# DISSECT-CF-Fog: A Simulation Environment for Analysing the Cloud-to-Thing Continuum

PhD Thesis Summary


András Márkus
Supervisor: Attila Kertész, PhD.

Doctoral School of Computer Science
Department of Software Engineering
Faculty of Science and Informatics
University of Szeged




Szeged
2022

# Introduction

The appearance of small computational devices connected to the Internet has led to the Internet of Things (IoT) paradigm, which resulted in a vast amount of data generation requiring the assistance of cloud services for storage, processing and analysis. According to Taylor et al. [21] the number of smart devices will exceed 75 billion all over the world by 2025, resulting in an inevitable increase of network traffic.

Cloud systems become good candidates to serve IoT applications because the huge amount of sensed data require elastic storage and processing services for further analysis, and their marriage created so-called smart systems [14]. One of their latest improvements addresses data locality meaning that data management operations are better placed close to their origins to reduce service latency. This idea created Fog Computing [15], which implied the appearance of IoT-Fog-Cloud systems with the highest complexity.

These IoT-Fog-Cloud systems, which are often associated with the Cloud-to-Thing Continuum, require significant investments in terms of design, development and operation, therefore the use of simulators for their investigation is inevitable. There are a large number of simulators addressing the analysis of parts of these systems, however it is obvious that only a state-of-the-art simulator is capable of modelling complex architectures in a realistic way, which meets modern challenges.

This PhD thesis consists of three theses separated into three major chapters. The first chapter presents a detailed survey and taxonomy of various IoT, cloud and fog simulators in order to determine the key requirements of a compact and well-defined IoT-Fog-Cloud simulator. The second chapter introduces the IoT and the pricing extension, exploiting a multi-cloud environment in the DISSECT-CF-IoT simulator. Finally in the third chapter the DISSECT-CF-Fog simulator is presented which is able to model a multi-layered fog topology with mobility and actuator events. The resulting DISSECT-CF-Fog simulator is open-source and available on GitHub[1].

## *Thesis I.* I analysed and classified numerous simulation approaches in terms of functionality, usability, maintainability, and code quality, in order to determine the most relevant properties for modelling IoT-Fog-Cloud systems. I also compared the two most prominent simulators in these fields, namely DISSECT-CF-Fog and iFogSim, with an in-depth performance analysis.

Cloud and fog technologies can be used together to aid data management needs of IoT environments, but their application gives birth to complex systems that still need a significant amount of research. It is obvious that significant investments, design and implementation tasks are required to create such IoT-Fog-Cloud systems in reality, therefore, it is

[1]DISSECT-CF-Fog simulator (accessed in October, 2022): https://github.com/sed-inf-u-szeged/DISSECT-CF-Fog

inevitable to use simulations in the design, development, and operational phases of such establishments. This rationale has led many scientists to create simulators to investigate and analyse certain properties and processes of similar complex systems. We believe that modelling IoT-Fog-Cloud architectures in such simulators is far from complete, and there is a need to gather and compare how key properties, especially of fogs, are represented in these works to trigger further research in this field.

The methodology for finding suitable works for our investigations was twofold. First, we looked for recently published surveys targeting Fog Computing and narrowed their scope to fog modelling and simulation. Second, to extend the group of considered solutions, we performed a literature review with search engines such as Google Scholar and Scopus. Our preliminary observation based on the literature analysis is that most simulation tools started to be developed as cloud simulators, and were later extended to model IoT and fogs as well.

