Behaviour analysis of complex, distributed systems

Attila Kertész

Dissertation for the Doctoral Degree of the
Hungarian Academy of Sciences

Department of Software Engineering

Institute of Informatics, University of Szeged, Hungary

Szeged
2023

Preface

ONE way to examine nature, as a complex ecosystem, is to go out and explore its beauty. Take a walk at the uphills of Barcelona, look down to the city that nests in a valley leading to the sea. You can hear the birds and cicadas playing in the nearby pine trees, as well as, some of the sharp noises of the crowded city life faded and sweeping through the distance disturbing stillness. You can feel that you are surrounded and exist in a complex system, but cannot grasp and understand it as a whole. You can zoom in by taking a narrow side track leading down through the forest, touching the rough bark of a log lying across the path, or petting the puppy approaching you with a friendly bark running after a stick. When you go further down and reach the backyard of a primary school, you are suddenly surrounded by a loud large group of pupils and parents just about to depart home. What have you learnt about Barcelona? A lot about the sense of that part of the city, but almost nothing about how it is working or governed. For that later you need a systematic investigation through certain, well-defined scenarios. One way is to take a bottom-up approach: go to a tapas bar downtown, eat 'pincho de anchoas', then start exploring your surroundings. Extend your scope step-by-step: walk the nearby streets, observe the neighbourhood, scan the seaside, grasp the San Marti region, then get a view on the whole city, which is a complex system. In this dissertation, I summarise the methods, approaches, and tools I developed in the past decade for a similar purpose, but in the world of ICT, first with my close colleagues and international collaborations, then with my own research group.

Around 2010, Cloud Computing [47] became a diverse research area encompassing many aspects of sharing software and hardware solutions, including computing and storage resources, application runtimes, or complex application functionalities. Unlike the previously ruling Grid Computing, the concept of Cloud Computing has been pioneered by commercial companies with the promise to allow elastic construction of virtual infrastructures. Cloud solutions provided businesses with the option to outsource the operation and management of IT infrastructure and services, allowing them to focus on their core competencies. Soon after this new technology set foot in ICT, a continuously growing number of powerful devices (smartphones, household appliances, etc.) started to join the Internet, significantly impacting on the global traffic volume (e.g. by data sharing, voice, multimedia) and foreshadowing a world of smart devices, or things in the Internet of Things (IoT) perspective. The CERP-IoT [173] defined IoT as a dynamic global network infrastructure with self-configuring capabilities based on standard and interoperable communication protocols. Things in this network can interact and communicate among themselves and with the environment by exchanging data and information sensed, and react autonomously to events and influence them by triggering actions with or without direct human intervention. In the past years, more than a hundred billion devices became online, most of them exploiting data services provided by clouds, realizing so-called IoT-Cloud systems. Such systems can be utilised in many application areas ranging from local smart homes to mid-range smart cities, or wider smart regions.

The rapid spreading of IoT applications raised new challenges. To cope with the huge number of communicating entities, data management operations are better placed close to their origins, resulting in the exploitation of edge servers of the cloud. This distributed computing paradigm is called Fog Computing [114], where groups of such edge nodes form a fog, to ensure reduced service latency and improved service quality compared to remote cloud usage for data processing and analysis. This evolution has led to so-called IoT-Fog-Cloud systems that are able to serve the masses. Nevertheless, applications built on top of these complex, integrated infrastructures faced further challenges, such as lack of standardisation, increased security and privacy issues implying social mistrust, as side effects of this technological revolution. The latest novelty in ICT provided Blockchain (BC), as a solution to most of these newly arisen problems. BC realises a distributed transactional database secured by cryptography, and governed by a consensus mechanism [35]. By integrating BC to IoT-Fog-Cloud systems, we can achieve trusted, immutable, and fully decentralised data management. In this setting, we arrive at the most complex case, when IoT applications are executed in a BC-Fog-Cloud system.

In this dissertation, I provide means for analysing the behaviour of complex systems relying on the aforementioned emerging technologies. In Chapter 1, I summarise my research results achieved mostly through international collaborations, concerning methods for creating Cloud Federations and enhancing their operation performance. In Chapter 2, 3 and 4, I present results mostly achieved together with the PhD students I supervised, concerning modelling and analysing IoT-Cloud systems, IoT-Fog-Cloud systems, and IoT-Blockchain-Fog-Cloud systems, respectively.

Referring to the city of Barcelona in the allegory of a complex system in the beginning of this preface is not by coincidence. My first research exchange within the framework of the CoreGRID EU project took place in BRC in 2007. My latest visit – I evoked here – happened in the fall of 2023, when I was invited as an opponent of a PhD defense of a student in UPC, whose supervisor was my former colleague I worked with, back in 2007.

Acknowledgements

OVER the past 12 years, I had the opportunity to be part of a remarkable technological evolution in the field of parallel and distributed systems. I contributed to it through active participation in both national and international projects addressing research challenges of Cloud, IoT, Fog and Blockchain systems. This means that I have achieved the results summarised in this dissertation not as an individual researcher, but as a member of research groups, where these groups became the driving forces of innovations by successfully accomplishing their projects. I am grateful to my colleagues, both formerly at MTA SZTAKI and currently at the University of Szeged, for providing such opportunities, and to all my coauthors, from all over the world, for their productive support and collaboration.

Szeged, 2023.06.02.

Attila Kertesz

”Ask and it will be given to you; seek and you will find;
knock and the door will be opened to you.
For everyone who asks receives; he who seeks finds;
and to him who knocks, the door will be opened.”

Matthew 7:7-8

Contents

List of Figures

List of Tables

1 Investigating Cloud Federations

An expert group set up by the European Commission published its view on the future of Cloud Computing (CC) in 2010 [173]. This report categorised cloud architectures into five groups: Private Clouds (i) consist of resources managed by an infrastructure provider (IP) that are typically owned or leased by an enterprise from a service provider (SP). Usually, services with cloud-enhanced features are offered, therefore this group includes Software as a Service (SaaS) solutions like eBay. Public Clouds (ii) offer their services to users outside of the company and may use cloud functionality from other providers. In this solution, enterprises can outsource their services to such cloud providers mainly for cost reduction. Examples of these providers are Amazon or Google Apps. Hybrid Clouds (iii) consist of both private and public cloud infrastructures to achieve a higher level of cost reduction through outsourcing by maintaining the desired degree of control (e.g., sensitive data may be handled in private clouds). The report stated that hybrid clouds had been rarely used by that time. In Community Clouds (iv) different entities contribute with their (usually small) infrastructure to build up an aggregated private or public cloud. Smaller enterprises may benefit from such infrastructures. Finally, Special Purpose Clouds (v) provide more specialised functionalities with additional, domain-specific methods, such as the distributed document management by Google's App Engine. This group is an extension or a specialisation of the previous cloud categories.

Cloud federation refers to a mesh of cloud providers that are interconnected based on open standards to provide a universal decentralised computing environment, where everything is driven by constraints and agreements in a ubiquitous, multi-provider infrastructure. Till 2010, the cloud ecosystem has been characterised by the steadily rising of hundreds of independent and heterogeneous cloud providers, managed by private subjects, which offer various services to their clients. Following and extending the hybrid cloud definition of the European Commission, cloud providers offering Platform as a Service (PaaS) solutions may form sub-federations simultaneously to these approaches. Specific service-based applications may be more suitable for these provisions, and projects like Reservoir and 4CaaSt started to work towards supporting them. Our considered federation-oriented works target IaaS-type providers, e.g., RackSpace, the infrastructure services of Amazon EC2, and providers using cloud middleware such as OpenNebula or Eucaliptus, by forming a kind of multi-cloud architecture managed by a higher level solution.

We named our approach as Federated Cloud Management (FCM) [93], where interoperability is achieved by high-level brokering instead of bilateral resource renting. Albeit this does not mean that different IaaS providers may not share or rent resources, but if they do so, it is transparent to their higher level management. Such a federation can be enabled without applying an additional software stack to provide low-level management interfaces. The logic of federated management is moved to higher levels, and there is no need to adapt interoperability standards by participating infrastructure providers, which is usually a restriction that some industrial providers are reluctant to undertake.

Not only does the interchangeability of user applications in different clouds participating in a federation represent an open issue as a whole, but it also has several related interoperability problems concerning the management of such a large distributed ecosystem. As I mentioned before, the European Commission has assigned an expert group to publish reports on future research challenges of clouds. In these reports, they also performed a gap analysis of already existing commercial and academic solutions, and highlighted certain topics that need further research. By addressing many of these concerns, we identified four important research fields that must be taken into account in building and operating cloud federations. These topics represent different facets of interoperability: (i) enhanced monitoring solutions are needed to enable optimised management of participating providers; (ii) legal regulations need to be considered during multi-tenant data processing; and (iii) energy efficient resource management have to be enabled for future ecosystems.

The research results we achieved by addressing these topics are summarised in this chapter. In Section 1.1 I summarise our research results in proposing a monitoring approach for a cloud federation, and in Section 1.2 I introduce the legal aspects of data management in clouds and beyond. In Section 1.3 I summarise our proposal for energy efficient datacenter management, and in Section 1.4 I present our results in analysing service quality in these systems. Finally, Section 1.5 presents solutions for exploiting Personal Clouds within federation scenarios.

1.1 Monitoring in Federated Clouds

An efficient cloud selection in a federated environment requires a cloud monitoring subsystem that determines the actual status of available IaaS systems. Since there is only limited monitoring information available for the users or higher-level managers in these clouds, there is a need for a sophisticated service monitoring approach to evaluate basic cloud reliability status, and to perform seamless service provisioning over multiple cloud providers in an interoperable way. We exemplify such an extension to a federation with our Federated Cloud Management solution, where we applied a web service monitoring approach to gather additional and more detailed service quality information from the participating cloud.

