Δ_{50}	Δ_{100}
No. of VMs	34	33	37	35	3	2
Node cost (USD)	398.6	405.5	401.2	411.63	2.6	6.13
Node energy (kWh)	19.44	24.74	20.27	25.89	0.83	1.15
Timeout (minute)	2.06	1.79	1.85	1.35	−0.21	−0.44
Data generated (MB)	343.3	343.3	343.3	343.3	0	0
Data processed by device (MB)	203.75	171.14	203.66	170.66	−0.09	−0.48
Device processing time (minute)	431.0	361.2	405.68	353.7	−25.32	−7.5
No. of initialized migrations	15,233	17,185	20,291	22,487	5058	5302
Simulation runtime (second)	305	311	339	341	34	30

4.2. Smart City Scenario

To further examine the effectiveness of motion prediction in the simulator, we used a smart city scenario, which is another representative IoT use case. Regarding the concept of intelligent cities, smart buses (Smart Public Transit (visited on 24 August 2022): https://www.nexcom.com/applications/DetailByDivision/smart-public-transit) help to organize the fleet in increasing traffic, ensure safety, and enhance the traveling experiences via GPS, motion sensors, and video surveillance systems.

In this artificial use case, we modeled the infrastructure of a bus route with five bus stops: (A, B, C, D, E). Within the city, there are two fog servers ($node_1$, $node_2$) located 5 km apart each other. The range of the fog servers is set to 2 km. Bus stops A and B are within the range of $node_1$, and stations D and E are within the range of $node_2$. Stop C is outside the range of both fog servers.

Smart devices on the buses are able to run services (e.g., real-time route planning based on GPS data measured). The bus drives with deterministic movement, at an average speed of 30 km/h in direction $A \rightarrow B \rightarrow C \rightarrow D \rightarrow E$. Stops B and C and C and D are far apart (3.5 and 4 km, respectively), so it is worth migrating the service to the devices.

A total of 10 devices were simulated on 2 buses, and the devices generated 50 bytes of data every 2 min. The simulation modeled 50 min of the operation; the network settings and the specifications of machines were the same used in the previous scenario.

The simulation was run in two ways: (i) without motion prediction and (ii) with motion prediction. Since VM startup could be time-consuming in the case of migration, many data may be processed with delays. The results are shown in Table 3. In the first case, it can be observed that only 76.4% of the IoT data were processed in time, but in the second case, this ratio increased to 96.0%, thanks to the proactive migration.

In the previous scenario, we mainly focused on the physical environment, such as the different ranges of the computing nodes, the number of used virtual machines, utilization costs, and energy consumption to see how an IoT-Dew-Fog system would behave within such circumstances. The critical point of the system is the following: due to an estimation, we initialized launching the VMs in advance, but later it turned out that VMs were created unnecessarily, which could increase the CPU load and energy consumption. Contrarily, no VM initialization was triggered, but it should have been to serve the higher needs. This time, instead of focusing on the physical environment (i.e., resources, utilization price, etc.), we examined the accuracy of our proposed proactive strategy. The triggering method of migration was called 250 times during the evaluation. The results of the simulation in this scenario can be seen in Table 3. The table depicts the main monitored metrics during the simulations.

We also compared the ratio of the predicted and actual migrations. As we can see from Table 4, false-negative migrations (missing instructions to start a VM) and false-positive attempts (unnecessary instructions to start a VM) occurred two and seven times using the

two-order Markov model, which is 3.6% of the total triggering events. These results confirm that the proposed migration strategy was successful during the evaluation scenario.

Table 3. Results of the second scenario simulating smart public transport.

	Without Prediction	With Prediction
Data generated (byte)	25,000	25,000
Data processed in time (byte)	19,100	24,000
Data processed delayed (byte)	5900	1000
Simulation runtime (second)	11	27

Table 4. Migration prediction rate of the second scenario simulating smart public transport.

		Actual	
		Migration Required	No Migration Required
Predicted	Migration required	41	7
	No Migration required	2	200

5. Conclusions

Distributed computing paradigms continuously evolve with the technical involvements. To track such changes faster, simulation approaches are used in order to investigate complex systems realistically.

In this paper, we utilized the DISSECT-CF-Fog simulator, presenting a proactive service deployment scheme via a weighted Markov model to handle the lack of coverage of computing nodes. We also proposed a model of Dew servers to deal with the limited internet access resulting from the movements of smart devices.

The results show that the DISSECT-CF-Fog can be applied to model the offline operations of the mobile devices, which is a typical feature of Dew computing. Furthermore, we compared the Markov model-based proactive migration with a regular, reactive migration strategy, with a drone-based weather forecasting scenario; in a smart transportation scenario, we validated the motion prediction effectiveness of our proposed strategy. In conclusion, the Markov model gives reliable prediction; thus, the processing time can be reduced. However, as a trade-off, the energy consumption and the utilization costs increased.

In the future, we would like to broaden the Dew-related functionality of the simulator to evaluate more diverse IoT-Dew-Fog use cases, considering objectives such as sparsely- or densely-distributed services and minimization of network-service latency.

Author Contributions: Conceptualization, A.K. and K.S.; methodology, A.M. and A.K.; software, M.B. and A.M.; validation, A.M. and Z.Š.; writing—review and editing, all authors; visualization, M.B.; supervision, A.K. and K.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research leading to these results was supported by the Hungarian Scientific Research Fund under Grant OTKA FK 131793, and by the Center of Scientific Excellence: Advanced Methods and Technologies in Data Science and Cooperative Systems (DATACROSS) (EK-KF-KK.01.1.1.01.009), and by the European COST programme under action identifier CA19135 (CERCIRAS).