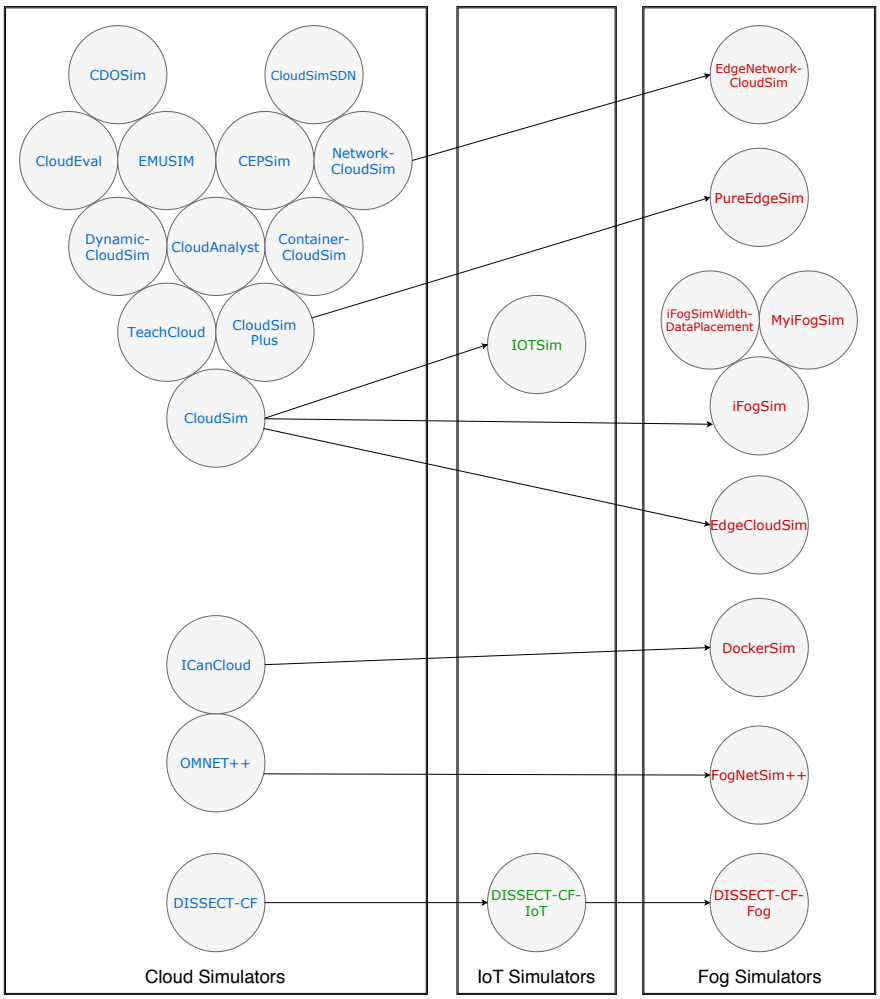


**Figure 1:** *Visualised relationships between the examined cloud, IoT, and fog simulators*

Overall 44 different simulation tools were analysed, but Figure 1 presents only those that built on each other in some way, the others can be considered as stand-alone implementations. The bottom circle represents the core or base simulators, their extensions

within the same system categories are placed on top of them, while the arrows lead to extensions for solutions modelling other systems as well. This graph shows that many variations of a base simulator exist, and we also know that a concrete simulator has many development versions that may have different features. This fact makes it very hard for researchers to choose the right version for their investigations, and for developers to create an improved solution of different versions of the same simulator. Our taxonomy aims to reveal some implementation details in terms of simulation type, publication date, cost model, geolocation, sensor model, network model, virtual machine (VM) management, energy model, and source code metrics. Table 1 depicts and compares properties of fog simulators focusing on code quality.

**Table 1:** *Comparison of the examined fog simulators with software metrics*

| Simulator | Language | Lines of code | Comments (%) | Duplication (%) | Files | Bugs | Vulnerabilities | Code smells |
|---|---|---|---|---|---|---|---|---|
| FogTorchII | Java, XML | 2,748 | 15.9 | 8.3 | 39 | 21 | 31 | 308 |
| FogDirSim | Python, YAML | 5,641 | 1.4 | N/A | 84 | N/A | | |
| OPNET | N/A | | | | N/A | | | |
| PDES | C | N/A | | | | | | |
| FogNetSim++ | C++ | 20,199 | 5.7 | | 59 | | | |
| Edge-Fog | Python | 887 | 17.2 | | 66 | | | |
| YAFS | Python, JS, HTML, JSON | 31,597 | 22.0 | | 208 | | | |
| EdgeNetworkCloudSim | Java, HTML | 113,654 | 27.5 | | 571 | | | |
| PureEdgeSim | Java, XML | 3,308 | 12.2 | 4.3 | 30 | 18 | 101 | 301 |
| iFogSim | Java, XML | 27,754 | 25.3 | 24.3 | 290 | 124 | 248 | 1.5k |
| MyiFogSim | Java, XML | 32,723 | 23.2 | 23.5 | 328 | 174 | 275 | 2k |
| iFogSimWithDataPlacement | Java, Protocol Buffers | 212,780 | 7.6 | N/A | 2,313 | N/A | | |
| EdgeCloudSim | Java, XML | 6,232 | 14.3 | 29.7 | 54 | 14 | 22 | 496 |
| SpanEdge | Java, XML | 1,417 | 10.3 | 34.1 | 17 | 9 | 11 | 232 |
| Matlab (Zhang et al.) | N/A | | | N/A | N/A | N/A | | |
| DockerSim | C++, INI | 48,118 | 22.7 | | 336 | | | |
| EmuFog | Java | 2,570 | 77.6 | | 52 | | | |
| DISSECT-CF-Fog | Java, XML | 9,870 | 33.3 | 2.0 | 118 | 31 | 192 | 482 |

The CloudSim-based extensions (e.g. iFogSim or EdgeCloudSim) are often used for investigating Cloud and Fog Computing approaches, and in general, they are the most referred works in the literature. On the other hand, the DISSECT-CF simulator is proven to be much faster, scalable and reliable than CloudSim (see [17]). This former research showed that the simulation time of DISSECT-CF is 2,800 times faster than the CloudSim simulator for similar cloud use cases. Taking into account the literature search results, the existing performance comparison of the core simulators, and the maturity and number of citations, our next goal was to make a comprehensive comparison with the original version of iFogSim and the DISSECT-CF-Fog simulator.

iFogSim and DISSECT-CF-Fog are quite evolved and complex simulators, however, they follow a slightly different logic to model Fog Computing. To facilitate their comparison, we gathered and compared their properties and components closest to each other.

Since we thoroughly analysed meteorological applications in our research, we decided to use this analogy to compare the performance of the two simulators. So in our scenario, sensors attached to IoT devices (i.e. weather stations) monitor weather conditions, and send the sensed data to fog or cloud resources for processing (i.e. for weather forecasting

and analysis). We defined four layers for the topology with one cloud layer, two fog layers and an IoT (smart) device layer. For the concrete resource parameters we defined one scenario with three different test cases utilising 20, 40 and 60 IoT devices, respectively. The simulation properties were set equally as much as possible in both simulators.

The results of executing the test cases with both simulators can be seen in Table 2. Comparing their runtime, DISSECT-CF-Fog is better suited for high-scale simulations, while iFogSim simulations become intolerably time consuming by modelling higher than a certain number of entities. In the third test case, we experienced process starvation caused by the Java Finalizer thread, which was also mentioned in [12].