The FCM approach uses the Generic Meta-Broker Service as the entry point for the users of the cloud federation. This service selects the most suitable cloud provider to perform the user's service requests by investigating the current state of the participating clouds according to the information stored in a generic service registry and the reliability metrics collected by the integrated SALMon service monitoring framework – as shown in Figure 1.1 [98]. The participating clouds are managed by cloud brokers that are capable of handling service requests and managing virtual machines within single IaaS cloud systems.

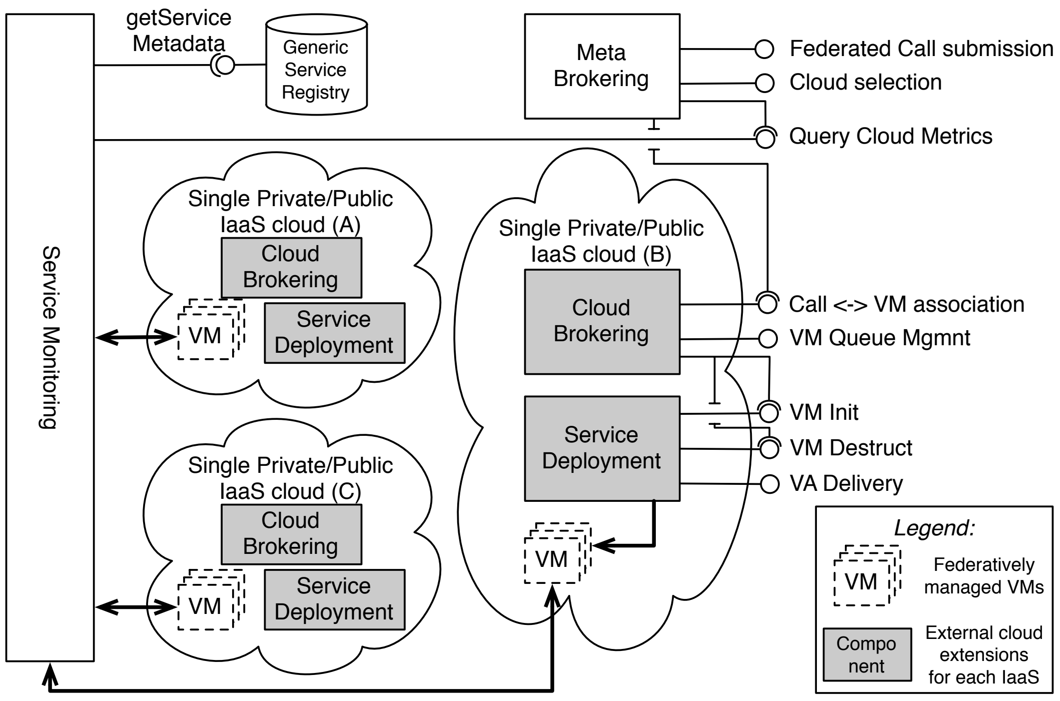


Figure 1.1: *Enhanced monitoring solution for FCM*

To enable the meta-brokering service to differentiate between cloud providers, we proposed to use a basic service that is used to efficiently determine the important characteristics of the virtual machines (VMs) available in the federation [98]. As a result, the system is capable of evaluating and choosing between public and private clouds based on the same kind of metrics. We refer to this basic service as the Minimal Metric Monitoring Service (M3S), which is capable of measuring the reliability of the infrastructure together with the integrated SALMon framework in public and private clouds. The M3S service is prepared to run on a virtual machine and offers three methods to evaluate the basic capabilities of its hosting VM. SALMon uses the response times of the method calls to express the reliability of the particular cloud that runs the M3S VM: it has (i) a generalised ping test to check the availability of the service; a (ii) CPU analyser method that performs several mathematical calculations in a large loop over a predefined set of variables, consisting of integer and floating point numbers in order to determine the computational capability of a given VM; and finally (iii) bandwidth analyser methods, which are used to compute the download and upload transfer speed of the system to determine its inbound and outbound data transfer capabilities.

Our investigations showed that both service reliability and responsiveness do vary over time and load conditions, and these measures can be used by our federated cloud management solution to select better execution environments to achieve a higher level of user satisfaction.

1.2 Data Protection in cloud federations

Cloud Computing (CC) allows the outsourcing of computational power, data storage, and other capabilities to a remote third-party. In the supply of any goods and services, the law gives certain rights that protect the consumer and provider, which also applies for CC: it is subject to legal requirements and constraints to ensure cloud services are accurately described and provided to customers with guarantees on quality and fitness-for-purpose. To exemplify legal issues arising from data management in cloud federations, we chose to perform an evaluation using requirements from data protection law. Data protection legislation is fundamental to CC as the consumer loses a degree of control over personal artifacts, when they are submitted to the provider for storage and possible processing. To protect the consumer against misusing their data, data processing legislation has been developed to ensure that the fundamental right to privacy is maintained. However, the distributed nature of CC (where cloud services are available from anywhere in the world) makes it difficult to analyse every country's data protection laws for common cloud architecture evaluation criteria. In 2012, at the time of performing this research, we chose the European Data Protection Directive (DPD) as the common directive that applies as widely as possible for our investigations. Although it is a directive of the European Union (EU), countries that want to collaborate in data transactions with EU Member States are required to provide an adequate level of protection. The requirements of the DPD are expressed as two technology-neutral actors that have certain responsibilities that must be carried out in order to fulfil the directive. These roles are the data controller and data processor, where a data controller is the natural or legal person, who determines the means of the processing of personal data, whilst a data processor is a natural or legal person, who processes data on behalf of the controller. However, following these definitions, a special case arises: if the processing entity plays a role in determining the purposes or means of processing, it is a controller rather than a processor.

We explored cloud federations through a series of use cases to demonstrate where legal issues can arise [97, 180]. In these use cases, the relevant actors and their roles were identified and the necessary actions that should be taken in order to prevent violations of the directive. We identified that there are complications when personal data is transferred to multiple jurisdictions. For example, considering a service provider (SP) located in the European Union offers services provisioned in a cloud federation, which utilises different infrastructure providers (IPs, usually operating private clouds), and one of which (IP2) is located in a non-Member State, we arrived to the following conclusion: since SP is the data controller and the participating IPs are processors, the law of the SP's Member State has to be applied, and IP2 has to provide at least the same level of protection as the national law of SP. Otherwise, if IP2 cannot ensure an adequate level of protection, the decision-making process should rule out IP2 from provider selection during data management. As a result of our investigation, we could state that service providers are mainly responsible for complying with the data protection regulation, and when personal data is transferred to multiple jurisdictions, it is crucial to properly identify the controller since this role may change dynamically in specific actions.

To react to the new situation introduced by technological improvements, we extended the scope of our investigation from cloud federations to systems exploiting Fog Computing (FC), the Internet of Things (IoT) and Artificial Intelligence, in an additional work in 2020 [181]. Since 2018, the General Data Protection Regulation (GDPR) must be applied as the general legal framework of the EU for personal data protection, and the novelty that this regulation introduced had to be considered as well. The main objectives of GDPR were to modernise the EU legal system for the protection of personal data to respond to the use of new technologies, and to revise and strengthen the previously introduced data processor and controller roles, as well as the influence of the users on processing their personal data. We identified various use cases in IoT-Fog-Cloud environments, and used them to exemplify dynamic role changing in terms of data processors and controllers. Our research results concluded that as we broaden the scope and complexity of the managed systems, the user control of the sensed private data weakens, and the responsibility of data protection are shifting towards fog, cloud, and service providers. Organisations must employ data mapping to be aware of their data flows (where the data flows from, within, and to). This would not only help identify the interaction of data between different stakeholders, but would also make a proper assessment of the privacy risks related to the storage, processing and transmission of data. Classification and data mapping are necessary to support data portability, right of access, and right of erasure.

1.3 Energy Efficient Cloud Datacenter Management

The Cloud Computing technology has created the illusion of infinite resources towards consumers, however, this vision raises severe issues with energy consumption: the higher levels of quality and availability require irrational energy expenditures. The consumed energy of resources spent for idling represents a considerable amount; therefore, the current trends are claimed to be clearly unsustainable with respect to resource utilisation, CO2 footprint, and overall energy efficiency. It is anticipated that further growth is objected by energy consumption, and the competitiveness of companies will be strongly related to these issues. Energy consumption is a major component of the operating costs of cloud datacenters.

In order to exemplify how energy consumption and CO2 emissions could be addressed in cloud federations, we introduced enhancements to our proposed Federated Cloud Management solution summarised in the previous section. At the meta-brokering layer, relying on an enhanced monitoring system within the federation, service executions can be directed to datacenters of providers consuming less energy, having higher CO2 emission quota, or have produced less amount of CO2 that expected within some timeframe. At the cloud brokering layer, if the energy consumption parameters of a cloud suddenly change, there should be strategies to limit or move around calls and even VMs federation-wise. In 2016 in [101], we targeted the latter, cloud-brokering layer, and we focused on the energy-aware management of datacenters of single cloud providers specialised in provisioning task-based cloud applications.