Data Availability Statement: The source code of DISSECT-CF-Fog is available at: https://github.com/sed-inf-u-szeged/DISSECT-CF-Fog (visited on 24 June 2022).

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

References

1. Bermbach, D.; Pallas, F.; Pérez, D.G.; Plebani, P.; Anderson, M.; Kat, R.; Tai, S. A Research Perspective on Fog Computing. In Proceedings of the Service-Oriented Computing—ICSOC 2017 Workshops, Malaga, Spain, 13–16 November 2017; pp. 198–210. [CrossRef]
2. 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]
3. Ray, P.P. An Introduction to Dew Computing: Definition, Concept and Implication. *IEEE Access* **2018**, *6*, 723–737. [CrossRef]
4. Gushev, M. Dew Computing Architecture for Cyber-Physical Systems and IoT. *Internet Things* **2020**, *11*, 100186. [CrossRef]
5. Skala, K.; Cloud, Z.Š. Fog and Dew Computing: A Distributed Hierarchy. 2015. Available online: https://www.researchgate.net/publication/343848915_Cloud_Fog_and_Dew_Computing_A_Distributed_Hierarchy (accessed on 28 July 2022).
6. Wang, Y. Cloud-dew architecture. *Int. J. Cloud Comput.* **2015**, *4*, 199–210. [CrossRef]
7. Skala, K.; Davidovic, D.; Afgan, E.; Sovic, I.; Sojat, Z. Scalable Distributed Computing Hierarchy: Cloud, Fog and Dew Computing. *Open J. Cloud Comput.* **2015**, *2*, 16–24. [CrossRef]
8. Šojat, Z.; Skala, K. The Rainbow: Integrating Computing into the Global Ecosystem. In Proceedings of the 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), Opatija, Croatia, 20–24 May 2019; pp. 222–229. [CrossRef]
9. Šojat, Z.; Skala, K. The Rainbow through the Lens of Dew. In Proceedings of the 43rd International Convention on Information, Communication and Electronic Technology (MIPRO), Opatija, Croatia, 28 September–2 October 2020; pp. 1916–1921. [CrossRef]
10. Mahmud, R.; Pallewatta, S.; Goudarzi, M.; Buyya, R. IFogSim2: An Extended iFogSim Simulator for Mobility, Clustering, and Microservice Management in Edge and Fog Computing Environments. *J. Syst. Softw.* **2022**, *190*, 111351. [CrossRef]
11. Puliafito, C.; Gonçalves, D.M.; Lopes, M.M.; Martins, L.L.; Madeira, E.; Mingozzi, E.; Rana, O.; Bittencourt, L.F. MobFogSim: Simulation of mobility and migration for fog computing. *Simul. Model. Pract. Theory* **2020**, *101*, 102062. [CrossRef]
12. Jha, D.N.; Alwasel, K.; Alshoshan, A.; Huang, X.; Naha, R.K.; Battula, S.K.; Garg, S.; Puthal, D.; James, P.; Zomaya, A.; et al. IoTSim-Edge: A simulation framework for modeling the behavior of Internet of Things and edge computing environments. *Softw. Pract. Exp.* **2020**, *50*. [CrossRef]
13. Sonmez, C.; Ozgovde, A.; Ersoy, C. EdgeCloudSim: An environment for performance evaluation of edge computing systems. *Emerg. Telekommunications Technol.* **2018**, *29*, e3493. [CrossRef]
14. Hirsch, M.; Mateos, C.; Rodriguez, J.M.; Zunino, A. DewSim: A trace-driven toolkit for simulating mobile device clusters in Dew computing environments. *Softw. Pract. Exp.* **2022**, *50*, 688–718. [CrossRef]
15. Sanabria, P.; Tapia, T.F.; Neyem, A.; Benedetto, J.I.; Hirsch, M.; Mateos, C.; Zunino, A. New Heuristics for Scheduling and Distributing Jobs under Hybrid Dew Computing Environments. *Wirel. Commun. Mob. Comput.* **2021**, *2021* , 8899660. [CrossRef]
16. Markus, A.; Kertesz, A. Investigating IoT Application Behaviour in Simulated Fog Environments. *Cloud Comput. Serv. Sci.* **2020**, 258–276. [CrossRef]
17. Mann, Z. Cloud simulators in the implementation and evaluation of virtual machine placement algorithms. *Softw. Pract. Exp.* **2018**, *48* , 1368–1389. [CrossRef]
18. Rejiba, Z.; Masip-Bruin, X.; Marín-Tordera, E. A Survey on Mobility-Induced Service Migration in the Fog, Edge, and Related Computing Paradigms. *ACM Comput. Surv.* **2019**, *52* , 1–33. [CrossRef]
19. Chang, B.; Yang, L.; Sensi, M.; Achterberg, M.A.; Wang, F.; Rinaldi, M.; Mieghem, P.V. Markov Modulated Process to Model Human Mobility. *Complex Networks Their Appl. X* **2022**, 607–618. [CrossRef]
20. Yan, M.; Li, S.; Chan, C.A.; Shen, Y.; Yu, Y. Mobility Prediction Using a Weighted Markov Model Based on Mobile User Classification. *Sensors* **2021**, *21*, 1740. [CrossRef] [PubMed]