**Table 2:** *Comparison of the two simulators*

| Property | DISSECT-CF-Fog | | | iFogSim | | |
|---|---|---|---|---|---|---|
| Test case | I. | II. | III. | I. | II. | III. |
| Runtime (ms) | 248.75 | 312.5 | 392.58 | 2,260.33 | 3,873.66 | 5,400,000* |
| Application delay (min) | 3.41 | 4.33 | 4.33 | 14.89 | 17.52 | N.A. |
| Generated data (byte) | 19,600,000 | 39,200,000 | 58,800,000 | 19,600,000 | 39,200,000 | N.A. |
| Lines of code | 50 lines + XML files for detailed configuration | | | 159 lines + some inline constants | | |

The results of this chapter belong to **Thesis I**, and its content was published in papers [P3], [P7], and [P11].

## *Thesis II.* I designed a generic model of IoT systems and implemented it in the DISSECT-CF-IoT simulator. I developed a novel cost estimation extension using real cloud and IoT provider pricing schemes. I proposed various resource allocation strategies to reduce IoT application execution time and utilisation costs for multi-cloud environments. I also evaluated these strategies with a real-world meteorological use case.

The Internet of Things (IoT) paradigm allows for interconnecting sensors (e.g. heart rate, heat, motion, etc.) and actuators (e.g. motors or lighting devices) in automated and customisable systems [20]. We aim at supporting the simulation of up to thousands of devices participating in IoT scenarios and examine those in terms of scalability, energy efficiency or management costs. Sensors are essential parts of IoT systems, and they are usually passive entities. Their performance is limited by their network gateway's (i.e. the device which polls for the measurements and sends them away) connectivity and maximum update frequency. Actuators are entities also limited by their network connectivity and reaction time (e.g. how long does it take to actually perform an actuation action). Finally, central computing services provide the large-scale background processing and storage capabilities needed for IoT scenarios. According to recent advances in IoT, these services are expected to be used only if unavoidable. The simulator extension presented in this chapter takes

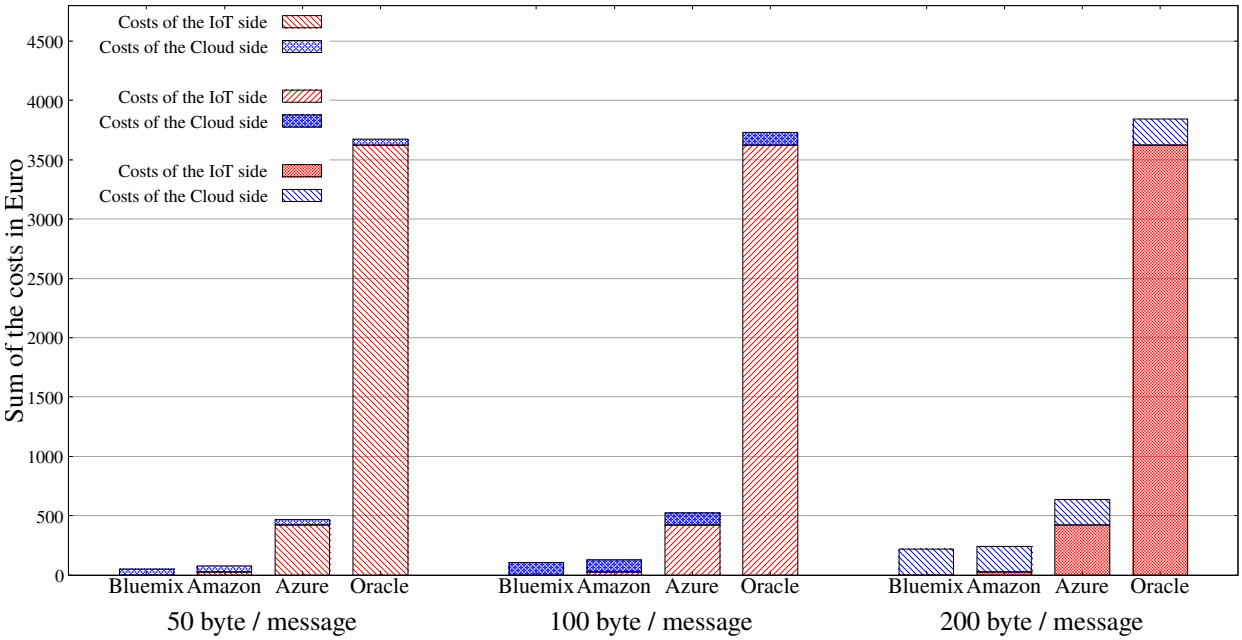


**Figure 2:** *IoT and cloud costs in the evaluation scenario*

into account the following IoT components: sensors, devices (i.e. gateways or brokers), and applications (deployed in a central computing service, i.e. cloud).

As one of the earliest examples of sensor networks is from the field of meteorology and weather prediction, we choose to model the crowd-sourced meteorological service. These kinds of applications aim to make weather analysis more efficient by allowing the purchase of a small weather station kit including light sensors (to potentially capture cloud coverage), wind sensors (to collect wind speed), and temperature sensors (to capture the current ambient temperature). The weather station will then create a summary of the sensor's findings over a certain period of time and report it to a Cloud service for further processing, such as detecting hurricanes or heat waves in the early stages. If many of these stations are set up over a region, it can provide accurate and detailed data flow to the cloud service to produce accurate results.