In order to enable experimentation in this field, we have developed a CloudSim-based simulation environment. For the evaluations, we used the log files of real VM utilisation provided by Prezi Inc. The power consumption model of the physical hosts in the simulator was based on a benchmark result provided by SPEC. To cope with the high uncertainty and unpredictable load present in these heterogeneous, virtualized large-scale systems, we applied Pliant system-based approaches to the management of these systems, which is similar to a fuzzy system. Our proposed Pliant algorithms calculated a score using the cloud's properties for each task (i.e. cloudlet in CloudSim [48]) to be scheduled on a VM. The calculation step included a normalisation step, where we applied a special Sigmoid function. For example, if the power consumption counter is high, the normalisation algorithm should give a value close to zero. One of our applied algorithms considered time and the other energy for optimisation. There were hosts in the simulated datacenters, and each host could run several VMs. Such an environment could be described with three properties: power usage counter (PUC) – which specifies the percentage of the CPU usages at a given time; power consumption counter (PCC) – the energy usage of the given host at a given time; and the number of processors of a host (PROC).

To evaluate the performance of our Pliant-based algorithms, in the first round of experiments, we created three initial strategies: the first one uses only one VM to execute all submitted jobs (referred to as MINIMUM), the second deploys a new VM for all jobs (MAXIMUM), and the third uses randomised VM selection from the available VMs (smartly prioritising the less loaded ones), and deploys a new one, if no free VM is found (SMARTRANDOM). Based on the results of these artificial strategies, we have created a Pliant-based strategy, called PLIANTDEFAULT, focusing on execution time reduction with some energy savings. After examining their results, we modified the normalisation parameters of the applied Pliant system and created more focused algorithms. We changed the sharpness of the Sigmoid function, in order to emphasise the importance of execution time. We tried several combination of normalisation parameters to achieve our goal. Our revised algorithms used a Pliant version that is more focused on execution time savings (PLIANTTIME), while in PLIANTENERGY we modified a Pliant parameter to focus on energy savings. The comparison diagram in Figure 1.2 shows the overall results [101]. Here, we can see that significant savings can be achieved in energy consumption with our proposed Pliant-based algorithms, and by fine-tuning the parameters of the proposed Pliant strategy, a beneficial trade-off can be set between energy consumption and execution time.

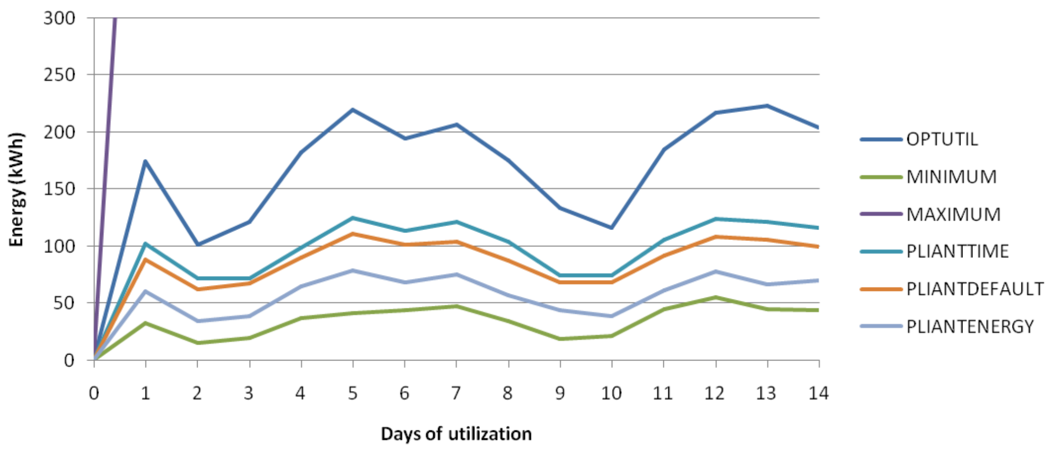


Figure 1.2: *Evaluation results of energy-aware Pliant algorithms for datacenter management*

1.4 Service Quality in Cloud Federation

Services must be described and understood both in terms of functional capabilities and service quality properties. Service Quality is a combination of several qualities or properties (e.g., availability, security, response time) of a service, and can be generally seen as an important factor in distinguishing the success of service providers. The service quality description is the main driver in selecting the best service among a set of functionally equivalent ones. Furthermore, quality is used to define a contract between a service provider and a service user in order to ensure that their expectations are met. In addition, such a contract feeds the service management system that is in charge of assessing the proper quality level during the service execution, enforcing it by taking appropriate adaptation actions, such as increasing the underlying service resources, substituting or recomposing the faulty service, and determining which settlement actions apply based on the way the service was executed, such as the final cost or penalties to be paid by the service requester or provider, respectively, and negotiations for contract termination. In 2013, in a survey on state-of-the-art service quality [106] we introduced and compared various Service Quality Models (SQM) and Metamodels (SQMM) to reveal categories of service quality attributes in sophisticated taxonomies that contain many categories and attribute types. Based on this analysis, we concluded that there is a need for a new language able to express Service Level Agreements (SLA) in a satisfactory way. This language should satisfy all the criteria of all the SLA life-cycle management activities and be capable of explicitly defining service levels with their respective Service Level Objectives (SLO), and appropriate settlement actions when the service levels are violated or surpassed.

The newly emerging demands of users and researchers that appeared around 2012, called for expanding service models with business-oriented utilisation (agreement handling), and support for human-provided and computation-intensive services. Though Grid Computing succeeded in establishing production grids serving various user communities, and both grids and Service-Based Applications (SBA) already provide solutions for executing complex user tasks, they were still lacking non-functional guarantees. Providing guarantees in the form of SLA received great attention in Grid Computing, but they have failed to be commercialised and adapted for the business world.

Its successor, Cloud Computing provided a novel infrastructure that focused on commercial resource provisioning and virtualisation. These infrastructures were also represented by services that were not only used but also installed, deployed, or replicated with the help of virtualisation. These services can appear in complex business processes, further complicating the implementation of SLAs. For example, due to changing components, workload and external conditions, hardware and software failures, already established SLAs may be violated. Frequent user interactions with the system during SLA negotiation and service executions (which are usually necessary in case of failures), might turn out to be an obstacle for the success of Cloud Computing. Thus, there is demand for the development of SLA-aware cloud middleware, and application of appropriate strategies for autonomic SLA attainment.

Despite business-orientation, the applicability of SLAs in the cloud has not gone so smoothly. By 2012, most of the existing work addressed provisioning of SLA guarantees to the consumer, and not necessarily the SLA-based management of loosely coupled cloud infrastructure. In such systems, it was hard to react to unpredictable changes, and to localise where the failures happened exactly, what the reason was for the failure and which reaction should have been taken to solve the problem. Last but not least, such systems were implemented in a proprietary way, making it almost impossible to exchange the components (e.g., using another version of a resource broker).

Autonomic Computing is one of the candidate technologies for the implementation of SLA attainment strategies. Autonomic systems require high-level guidance from humans and decide, which steps need to be done to keep the system stable. Such systems continuously adapt to changing environmental conditions. Similar to biological systems (e.g. human body), autonomic systems maintain their state and adjust operations considering their changing environment. Usually, autonomic systems comprise one or more managed elements, e.g. Quality of Service (QoS) elements.

In a research building on Autonomic Computing principles, we proposed a novel holistic architecture that considers resource provisioning using a virtualisation approach combined with business-oriented utilisation used for SLA agreement [99]. Thus, we provided an SLA-coupled infrastructure for SLA-based on-demand service provisioning. First, we gathered the requirements of a unified service architecture, then presented our solution called SLA-based Service Virtualisation (SSV) built on agreement negotiation, brokering, and service deployment combined with business-oriented utilisation. We examined this architecture and investigated how the principles of Autonomic Computing appear in the basic components of the architecture, in order to cope with changing user requirements and on-demand failure handling.

An important characteristic of an autonomic system is an intelligent closed loop of control. It is able to sense state changes of the managed resources, and to invoke appropriate set of actions to maintain some desired system state. Typically, control loops are implemented as MAPE (monitoring, analysis, planning, and execution) functions. The monitor collects state information and prepares it for analysis. If deviations to the desired state are discovered during the analysis, the planner elaborates change plans, which are passed to the executor. For the successful implementation of autonomic principles to loosely coupled SLA-based distributed system management, the failure source should be identified based on violated SLAs, and firmly located considering different components of the heterogeneous middleware components (virtualisation, brokering, negotiation, etc.). Thus, once the failure is identified, Service Level Objectives (SLOs) can be used as a guideline for autonomic reactions. We examined how the self-management and autonomous capabilities of the SSV architecture were used in representative use cases in the three SSV layers. At the highest layer, which is the meta-negotiator (MN) part of SSV, we proposed novel concepts for automatic bootstrapping between different protocols and contract formats, increasing the number of services a consumer may negotiate with. Consequently, the full potential of publicly available services could be exploited. Autonomic behaviour is also needed by brokers to survive failures of lower-level components, thus restoring healthy state after such failures. To overcome some of these difficulties, in the second, meta-brokering (MB) layer of SSV, brokers use the help of the Autonomic Service Deployment (ASD) component to re-deploy some services. Finally, in the third layer, the ASD component needs to cope with the ever-varying demand of services in SBAs. Service instances should form autonomous groups based on locality and on the service interfaces they offer (e.g., neighbouring instances offering the same interface should belong to the same group). Using peer-to-peer mechanisms, the group can decide to locally increase the processing power of a given service. This decision could involve reconfiguring an underperforming service instance in the group to allow heavier loads, or it could also introduce the deployment of a new instance to handle the increased needs.

In order to address the challenges arising from providing an enhanced level of quality in service provisioning in federated cloud environments, we designed a set of novel technologies for efficient operation of distributed VM repository management in [104]. This work was carried out in 2016 in the frame of the ENTICE EU H2020 project that aimed to develop distinctive software modules to support the following operations: (i) simplified creation of light weight and highly optimised VM images (VMI) tuned for specific application requirements; (ii) VMI repository optimisation based on a multi-objective approach; and (iii) efficient reasoning mechanism to streamline support of complex VMI operations. The introduced VMI management techniques were implemented, integrated, and evaluated in the ENTICE environment. We mostly researched image size reduction techniques, and designed a VMI synthesis tool that is able to modify the original images using two approaches. The first direct method was to alter some of the image files (e.g. by purging unnecessary packages that were irrelevant for the provided services). While the second, indirect method was to create alternative recipes that lead to more compact images upon image assembly. The evaluation results, detailed in [104], showed that our proposed VMI synthesis and analysis technique could reduce the size of the VMIs by up to 55%, while trimming the image creation time by 66%.

1.5 Federating Personal and Infrastructure Clouds

Besides our research investigations on the interoperation of purely cloud infrastructure providers, I also focused on interoperating computational and data services in clouds. In 2015, I proposed new approaches to allow the management and processing of user data produced by mobile devices in different clouds transparently, and to share these data among cloud providers in an autonomous way [94]. Although the computing capacity of mobile devices rapidly increased by that time, there were still numerous applications that could not be solved with them in reasonable time. The goal of my research team was to utilise cloud infrastructure services by executing such applications with mobile data stored in Personal Clouds. The basic concept of our solution was the following: data management services should run in one or more IaaS systems that keep tracking the cloud storage of a user, and execute data manipulation processes when new files appear in the storage – see Figure 1.3 [94].

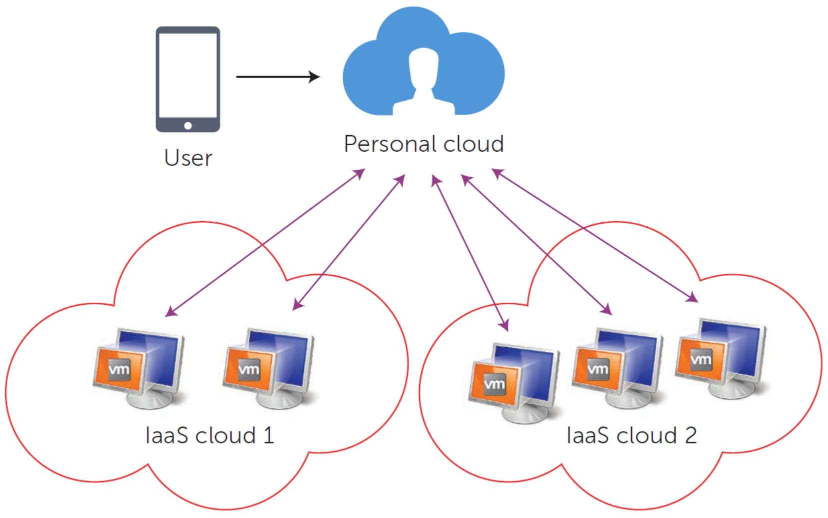


Figure 1.3: *Enhancing data management of mobile devices by interoperating clouds*

We have identified real world scenarios that require interoperable data management among cloud infrastructures to manage user data produced by mobile devices. In general, services running in IaaS clouds can download user data files from the cloud storage, execute the necessary application on these files, and upload the modified data to the storage service. Such files can be for example a photo or video made by the user with his/her mobile phone to be processed by an application unsuitable for mobile devices. In our solution developed for Android devices, there is a possibility to configure the processes to be performed on the data with a separate configuration file, which is automatically created and managed by a mobile application running on the user's device. This application is also responsible for communicating with cloud storage, which was Dropbox in our case. The file manipulation applications have been created as a virtual appliance, and have been pre-deployed in the participating IaaS clouds, before evaluating them.

In order to exemplify the usability of this generic approach, we have developed an

Android application, which can be used to manipulate pictures produced by mobile devices. This program creates thumbnails of each image of the appropriate folder then ensembles them into a single image that represents the folder and gives an overview of its contents to the user. This app can be really useful by providing a glimpse of a directory, when a user has thousands of pictures spread over numerous directories, and he/she is looking for a specific one. We conducted a performance evaluation with an academic IaaS solution called the SZTAKI Cloud, where we deployed the ImageConverter VA in two different types of virtual images, meanwhile we have also tested the Android application on two different mobile devices: on a phone (Samsung Galaxy Mini) and a tablet (Asus Slider). The results clearly showed the following differences: the local execution on the Android devices was significantly slower (more than 100 times), than the image generations performed in the cloud. These measurements fulfilled our expectations; it is worth both in terms of computation time and energy efficiency to move computation-intensive tasks to clouds from mobile devices.

To increase heterogeneity, and to show a scenario when academic and commercial IaaS clouds are interoperated through a Personal Cloud, we created another evaluation by using Dropbox, OpenNebula and Amazon. For this scenario, we ported a biochemical application to this environment that generates conformers by unconstrained molecular dynamics at high temperature to overcome the conformational bias, then finishes each conformer by simulated annealing and energy minimisation to obtain reliable structures. The end users of this app are biologists or chemists, who need to examine molecular modelling for QSAR studies for drug development, and the execution of the whole application on a single PC takes about 5-8 days [100].

By using our approach, users only need to make available their data in a Personal Cloud, and to specify with a configuration file the order of data processing (by linking VM methods to data). Once this configuration file is available, and at least one VM (executing the necessary service for processing user data) is running in an IaaS Cloud, the autonomous data processing starts and goes on till all data are processed. We used the former SZTAKI Cloud and Amazon to deploy the VMs of our biochemical application, while Dropbox was hosting the application data. With this solution, we managed to reduce the execution time to less than a day.

Summary of Chapter 1.

Contributions:

- *I designed and proposed a federated cloud management approach, and an integrated monitoring solution for Cloud Federations;*
- *I proposed a classification of data protection issues in Cloud Federations and in IoT-Fog-Cloud systems for identifying legal responsibility;*
- *I designed and proposed methods for energy-efficient datacenter management in a Cloud Federation;*
- *I proposed an SLA-based service virtualization approach exploiting meta-brokering in a Cloud Federation to maintain service quality;*

Project involvement:

Duration	Project name	Type	Role
2007-2013	S-Cube	EU FP7	Participant
2013-2014	Magyary Postdoc. Fellowship	TKP / TAMOP	Leader
2013-2016	CloudSME	EU FP7	Participant
2013-2017	ACROSS (IC1304)	EU COST Action	MC Member
2016-2017	Bolyai Research Scholarship	MTA postdoc.	Leader
2015-2018	ENTICE	EU H2020	Participant

Resulting publications:

No.	Title	Venue	Rank	IF
1	Characterizing cloud federation ... [93]	Springer CCN	-	-
2	Enhancing Federated Cloud Man. ... [98]	Springer JoGC	Q2	1.667
3	The necessity of legally compliant ... [180]	Elsevier CLSR	Q2	-
4	Legal Aspects of Data Protection ... [97]	Springer	-	-
5	Legal Issues of Social IoT Services ... [181]	Springer SCI	-	-
6	A pliant-based virtual machine ... [101]	Springer JoGC	Q1	2.766
7	A Survey on Service Quality Desc. [106]	ACM CSUR	D1	4.034
8	An interoperable and self-adaptive ... [99]	Elsevier FGCS	D1	2.639
9	Distributed environment for eff. ... [104]	WILEY CPE	Q2	1.167
10	Interoperable Data Management ... [94]	IEEE CC	Q3	-

Investigating IoT-Cloud Systems 2

CLOUD computing (CC) [47] has become a widespread and reliable solution over the past decade by providing scalable, virtualized data storage and computation. Over the years, we experienced an evolution in CC: the first clouds appeared in the form of a single virtualized datacenter, then broadened into a larger system of multiple, interconnected datacenters. As the next step, cloud bursting techniques were developed to share the resources of different clouds, then cloud federations were realised by interoperating formerly separate cloud systems. By overcoming the interoperability issues of public cloud providers and various middleware implementations, the process of creating and managing cloud federations was clarified and applied [46]. There are various reasons to optimise resource management in such federations: to serve more users simultaneously, to increase the quality of service, to earn higher profit from resource renting, or to reduce energy consumption or CO_2 emissions.

After solving optimisation issues in federations that address datacenter consolidation, operating costs, and energy efficiency, further research directions started using clouds to support newly emerging domains, such as the Internet of Things (IoT) [72]. An IoT system is a form of a global, dynamic network infrastructure composed of smart devices and sensors that have self-configuring capabilities [173]. These things can interact and communicate through the Internet by exchanging sensor data and can react autonomously to certain events. We can also influence them by triggering actions without direct human intervention. It is obvious that such systems are suitable for numerous application areas having different properties and requirements. According to reports and predictions in the IoT field, nowadays there are tens of billions of devices connected to the Internet [88].

Many cloud providers offer IoT-specific services nowadays, because CC can potentially serve most IoT needs, including transparent data generation, processing, and visualisation. Designing and analysing IoT-Cloud environments represents a great challenge (due to the huge number of devices to be managed at the same time), and these systems generally include a wide range of devices with heterogeneous components, supporting different data formats for their communication.

In this chapter, I summarise the research results we achieved by providing tools and methods for investigating IoT-Cloud environments. First, in Sections 2.1 I introduce related work in the field of IoT system simulation and modelling, and in Section 2.2 I summarise our research in analysing the widely used MQTT protocol in IoT systems. Finally, in Section 2.3 I introduce our proposed semi-simulation environment for IoT-Cloud systems. Our efforts have contributed to the proliferation of the integrated use of IoT-Cloud technologies, by providing means for evaluating and demonstrating the inner workings of such ecosystems.