In our evaluation scenarios, we mainly focus on how resource utilisation and management patterns alter based on changing sensor behaviour and how these affect the incurred costs of operating the IoT system (e.g. how different sensor data sizes and the varying number of stations and sensors affect the operation of the simulated IoT system). For instance, we varied the amount of data produced by the sensors as follows: we set 50, 100, and 200 bytes for different cases (allowing overheads for storage, network transfer, different data formats, and secure encoding etc.). We simulated 486 stations of the weather service for 24 hours and we also applied cloud and IoT-side pricing schemes of providers such as IBM Bluemix, Amazon, Azure, and Oracle.

Figure 2 presents a cost comparison for all considered providers. We can see that Oracle costs are much higher than the other three providers in all cases (50, 100, 200 bytes messages). The main cause of this issue is that Oracle charges after each utilised device, which is not the case for other providers. Our initial estimations show that only such an IoT cloud system operation is beneficial with Oracle, which has at most 200 devices

and transfers 1-2 messages per minute per device.

The further research question of this work is how we can influence the behaviour of an IoT application, if the sensors can have different allocation strategies for multiple clouds. Earlier we could only exploit one cloud data centre to start VMs, therefore, all sensors and smart devices were connected to this specific cloud, and all the generated data of the sensors were processed by virtual machines running in the same cloud. As a result, a single cloud could be easily overloaded, and the unprocessed data could hinder the operation of the IoT application causing longer response times and even service unavailability in real-world services.

To resolve this issue, during the start of the simulation, we can set up different IaaS clouds, and another improvement is the introduction of a cloud broker, which can manage different IoT applications and their VM queues. These queues may have virtual machines with different pricing policies, and within a simulation the broker can decide to which cloud (and to which application) the IoT devices should be connected, thus where the generated data should be sent and processed in an application.

Currently, four different resource allocation (i.e. device) strategies can be chosen to perform cloud provider selection during each IoT device start-up. With the *Random* strategy, the cloud broker chooses one of the available applications running in the simulated clouds randomly for an actual IoT device (sensor or station). The *Cost-aware* strategy looks for the cheapest available VM in a cloud (based on their static pricing properties), thus it compares the prices of the required VM flavors for a given device. This solution may be more suitable for IoT applications having relatively small data processing needs or that are less susceptible to the processing time because cloud providers usually offer lower resource capacities for lower costs. In the *Load-balanced* strategy, the corresponding algorithm ranks the available VMs (residing in different clouds) by a specific value defined by the ratio of the number of already connected devices and the number of the available physical machines in the hosting cloud. This is a dynamic strategy that takes into account the actual load of the available clouds. Applications having longer data processing needs may prefer this strategy.

The last device strategy is based on the *Pliant* logic; The Pliant system is a kind of fuzzy theory that is similar to a fuzzy system [16]. This algorithm calculates a score for each cloud using the environment properties. The calculation includes a normalisation step, where we apply the Sigmoid function. In the normalisation step, it should be mentioned that if the normalised value is close to one, that means it is a more valuable property, and if the normalised value is close to zero, that means it is a less prioritised property. After the normalisation step, we modify the normalised value to emphasise the importance of the result by using the Kappa function. The final step is to calculate a score number for the node. To achieve this, we can use the conjunction, disjunction or aggregation operator. The conjunctive operator is similar to the AND operator. This means that if one of the values is small, then the result will be also small. The opposite is true for the disjunctive operator, which is similar to the OR operator. The aggregation operator lies between the disjunctive or conjunctive operator, that is why we use this operator. Finally, when the score number is calculated for all clouds, we create a distribution function based on the score number and choose one from this distribution randomly. For our Pliant strategy we consider the following properties for each cloud VM: general VM cost, current cost of application, workload, number of running VMs in the hosting cloud, number of devices

**Table 3:** *Evaluation results*

| Strategies | Cost-aware | Random | Load-balanced | Pliant |
|---|---|---|---|---|
| Total cost (Euro) | 26.419 | 66.527 | 65.451 | 65.651 |
| No. of used VMs | 109 | 180 | 170 | 173 |
| Total tasks | 1,722 | 1,830 | 1,819 | 1,838 |
| Timeout (min) | 631 | 86 | 86 | 71 |

that are already connected to a cloud, memory size and number of CPUs.

For the evaluation we also aimed to simulate a worldwide weather forecasting system. We used three clouds configured with Amazon, Azure, and IBM Bluemix cloud provider pricing. The number of running weather stations has been increased to 40,000, each of them works with 8 sensors and generates 50 bytes of data every minute. We run this scenario to simulate 6 working hours. In the beginning, we started 10,000 stations, then we added 10,000 stations more in the next hours to reach 40,000 stations by the fourth hour. The results of the second scenario are shown in Table 3, where the *Pliant* strategy reached the most favourable timeout (71 minutes) with a slightly worse cost (65.651 Euros) than in the case of *Random* and *Load-balanced* strategies.

The results of this chapter belong to **Thesis II**, and its content was published in papers [P1], [P2], [P5], [P6], and [P10].

## *Thesis III.* I designed a generic model of Fog Computing and implemented it in the DISSECT-CF-Fog simulator to enable the modelling of the Cloud-to-Thing Continuum. I developed various task offloading policies for fog and cloud infrastructure management, to optimise IoT application makespan, utilisation costs, and energy consumption. I also proposed novel extensions to enable mobility and actuator behaviour analysis, and I evaluated these extensions with different smart system use cases.