2.1 Modelling and Simulating IoT Systems

Gubbi et al. [72] have identified that to support the IoT vision, the current computing paradigm needs to go beyond traditional mobile computing scenarios. CC has the potential to address these needs, as it is capable of hiding data generation, processing, and visualisation tasks. For this reason, there are more and more Platform-as-a-Service (PaaS) cloud providers offering IoT specific services. Some of these IoT features are unique, but every IoT PaaS provider has the basic capabilities to connect and store data from devices. Many things have to be managed at the same time and a wide range of devices and data formats are available, therefore, creating and examining such applications are not trivial. Botta et al. presented the main properties, features, and open issues of these systems in [42], while Nastic et al. [137] listed the numerous challenges of realising IoT-Cloud systems in practise. Studying these related works also served as a motivation for our research by raising the need to manage a large number of protocols and data formats by means of simulation. Due to the fierce competition that occurred in the past decade among leading cloud providers, such as Amazon, Google, IBM, and Microsoft, to attract users who require emerging technologies, such as CC and IoT, incompatible service interfaces and data description formats appeared in the field of IoT [113], causing severe interoperation and standardisation issues.

Several cloud providers have started to offer IoT specific services to ease the development of IoT-Cloud applications, but such systems are hard to realise. In a targeted literature survey we gathered 20 typical IoT use cases and categorised them with a taxonomy [142]. We found that these applications can be grouped into three broad categories, namely smart home, smart city, and smart region applications. Concerning the surveyed cases, the number of end-users was rather small-scale, and the number of devices was medium-scale. The sensor types and the context of the use cases were really diverse; this came from the nature of these systems to be everywhere by assisting our daily lives. The considered systems incorporated devices with low energy consumption to maintain a good battery lifetime. Regarding networking features, we could comprehend that the frequency and size of the generated sensor data were relatively small, but if we take into account the type of networks used, it can be seen that wireless networks are in the majority, and the bandwidth and error rate of these networks are not as good as in wired networks. We used the results of this taxonomy to build on and plan suitable tools for simulating IoT-Cloud systems. After conducting these surveys in 2018, we envisioned designing an IoT simulator capable of mimicking IoT behaviour to represent various IoT devices and sensors, and at the same time being able to connect to real cloud services. Such tools would save money and time for IoT application developers, hence no real devices would be needed for developing and evaluating the applications.

In 2019, the existing simulators for distributed systems could be classified as general network simulators, such as NetSim [10], Qualnet [8] and OMNeT++ [182]. With these tools, IoT-related processes, such as device placement or message handling can only be examined with great programming efforts. The OMNeT++ discrete event simulation environment is a popular and generic tool for simulating communication networks and distributed systems [182]. It can be used in various domains ranging from queueing network simulations to wireless and ad-hoc network simulations.

There were also specialised IoT simulators that addressed the issues raised by our work. Chernyshev et al. [55] presented a recent survey of IoT simulators and testbeds. They argued that research in this area was especially challenging and that existing simulators were specialised in the sense that they focused on a particular architectural layer. Han et al. [75] designed DPWSim, which is a simulation toolkit to support the development of service-oriented and event-driven IoT applications with secure web service capabilities. SimIoT [172] was derived from the SimIC simulation framework [171], and introduced several techniques to simulate the communication between an IoT sensor and the cloud, but was limited to computing activity models. Zeng et al. [190] proposed a tool called IOTSim, which provides simulation capabilities for big data processing in IoT systems, but their solution is limited to applications using the MapReduce model. Moschakis and Karatza [132] introduced several simulation concepts for IoT systems. First, they showed how the interaction between various cloud providers and IoT systems could be modelled in a simulation. Compared to our proposal, they are focusing on the behaviour of cloud systems that support the processing of data originating from an IoT system. Silva et al. [167] investigated the dynamic nature of IoT systems; therefore, they analysed fault behaviours and introduced a fault model for such systems. Khan et al. [103] introduced a novel infrastructure coordination technique that supports the use of larger-scale IoT systems, built on top of CloudSim [48], They provided customisation for their specific home automation scenarios, and as a result, they limited the applicability of their extensions. The iFogSim [73] solution is also an extension to the CloudSim toolkit. It can be used to simulate IoT and fog environments by measuring resource management techniques with several metrics, such as latency, network congestion, energy consumption, and cost.

Regarding device simulation, the SimpleIoTSimulator [9] is an IoT sensor simulator, which can be used to set up and evaluate test environments that have thousands of sensors and gateways, on a single computer. Its drawback was that it needed a specific, RedHat Linux environment, while our approach was more heterogeneous and focused on IoT device simulation with mobile devices, which was easier to apply. The Atomiton simulator [3] seemed to be very close to our concept by managing virtual sensors, actuators, and devices with unique behaviours that make up complex dynamic systems. Unlike our planned mobile, open solution, it was commercial and provided a web-based environment with limited documentation.

Table 2.1: *Comparison of related simulators for IoT*

Simulator	Cloud nodes /services	IoT sensors /devices	Artificially gen. sensor data	Real sensor /device data	Learning curve	GUI support
NetSim	Simulated	no	yes	no	long	yes
Qualnet	Simulated	no	yes	no	long	yes
OMNet++	Simulated	no	yes	no	long	limited
DPWSim	Simulated	yes	yes	no	long	yes
SimIoT	Simulated	yes	yes	no	long	limited
CloudSim	Simulated	yes	yes	no	long	no
IoTSim	Simulated	yes	yes	no	long	no
iFogSim	Simulated	yes	yes	no	long	yes
SimpleIoTSim.	Real	yes	yes	limited	short	yes
Atomiton	No inf.	yes	yes	no inf.	short	yes
MobIoTSim	Real	yes	yes	yes	short	yes

A comparison of the above-mentioned simulators is presented in Table 2.1 [142]. We used six categories to compare the main properties of these solutions. The ”Cloud nodes / services” column uses the value ”Simulated”, if the application can only simulate cloud resources in a certain use case, or ”Real”, if the system can send data to a real application in the cloud, thus using a semi-simulated environment. The option ”No inf.” means that we could not find any information about this feature. The ”IoT sensors / devices” category has the ”yes” value, when the actual tool supports the simulation of IoT devices. The ”Artificially gen. sensor data” column denotes ”yes”, if the simulator can generate sensor data in an artificial way (e.g. using certain algorithms and deviation), and the ”Real sensor / device data” provides the information about whether or not a real life IoT device data can be used in the simulation. The ”Learning curve” can be long, if the learning of the usage of the simulator and setting up a simulation environment takes a longer time, and short, if the simulator can be used almost immediately after installation, in a self-explanatory way. The GUI support feature has the value ”no”, if there is no graphical interface provided, ”limited”, if the application requires code writing and editing for creating a visualised interface, and ”yes”, if it provides graphical interaction and configuration by default. We can see from this comparison that our proposed tool (i.e. MobIoTSim, to be introduced later in Section 2.3) can bring novelty to the state-of-the-art with a graphical configuration and execution interface, offering a unique way for simulating real IoT sensor operation through loading trace data. Due to its mobile design, simulations could also be performed in wireless locations, composing a semi-simulated environment.

2.2 Analysis of MQTT Brokers for IoT Systems

From the literature review presented in the previous section we can see that a semi-simulation tool we envisioned to support IoT application developers was missing. The next step of our design process was to select a suitable communication protocol to send sensor data from our simulator to real cloud services.

IoT networks deploy several radio technologies like RFID, WLAN (IEEE 802.11), WPAN (IEEE 802.15) and WMAN (IEEE 802.16), etc. for communications at the lower level. Lower-level communication protocols may include LoRaWAN, SigFox in long-range(km)-low data rate (bps-kbps), Cellular/4G/5G in long-range(km)-high data rate (Mbps), Zigbee, Zwave in medium range(m)-medium-data rate (kbps), WiFi in medium range(m)-high-data rate (Mbps), and NFC short-range(cm)-medium data rate (kbps) category. No matter which radio technology is used to deploy an IoT network, all independent data generation by IoT devices must make their data available through the Internet for further processing, and send control information back. The performance of Machine-to-Machine (M2M) communication heavily relies on the special messaging protocols designed for M2M communication within IoT applications [17]. The web uses a single standard messaging protocol HTTP, which may be too diverse in its characteristics for IoT. In addition, there are many messaging protocols available to select from better suited for the needs of IoT systems: MQTT, CoAP, AMQP, and HTTP are the four widely accepted and emerging messaging protocols in this area [134].

To support our decision on which protocol to use for our proposed simulator, we performed a literature search for published papers in this area between 2015 and 2019 [128]. To better see how the popularity of MQTT-related research is compared to other IoT protocols, we used the following keywords: MQTT, CoAP, and AMQP. Our result showed that MQTT was the most popular choice in the field of IoT in the past five years. MQTT is an open OASIS and ISO standard (ISO/IEC PRF 20922) for client-server, publish/subscribe type messaging transport protocol [1]. The focus in the design principles of this protocol was on minimising network bandwidth and device resource requirements, ensuring reliable delivery. It is capable of transmitting data over low-bandwidth or unreliable networks with very low power consumption [134]. Characteristics like lightweight, open, simple, and easy deployment make MQTT an ideal communication protocol for constraint environments; these were the reasons why we have chosen this protocol for our proposed semi-simulator. MQTT also offers three QoS levels for message delivery (0, 1 and 2). 2 provides the highest quality and the longest transfer time, which means that messages are ensured to be delivered exactly once.

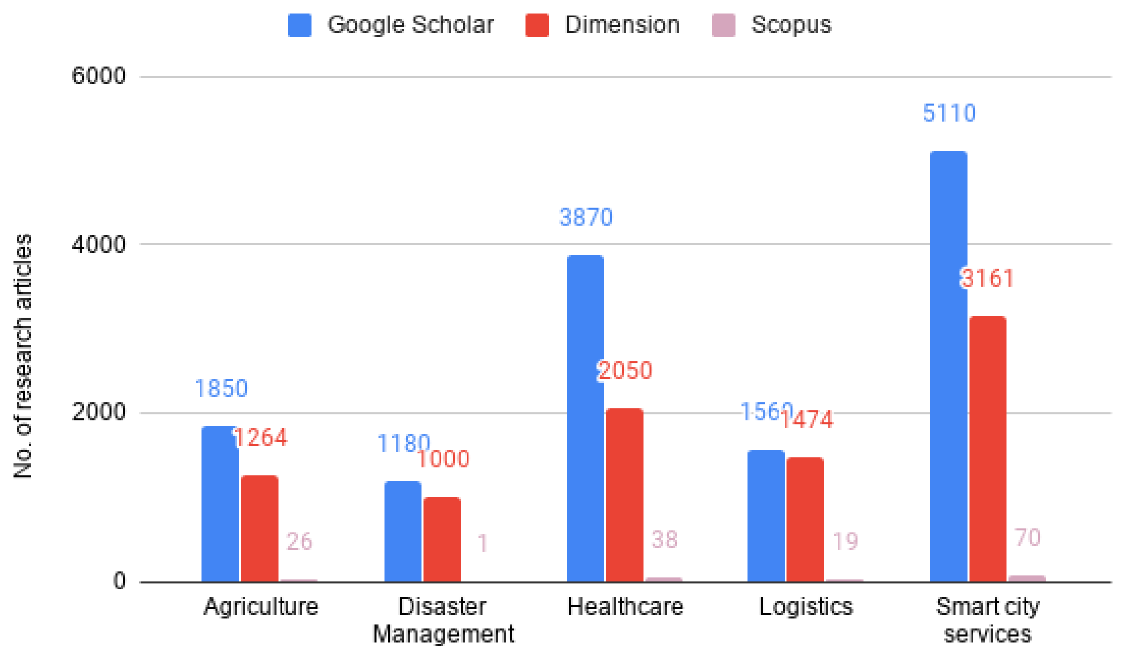


Figure 2.1: *Research in MQTT application areas (2000-2019)*

Taking into account MQTT application areas, we aggregated 20 years of research (2000-2019) on MQTT-related publication data. We performed our search in three major indexing services (Google Scholar, Dimensions, and Scopus) for the following domains: agriculture, disaster management, healthcare, logistics, and smart city services. Figure 2.1 highlights the number of papers found for each domain in the corresponding indexing databases [128]. The figures suggest that smart city services were the most popular, followed by healthcare applications.

After putting our vote for MQTT, we performed a detailed survey on the available MQTT implementations, which provide slightly different interfaces, and have different performance characteristics [128]. Next, we summarise the results of these investigations.

We found 14 different MQTT broker and 12 MQTT client library implementations, and provided a taxonomy, through which we compared and classified them by the following categories: source code availability, design and implementation, protocol features, security, and data visualisation support. Concerning the brokers, we observed that almost 50% of investigated solutions were open-source, and 50% of the tools are closed-source in nature. Almost 57.2 % of the solutions were written in C and Java, 14.3% of solutions were written in Erlang, C++, and Python, and almost 9.6% of the reviewed solutions were written in JavaScript and Go. Almost all solutions supported most QoS categories and persistent connection features. Of all the reviewed broker solutions, up to 61.53% supported the shared subscription feature, 57.14% of the solutions adopted MQTT 5.0. All the brokers had enabled security features through authentication, MQTT over TSL/SSL, and WS/WSS. Around 64.28% of the tools supported cloud hosting, 85.71% of the solutions had Docker container availability, and 92.3% had UI and dashboard. Concerning MQTT client libraries, most of the surveyed solutions were open-source in nature. After studying their implementation details and slightly different features, it was hard to choose one as the best solution, there was no clear decision to be the most suitable for our needs. We evaluated the most promising ones, and chose the Paho MQTT client library [2] for our semi-simulator implementation due to its ease of use and Eclipse integration.

Concerning MQTT brokers, we further conducted a stress testing of six broker implementations discussed in detail in [129]. We measured their message processing rate at 100% process/system CPU use, normalised message rate at unrestricted resource (CPU) usage, and average latency, both in a single-core and multi-core processor environments. We conducted our experiments in a low-end local testing infrastructure, and in a high-end cloud-based infrastructure. The local evaluation testbed was created using an Intel NUC (NUC7i5BNB), a Toshiba Satellite B40-A laptop PC, and an Ideapad 330-15ARR laptop PC. To diminish network bottleneck issues, the devices were connected through a Gigabit Ethernet switch. The Intel NUC was configured as a server running an MQTT broker, the Ideapad 330-15ARR laptop was used as a publisher machine, and the Toshiba, Satellite B40-A was used as a subscriber machine. The Ideapad 330-15ARR (publisher machine), with 8 hardware threads, was capable enough of firing messages at higher rates. The cloud-based evaluation testbed was configured on the Google Cloud Platform (GCP) [71]. We created three c2-standard-8 virtual machine (VM) instances that have 8 vCPUs, 32 GB of memory, and 30 GB local SSD each to act as publisher, subscriber, and server, respectively. All the VM instances were placed within a Virtual Private Cloud (VPC) Network subnet using Google's high-performing premium-tier network service.

To perform our experiments, we developed a Paho MQTT library-based benchmarking tool called MQTT Blaster. In our measurements, we used the following brokers: Mosquitto 1.4.15, Bevywise MQTT Route 2.0, ActiveMQ 5.15.8, HiveMQ CE 2020.2, VerneMQ 1.10.2 and EMQ X 4.0.8 [129]. Out of these MQTT brokers, Mosquitto and Bevywise MQTT Route are non-scalable implementations, and the rest are scalable in nature. Mosquitto is a single-threaded implementation, and Bevywise MQTT Route uses a dual thread approach, in which the first thread acts as an initiator of the second that processes messages. During the test, we assumed that by increasing the number of subscribers or the number of topics or message rate results in an increased load on the broker.

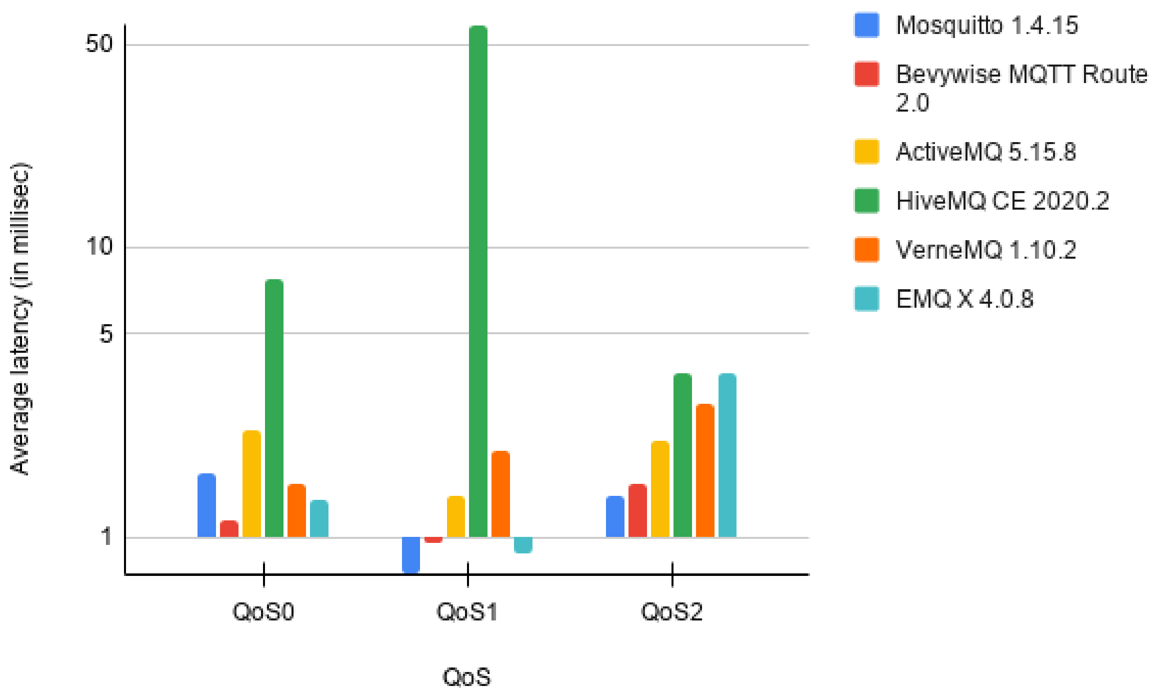


Figure 2.2: *A comparison of average latency of all scalable and non-scalable brokers in the local evaluation environment*

We used the combination of three different publishing threads (one topic per thread) and 15 subscribers, and we were able to push the broker to 100% process usage, and limited the CPU usage on publisher and subscriber machines below 70% and 80%, respectively. We considered latency as the time taken by a system to transmit a message from a publisher to a subscriber. With this experiment, we tried to simulate a realistic scenario of a client trying to publish a message, when the broker is overloaded with many messages on various topics from different clients. Concerning the message payload size setting, we used 64 bytes for the entire testing. We separated our experimental results into three categories for better interpretation and understanding: (i) projected message processing rates of non-scalable brokers at 100% process CPU usage, (ii) projected message processing rates of scalable brokers, and (iii) latency comparison of all the brokers. We performed three evaluation runs for each QoS level in each category, and considered the best result with the maximum rate of message delivery, and zero message drop for comparison. Here we selected two evaluation result comparisons to show, the rest with detailed results and discussion can be read in [129].