In the surrounding world of IoT devices, location is often fixed, however, the Quality of Service (QoS) of these systems should also be provided at the same level in the case of dynamic and moving devices. Systems composed of IoT devices supporting mobility features are also known as the Internet of Mobile Things (IoMT) [18]. Mobility can have a negative effect on the QoS to be ensured by fog systems, for instance, they could increase the delay between the device and the actual node it is connected to. Besides scalability, latency and resource management issues, energy consumption of a fog environment and the corresponding smart devices is also a great challenge as stated in [13].

When the number of tasks is growing, a single fog node may not be able to process them continuously, therefore, a forwarding function for some of the tasks to other nodes can be useful to manage a higher number of tasks of an IoT application. A fog topology consisting of several nodes with different locations can handle the unforeseen appearance of smart devices (and new tasks) more effectively, than a single, heavyweight cloud node. To manage the offloading decisions separately for each node, we introduce the application strategy components with which different task allocation approaches can be created and implemented taking in consideration the characteristics of the topology.

We defined four basic strategies for task allocation to validate the usability of our proposal. The *Random* strategy is the default, which always chooses one from the connected nodes randomly. The *Push Up* strategy always chooses the connected parent node (i.e. a node from a higher layer), if available. This approach does not take into account the properties of the neighbours, and basically ensures the fastest way to forward unprocessed tasks to the cloud, where more powerful VMs may reduce the processing time of it. The disadvantage of these strategies is the disability to consider increased network traffic and costs of the operation in the decision. The third strategy called *Hold Down* aims to address privacy needs because the system can keep application data as close to the end-user as possible. In this way, the network traffic is minimal, but the execution time of the application can increase dramatically (due to the possible overload of constrained resources at the lowest layer). The *Runtime-aware* strategy ranks the available parent nodes and all neighbour nodes (from its own layer) by network latency and by the ratio of the available CPU capacity and the total CPU capacity. The algorithm picks the node with the highest rank (i.e. the closest and least loaded one). The last strategy we propose is an algorithm that can predict which computing node could be the best for managing a given IoT device (according to the actual state of the system, represented by its properties). This algorithm is also based on the *Pliant* logic, therefore, for each reachable fog and cloud node it calculates a score number using normalisation, Sigmoid and Kappa functions, and the aggregation operator. We define the following three properties for each system node: load, cost, and unprocessed data of a node.

In the layered architecture of IoT, actuators are located in the perception layer, which is often referred to as the lowest or physical layer that requires the most detailed level of abstraction in IoT. Since the actuator has the ability to control the sensing process itself [19], half of the predefined actuator events foster low-level sensor interactions. The *Change file size* event can modify the size of the data to be generated by the sensor. Such behaviour reflects use cases, when more or less detailed data are required for the corresponding IoT application, or the data should be encrypted or compressed for some reason. *Increase frequency* and *Decrease frequency* might be useful when the IoT application requires an increased time interval between the measurements of a sensor. A typical use case of this behaviour is when a smart traffic control system of a smart city monitors the traffic at night when usually fewer inhabitants are located outside. The *Decrease frequency* is the opposite of the previously mentioned one, a typical procedure may appear in IoT healthcare, for instance, the blood pressure sensor of a patient measures continuously increasing values, thus more frequent perceptions are required. The *Stop device* event imposes a fatal error on a device, typically occurring randomly. Finally, the *Restart device* reboots the given device to simulate software errors or updates.

Currently two mobility strategies are implemented in DISSECT-CF-Fog. The goal of the

**Table 4:** *Results of the Transport actuator strategy and number of events during the evaluation scenario*

| Actuator strategy | Transport | | | | | |
|---|---|---|---|---|---|---|
| Fog node range (km) | 25 | | | 50 | | |
| Vehicle (pc.) | 2 | 20 | 200 | 2 | 20 | 200 |
| VM (pc.) | 19 | 19 | 19 | 19 | 19 | 19 |
| Generated data (MB) | 65 | 642 | 6,445 | 83 | 851 | 8,469 |
| Fog + Cloud cost ($) | 1,974.7 | 4,492.9 | 10,231.1 | 2,557.8 | 5,006.5 | 10,312.7 |
| Delay (min.) | 5.0 | 4.03 | 2.02 | 5.0 | 4.04 | 4.01 |
| Runtime (sec.) | 3 | 13 | 119 | 4 | 15 | 128 |
| Change file size (pc.) | 20,012 | 198,221 | 1,986,157 | 20,107 | 189,693 | 1,870,594 |
| Change node (pc.) | 0 | 0 | 0 | 6,111 | 65,424 | 654,135 |
| Change position (pc.) | 91,167 | 910,014 | 9,122,057 | 93,088 | 970,373 | 9,791,859 |
| Connect / disconnect to node (pc.) | 13,140 | 131,455 | 1,314,037 | 7,029 | 66,349 | 659,573 |
| Increase frequency (pc.) | 19,833 | 198,888 | 1,982,648 | 19,573 | 66,117 | 1,872,881 |
| Decrease frequency (pc.) | 19,735 | 199,759 | 1,983,997 | 19,646 | 189,298 | 1,875,489 |
| Restart / stop device (pc.) | 0 | 0 | 0 | 0 | 0 | 0 |
| Timeout (pc.) | 35,379 | 354,788 | 3,536,881 | 0 | 0 | 0 |
| Timeout data (MB) | 15 | 149 | 1,557 | 0 | 0 | 0 |

(i) *Nomadic* mobility model is that entities move together from one location to another, in our realisation multiple locations (i.e. targets) are available. It is very similar to the public transport of a city, where the route can be described by predefined points (or bus stops), and the dedicated points are defined as geographical positions. An entity reaching the final point of the route will no longer move but may function afterwards. Between the locations, a constant speed is considered, and there is a fixed order of the stops. The (ii) *Random Walk* mobility takes into consideration entities with unexpected and unforeseen movements, for instance, the observed entity walks around the city, unpredictably. The aim of this policy is to avoid moving in straight lines with a constant speed during the simulation because such movements are unrealistic. In this policy, a range of the entity is fixed, where it can move with a random speed. From time to time, or if the entity reaches the border of the range, the direction and the speed of the movement dynamically change.