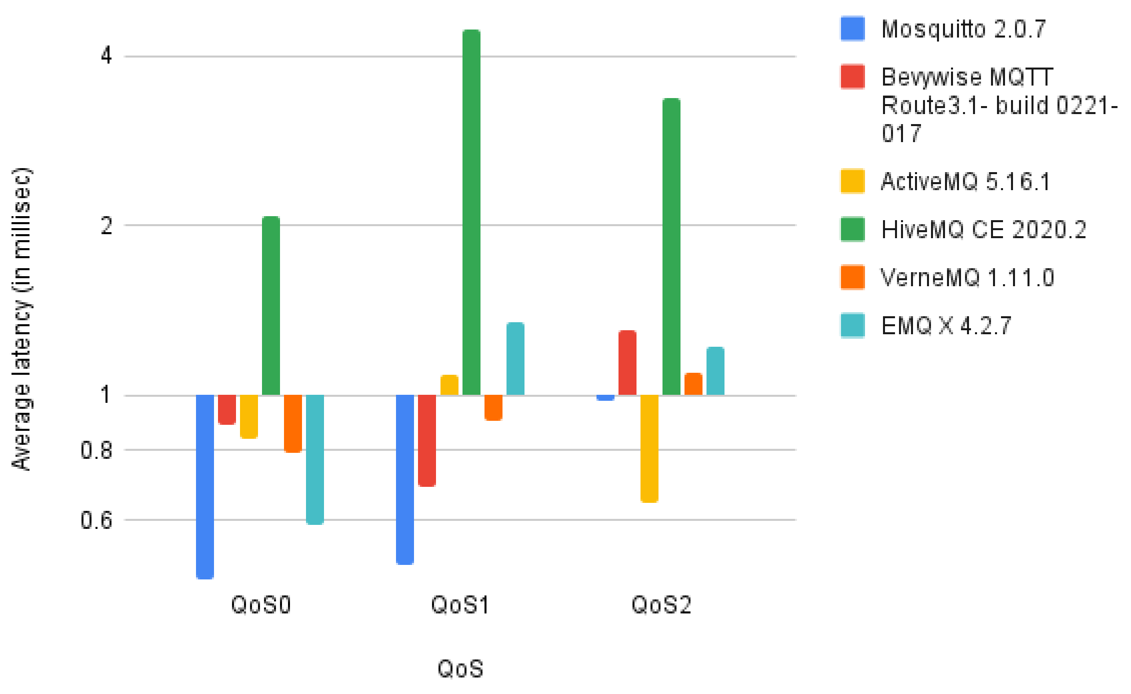


Figure 2.3: *A comparison of average latency of all scalable and non-scalable brokers in the cloud evaluation environment*

Regarding the results of the evaluation for the local infrastructure experiments, Figure 2.2 shows the results in terms of average latency (or round-trip time) [129]. Here, we found that at QoS0 Bevywise MQTT Route 2.0 performed best, while in all other QoS categories (QoS1 and QoS2), Mosquitto v. 1.4.15 had the best results. Figure 2.3 shows a side-by-side comparison of both scalable and non-scalable brokers in terms of average latency recorded, when the evaluation was performed in the cloud-based infrastructure [129]. The results showed that Mosquitto performed best at 0 and 1 QoS levels, while it came second right after ActiveMQ for QoS level 2. This detailed performance analysis strengthened our decision for choosing the Mosquitto MQTT broker to be the base for our future private cloud gateway service, besides its widespread use in related cloud provider implementations (e.g. in IBM Cloud).

2.3 Semi-simulating IoT-Cloud Environments

In our research, we did not aim to simulate all possible IoT systems and networks, but we still wanted to aid the design, development, and testing processes of these systems with a reasonable coverage. Our goal was to develop a mobile IoT device simulator that can emulate real devices and sensors, thus it can be used in the processes mentioned above instead of real resources.

Our refined requirements for basic functionalities of such a simulator were to send and receive messages, generate sensor data (for possibly multiple devices), and react to received messages. These capabilities are sufficient for a simulator to be used in IoT system analysis. Other requirements for advanced functionalities (such as simulating network errors, recording and replaying concrete simulation cases, and connecting real IoT devices to the simulator) could contribute to the analysis of more realistic systems.

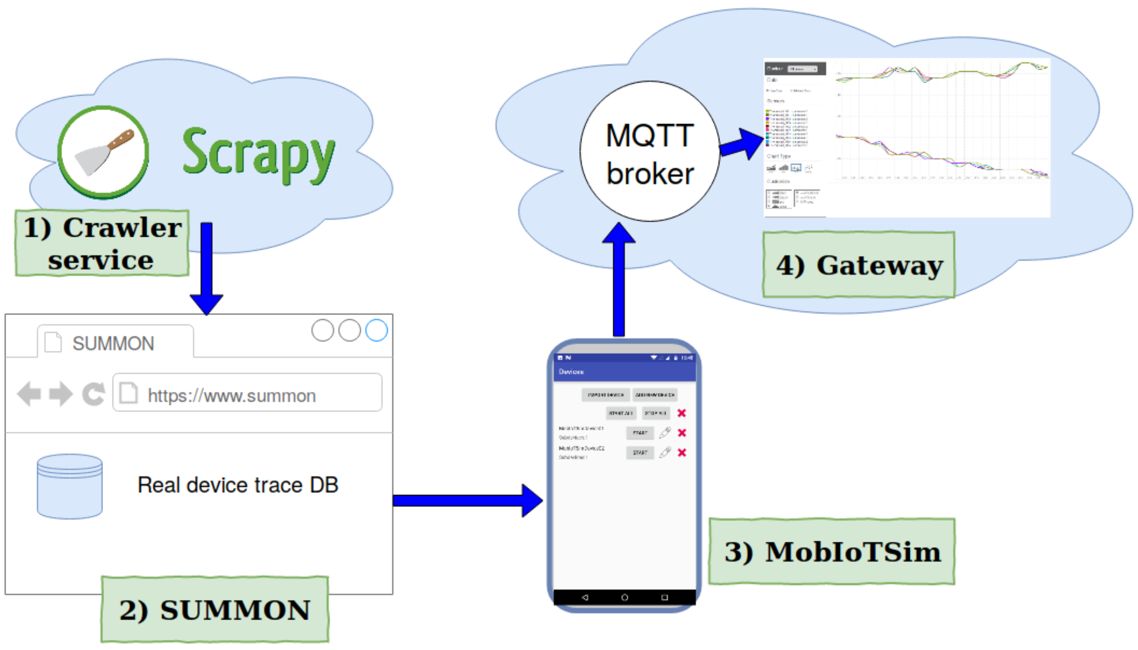
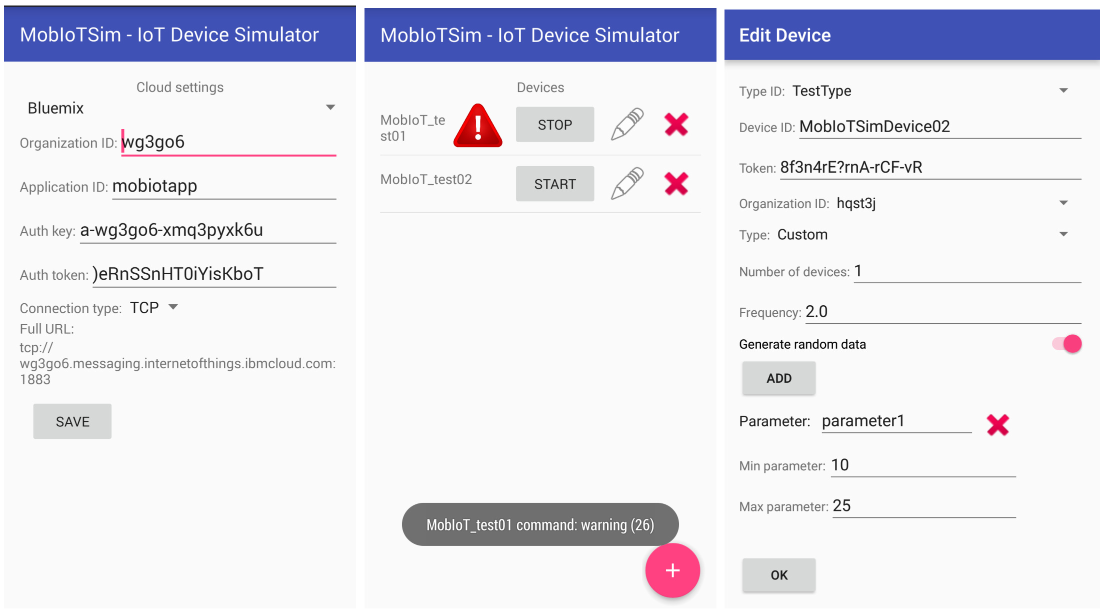


Figure 2.4: *A framework for semi-simulating IoT-Cloud systems*

First, we aimed to support the basic IoT functionalities. To this end, a simulated device should have a unique identifier, as well as tokens for registration and authentication. The generated sensor data should be made available in plain text, such as JSON, or in binary format, including metadata information, such as date, time, and device state. Lastly, the MQTT communication protocol should be supported.