The simulator monitors the position of the fog nodes and IoT devices continuously and makes decisions knowing these properties. A connection of an IoT device is closed with the corresponding node in case the latency exceeds the maximum tolerable limit of the device, or the IoT device is located outside of the range of the node. When a device finds a better fog node instead of the current one, or the IoT device runs without connection to any node, it finds an appropriate one. To cover such actions, we introduce five actuator events related to mobility.

To evaluate this extension, we simulated a one-year-long operation of a smart transport route across cities located in Hungary. This track is exactly 875 kilometres long, and it takes slightly more than 12 hours to drive through it by a car based on Google Maps, which means the average speed of a vehicle is about 73 km/h. We placed fog nodes in 9 different cities maintained by a domestic company, and applied the *Transport* actuator policy: if the asset was located closer than five kilometres, it would send position data

every two minutes. In the case of five to 10 kilometres, the data frequency is five minutes, and from 10 to 30, the data generation is set to 10 minutes, lastly if it is farther than 30 kilometres, it informs changes in 15 minutes. The results are shown in Table 4.

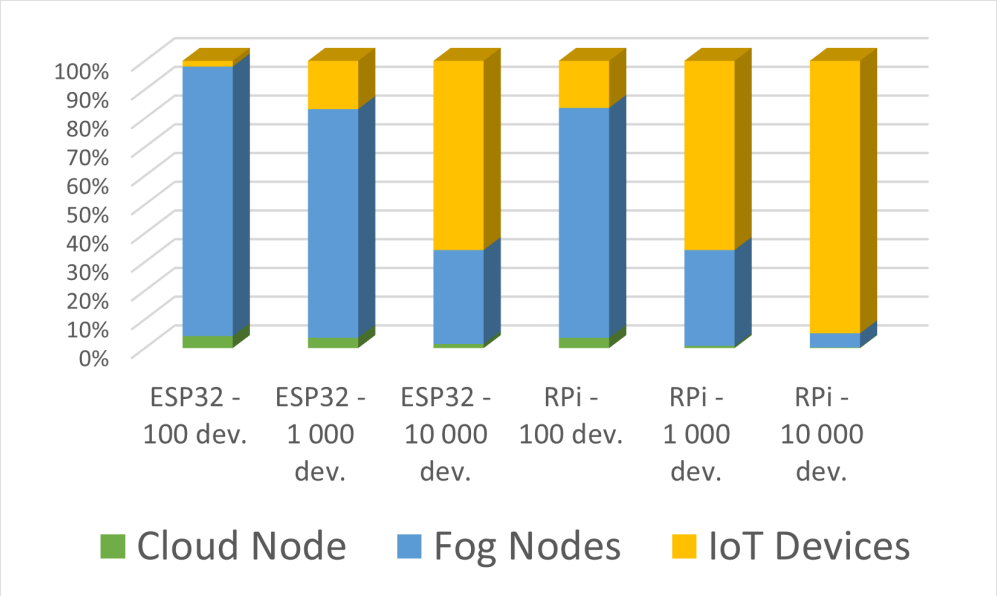


**Figure 3:** *Energy consumption percentage of cloud, fog nodes and IoT devices*

Concerning power consumption of IoT resources, we had to build up the energy model from scratch. In this work, we had to extend the IoT device representation of DISSECT-CF-Fog, which represents any smart objects, and responsible for power consumption metering for IoT devices during simulations. To resolve this issue, we decided to create a more detailed physical layer called a microcontroller for implementing our energy model. Such realisation keeps the already existing functionalities (e.g. data sensing of IoT sensors, temporary data storing, and data forwarding to fog or cloud nodes), and introduces predefined states for microcontrollers, which allow mapping a certain power consumption to a certain state.

Our findings and experiments revealed that the power consumption values of microcontrollers are highly dependent on their actual behaviour and their use cases. Typical modifying circumstances may be the usage of a wired connection instead of wireless, and/or different types of power supply cables or converters. To handle such extreme cases and to be able to simulate uncertainty, we introduce three different states of a microcontroller in our model.

The state *OFF* indicates a fully turned-off device with static minimal energy consumption using the min power preset value. The *RUNNING* state represents a high energy consumption state, where the actual power consumption can change dynamically with regard to the actual CPU utilisation. The minimal and maximal consumption values in this state are set by the predefined idle and max power values. To simulate specific events when high power spikes appear (caused by e.g. activating a previously unused port of the device), we introduce the *ACTIVE* state. It also represents a high energy consumption state allowing dynamic changes, but its minimal value should be higher than in the *RUNNING* state; by default it is set to double the idle power value.

In our simulation, the microcontrollers can use either ESP32 or Raspberry Pi energy models, and they are equipped with a temperature-humidity sensor (similar to our real-world measurements). In our weather forecasting use case, we defined three different scenarios by scaling up the number of operating devices. In the first case, we utilised 100 IoT devices, then we increased the number of devices to 1,000, and finally, in the last case, the maximum device number was 10,000, operated for 60 minutes within the experiments. The microcontrollers measured the environmental parameters every 60 seconds, similar to the real device evaluation, hence our goal was to map the real monitoring execution in the DISSECT-CF-Fog simulation environment. Figure 3 highlight the results by comparing the energy consumption ratio of the utilised cloud node, fog nodes and IoT devices (i.e. microcontrollers) by depicting their ratio in percentage.

The results of this chapter belong to **Thesis III**, and its content was published in papers [P4], [P8], and [P9].

**Table 5:** *Publications, theses and citations*

| | Thesis I | Thesis II | Thesis III | Citations | |
|---|---|---|---|---|---|
| | | | | Google Scholar | MTMT |
| [P3] | ♦ | | | 48 | 36 |
| [P7] | ♦ | | | 4 | 4 |
| [P11] | ♦ | | | 1 | 2 |
| [P1] | | ♦ | | 13 | 11 |
| [P2] | | ♦ | | 10 | 7 |
| [P5] | | ♦ | | 10 | 7 |
| [P6] | | ♦ | | 3 | 4 |
| [P10] | | ♦ | | 5 | 3 |
| [P4] | | | ♦ | 1 | 1 |
| [P8] | | | ♦ | - | - |
| [P9] | | | ♦ | 1 | - |
| Sum | 3 | 5 | 3 | 96 | 75 |

# Publications, Theses and Citations

Table 5 summarises the relation between the thesis points and the corresponding publications and it also presents the citations received so far according to Google Scholar and MTMT.

## Journal publications

[P1] **A. Markus**, G. Kecskemeti and A. Kertesz. Cost-aware IoT extension of DISSECT-CF. *Future Internet*, Volume 9, 2017. DOI: 10.3390/fi9030047

[P2] **A. Markus** and J. D. Dombi. Multi-Cloud Management Strategies for Simulating IoT Applications. *Acta Cybernetica*, Volume 24, 2019. DOI: 10.14232/actacyb.24.1.2019.7

[P3] **A. Markus** and A. Kertesz. A Survey and Taxonomy of Simulation Environments Modelling Fog Computing. *Simulation Modelling Practice and Theory*, Volume 101, 2020. DOI: 10.1016/j.simpat.2019.102042

[P4] **A. Markus**, M. Biro, G. Kecskemeti and A. Kertesz. Actuator behaviour modelling in IoT-Fog-Cloud simulation. *PeerJ Computer Science*, 7:e651, 2021. DOI: 10.7717/peerj-cs.651

## Full papers in conference proceedings

[P5] **A. Markus**, G. Kecskemeti and A. Kertesz. Flexible Representation of IoT Sensors for Cloud Simulators (PDP). In *Proceedings of 25th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP)*, 199-203, 2017. DOI: 10.1109/PDP.2017.87

[P6] **A. Markus** and A. Kertesz. Simulating IoT Cloud Systems: A Meteorological Case Study. In *Proceedings of Second International Conference on Fog and Mobile Edge Computing (FMEC)*, 171-176, 2017. DOI: 10.1109/FMEC.2017.7946426

[P7] **A. Markus**, P. Gacsi and A. Kertesz. Develop or Dissipate Fogs? Evaluating an IoT Application in Fog and Cloud Simulations. In *Proceedings of 10th International Conference on Cloud Computing and Services Science (CLOSER)*, 193-203, 2020. DOI: 10.5220/0009590401930203

[P8] **A. Markus** and A. Kertesz. Modelling Energy Consumption of IoT Devices in DISSECT-CF-Fog. In *Proceedings of 11th International Conference on Cloud Computing and Services Science (CLOSER)*, 320-327, 2021. DOI: 10.5220/0010500003200327

[P9] **A. Markus**, J. D. Dombi and A. Kertesz. Location-aware Task Allocation Strategies for IoT-Fog-Cloud Environments. In *Proceedings of 29th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)*, 185-192, 2021. DOI: 10.1109/PDP52278.2021.00037

## Further related publications

[P10] **A. Markus**, A. Marques, G. Kecskemeti and A. Kertesz. Efficient Simulation of IoT Cloud Use Cases. In *Autonomous Control for a Reliable Internet of Services*, 313-336, 2018. DOI: 10.1007/978-3-319-90415-3_12

[P11] **A. Markus** and A. Kertesz. Investigating IoT Application Behaviour in Simulated Fog Environments. In *Cloud Computing and Services Science*, 258-276, 2021. DOI: 10.1007/978-3-030-72369-9_11

## Other References

[12] D. Perez Abreu, K. Velasquez, M. Curado, and E. Monteiro. A comparative analysis of simulators for the cloud to fog continuum. *Simulation Modelling Practice and Theory*, 101:102029, 2020.

[13] H. F. Atlam, R. J. Walters, and G. B. Wills. Fog computing and the internet of things: A review. *Big Data and Cognitive Computing*, 2:2, 2018.

[14] A. Botta, W. de Donato, V. Persico, and A. Pescapé. Integration of cloud computing and internet of things: A survey. *Future Generation Computer Systems*, 56:684–700, 2016.

[15] A. V. Dastjerdi and R. Buyya. Fog computing: Helping the internet of things realize its potential. *Computer*, 49:112–116, 2016.

[16] J. Dombi. Pliant system. *In Proceedings of IEEE International Conference on Intelligent Engineering Systems*, pages 289–294, 1997.

[17] Z. Mann. Cloud simulators in the implementation and evaluation of virtual machine placement algorithms. *Software: Practice and Experience*, 48:7, 2017.

[18] K. Nahrstedt, H. Li, P. Nguyen, S. Chang, and L. Vu. Internet of mobile things: Mobility-driven challenges, designs and implementations. *IEEE First International Conference on Internet-of-Things Design and Implementation (IoTDI)*, pages 25–36, 2020.

[19] F. Pisani, F. M. C. de Oliveira, E. S. Gama, R. Immich, L. F. Bittencourt, and E. Borin. Fog computing on constrained devices: Paving the way for the future iot. *Advances in Edge Computing: Massive Parallel Processing and Applications*, 35:22–60, 2020.

[20] D. Miorandi S. Sicari, F.De Pellegrini, and I. Chlamtac. Internet of things: Vision, applications and research challenges. *Ad Hoc Networks*, 10:1497–1516, 2012.

[21] R. Taylor, D. Baron, and D. Schmidt. The world in 2025 - predictions for the next ten years. *10th International Microsystems, Packaging, Assembly and Circuits Technology Conference (IMPACT)*, pages 192–195, 2015.

# Összefoglalás

Az internethez csatlakoztatott kis számítási eszközök megjelenése vezetett a Dolgok Internete (IoT) paradigmájához, ami hatalmas mennyiségű adatot eredményezett. Azonban ezen adatok tárolása, feldolgozása és elemzése nem triviális feladat. Az elosztott rendszerek fejlődése lehetővé tette, hogy az IoT rendszereket ötvözzük bizonyos számítási (például Felhő/Cloud és Köd/Fog) szolgáltatásokkal, amelyek a korábban említett feladatokban segítséget nyújtanak. Ilyen IoT-Köd-Felhő rendszerek elemzése a való világban igen költséges lehet, ezért számos szimulációs megoldás született annak érdekében, hogy ezeket a komplex rendszereket vizsgálhassuk. Ez a doktori értekezés három fejezetben foglalja össze az IoT-Köd-Felhő rendszerek modellezését célzó DISSECT-CF-Fog szimulátort, amellyel képesek vagyunk nagy számú IoT eszközt és alkalmazást valósághűen modellezni.

Az első fejezetben összesen 44 darab, felhő, IoT és köd rendszereket modellező szimulációs megoldást részletesen elemeztünk és osztályoztunk. A vizsgált szimulátorokat három csoportra osztottuk, és az osztályozás alapján összehasonlító táblázatokat és ábrákat mutattunk be, hogy feltárjuk a különbségeket, és rávilágítsunk, hogyan modellezik az IoT-Köd-Felhő rendszerek egyes elemeit. Az összehasonlító elemzésünk alapján megállapíthatjuk, hogy a köd rendszerek átfogó modellje még mindig hiányzik, és az összetett környezeteket nehéz egyetlen szimulátorral kezelni. Fő javaslatunk a további kutatáshoz a szimulátorokban található modellek folyamatos bővítése az igények alapján a megfelelő szoftverminőség fenntartása mellett. Az átfogó elemzésünk alapján ezt követően összehasonlítottuk az iFogSim és a DISSECT-CF-Fog szimulátorokat.

A második fejezetben bemutattuk a DISSECT-CF-IoT szimulátort, amely képes részletes árképzési sémákkal modellezni az IoT szenzorokat, eszközöket és alkalmazásokat, ehhez figyelembe vettük valós szolgáltatóknak, például az Amazon-nak vagy az Azure-nek az árazási modelljeit. Mivel az IoT eszközök száma a tízezres nagyságrendet is meghaladhatja, egyetlen felhő könnyen túlterheltté válhat, és a feldolgozatlan adatok akadályozhatják az IoT alkalmazás működését, ami hosszabb válaszidőt, sőt – valósidejű szolgáltatások esetén – a szolgáltatás elérhetetlenségét is okozhatja, ezért ebben a munkában bemutattuk a több felhőből álló architektúra kezelésének lehetőségét. Ahhoz, hogy az IoT eszközök kihasználhassák a megnövekedett számítási erőforrásokat, négy felhő választási stratégiát is javasoltunk, amelyek célja az IoT alkalmazások végrehajtási idejének és költségeinek csökkentése.

A harmadik fejezetben bevezettük a DISSECT-CF-Fog szimulátort köd rendszerek modellezésre, amely figyelembe veszi a köd csomópontok pozícióját is. A többrétegű köd architektúrák kezelésének érdekében különböző alkalmazás vezérlési stratégiákat vezettünk be a komplex IoT-Köd-Felhő rendszerek tehermentesítő döntéseinek kezelésére, így amennyiben egy köd vagy felhőcsomópont túlterheltté válik, képes a várakoztatott számítási feladatokat másik számítási csomópontnak továbbítani. Végül a szimulációs eszközt kiterjesztettük az IoT eszközök energiafogyasztásának mérésére valamint az aktuátorok és a mobilitási jellemzők modellezésére, amelyek segítségével az IoT eszközök mozgását is figyelembe veszi a szimulátor.

Összefoglalásként, a DISSECT-CF-Fog részletes IoT, köd és felhő modellje, kiegészítve a felhő választási és alkalmazás stratégiákkal, lehetővé teszi komplex IoT-Köd-Felhő rendszerek modellezését az IoT alkalmazás végrehajtási ideje, a köd és felhő erőforrások felhasználása, a költségek és az energiafogyasztás szempontjából.