By examining related simulators in the IoT field, we found that we wanted to have a solution that puts an emphasis on mobility, and provides means for performing simulations at specific geolocations. This fact served as the main motivation for designing our simulator for mobile devices. The main purpose of our proposed mobile IoT device simulator was to help cloud application developers learn IoT device handling without buying real sensors, and to test and demonstrate IoT applications that utilise multiple devices.

The architectural view of our proposal for simulating IoT-Cloud systems is depicted in Figure 2.4 [102]. Compared to traditional simulators, we proposed a semi-simulated environment to stay as close to real world systems as possible. We started to gather real sensor trace files of public IoT initiatives, and made them available through an archive (2 and 1), which can be connected to our simulator called MobIoTSim (3) to replay real device behaviour. We also developed gateway services (4) to manage IoT devices by processing and visualising sensor data. These gateways can be instantiated and operated at private or public cloud providers. Our proposed framework has been presented in detail in [102], next I summarise their main components in the following subsections.


MobIoTSim - IoT Device Simulator

Cloud settings

Bluemix

Organization ID: wg3go6

Application ID: mobiotapp

Auth key: a-wg3go6-xmq3pyxk6u

Auth token:)eRnSSnHT0iYisKboT

Connection type: TCP

Full URL: tcp://wg3go6.messaging.internetofthings.ibmcloud.com:1883

SAVE

MobIoTSim - IoT Device Simulator

Devices

MobIoT_test01 STOP

MobIoT_test02 START

MobIoT_test01 command: warning (26)

Edit Device

Type ID: TestType

Device ID: MobIoTSimDevice02

Token: 8f3n4rE?rnA-rCF-vR

Organization ID: hqst3j

Type: Custom

Number of devices: 1

Frequency: 2.0

Generate random data

ADD

Parameter: parameter1

Min parameter: 10

Max parameter: 25

OK

Figure 2.5: *MobIoTSim GUI: a.) On the left - Cloud settings, b.) In the middle - Devices, c.) On the right - Device settings*

2.3.1 MobIoTSim

Our mobile IoT device simulator was designed to simulate up to hundreds of IoT devices, and it was implemented as a mobile application for Android platforms. Sensor data generation of the simulated devices can be done through artificially generated, random values in a range given by the user by default. For more advanced simulations, data from IoT-specific trace files can also be loaded and replayed. The data sending frequency can be specified for each device for both cases. We created trace files from the OpenWeatherMap public weather data provider [7], which monitors multiple cities and stores many parameters of them, including temperature, humidity, air pressure, and wind speed. Using the trace loader feature of MobIoTSim, the simulation can mimic a real life scenario by using real world application traces. The simulator uses the MQTT protocol to send messages with the use of the Eclipse Paho library, as we discussed in the previous section. Message data is represented in a structured JSON object, which is compatible with the IBM IoT Foundation message format [6].

The basic usage of the simulator is the following: (i) connect the application to a cloud, where the data are to be sent, (ii) create and configure the devices to be simulated, and (iii) start the (data generation of the) required devices. These main steps are represented by the three main parts of the application: the *Cloud settings*, the *Devices*, and the *Device settings* screens – as shown in Figure 2.5 [102].

In the *Cloud settings* screen (Figure 2.5.a), the user can set the required information about the target cloud, where the data will be received and processed. Default connections are predefined for IBM Cloud (formerly Bluemix) and Microsoft Azure public cloud providers. If a simulated device wants to send data such as an IoT service, it has to be registered beforehand. The registered devices have device IDs and tokens for authentication. The MobIoTSim application handles the device registration to the cloud with REST API calls, so the user does not have to register the devices manually on the graphical web interface (which is also possible). The main part of an IoT service is an MQTT broker, where device messages are received, and are possibly forwarded to other cloud services. Such cloud services or applications can process the data, react to it or just perform some visualisation tasks. The required configuration parameters for the default providers are preset in the simulator (and have to be updated in the application for each user registration at the utilised providers to allow proper authentication).

In the *Devices* screen (Figure 2.5.b), we can create or import devices, and we can also view the list of existing devices, where every row is a device or a device group. These entities can be started and stopped by the user at will, both together or separately for the selected ones. Some devices have the ability to display warnings and notifications coming back from the gateway, so backward communication is also supported. By clicking on the Add button we can create new devices or device groups. Predefined device templates can also be used for this purpose. These device templates help to create often used devices, such as temperature sensors, humidity sensors, or a thermostat. The import feature is applicable when we previously saved a device to a file with its sent messages for a certain simulation period.

The creation of a new device and the editing of an existing one can be done in the *Device settings* (or *Edit device*) screen (Figure 2.5.c). If the user selects an existing template for the base of the device in the *Type* field, the message content and frequency will be set to some predefined values. For example, the *Thermostat* template has a temperature parameter, and the device turns on by reaching a predefined low-level value and turns off at reaching a high-level value. The On/Off state of the device is displayed on the screen of the application all the time. It is possible to select the *Custom* template to configure a unique device with various sensor parameters in detail. By default, random values will be generated within the set ranges as parameters. A device group is a group of devices with the same base template, and they can be started and stopped together. A group can be defined by setting the value of the *Number of devices* field (which is one by default).

To evaluate the simulator, we examined the device simulation scalability. For this purpose, we created a simplified version of MobIoTSim (called MQTTDemo), containing only its core parts for device management functionality. It enables access to low-level configurations, such as specifying the number of threads used for managing the simulated devices. It also collects detailed statistical information regarding the simulation by measuring elapsed times for executing certain functions. It can be connected to our private gateway deployed in the IBM Cloud, and can send messages to it using the MQTT protocol. By the time we performed the evaluations, the number of simulated devices was limited to 20 by the IBM Cloud platform, therefore we had to obey this restriction. The evaluation was performed with a Samsung S5 device, and a Node.js visualisation gateway application running in the IBM Cloud. The Samsung S5 was released in 2014 having Snapdragon 801, Quad-core 2.5 GHz processor, 2 GB RAM and the OS is Android 4.4. The most informative measurements were made with 1 sec. message frequency, simulating 10 to 20 devices, and using 3 to 12 threads. The message contents were five randomly generated values in a JSON object (according to the Weather template). The duration of performing a test was around one minute.

The executed test results showed that the random data generalisation consumed an almost negligible amount of time, so it did not interfere with the simulation. First, we started using three threads, because Samsung S5 has four cores, but from the results we could see that the number of threads is an important factor. With 10 simulated devices, three threads had difficulties coping with simultaneous message sending, it required almost 1.5 seconds for the devices to start a message sending cycle again, instead of the one second specified in the settings. To manage 15 devices, four threads are few, but six threads would be enough to perform well. To manage 20 devices, we can see that six threads were too few for the acceptable performance, therefore, we had to use eight threads instead. We also made additional experiments to find the limits of MobIoTSim using 20 simulated devices. The minimum time required to send a message is around half a second, in general. For lower frequency (e.g. 0.25 second), there was no difference in the measured time even by using 16 threads. To conclude these experiments, we can state that our proposed simulator can send a total number of 2300 messages in a minute with 20 registered devices to the IBM Cloud.

In a work within the ACROSS EU COST Action [46], we showed how Cloud Federation can help IoT systems by providing more flexibility and scalability. Higher-level decisions can be made on where to place a gateway service to receive IoT device messages, e.g. in order to optimise resource usage costs and energy utilisation. Such complex IoT-Cloud systems can hardly be investigated in real world, and simulators can help in providing a scalable experimentation platform, such as MobIoTSim.

2.3.2 SUMMON

To analyse complex systems exploiting IoT and CC technologies, we need open datasets of real world IoT applications to enable realistic simulations. The European Commission has published a guideline document [62] for encouraging open access to reuse digital research data generated by Horizon 2020 projects. In this report, they defined the so-called FAIR data principles to help Horizon 2020 beneficiaries to make their research data findable, accessible, interoperable and reusable. Following these guidelines, we aimed to create an open IoT trace archive to facilitate data search and accessibility, and to reuse sensor data for experimenting with IoT-Cloud systems. In research works addressing distributed systems we can find data sources used for a similar purpose, e.g. in the Grid Workloads Archive [83], which contains trace files of job executions in real-world Grid systems. Our aim was to propose a way to build up a similar archive for the IoT-Cloud research community to facilitate simulation experiments.

As the first step of this research, we looked for publicly available projects of various IoT application areas. We found three smart city initiatives that met our criteria, and publish open data of IoT sensors and devices: the SmartME project [45], which is a crowdfunded system using the IoT paradigm in the municipality area of the city of Messina, Italy; the CityPulse project [38], which is a smart city data analytics framework providing various datasets with time series data and live data feeds; and finally, the Open Data Program of Surrey, Canada [11], which provided governmental datasets viewable and searchable through an online service. In [144], we analysed these projects and their sensor data sharing methods in detail.

To efficiently and easily work with several datasets located at different sources, we need a service, which gathers and filters such data and unifies their different data formats. To realise such a service, we proposed a tool called SUMMON (Smart city Usage data Management and MONitoring). It is a web application implemented in Node.js, and uses the CKAN system API. The CKAN system has a hierarchical structure, where the so-called organisations are on the top. It is possible to list all organisations, and we can even request more information through the provided API, but unfortunately the datasets (or in CKAN dialect: packages) of an organisation cannot be listed. Therefore, we used a different API call to query the datasets. We can get more detailed information about a dataset (i.e. licence, maintainer, author). SUMMON uses an API call to receive the schema of the dataset, so it can generate a form based on that. Finally, the results are displayed as a preview, embedded in an HTML code with the possibility to modify the form and see some example results, or in JSON format to be downloadable.

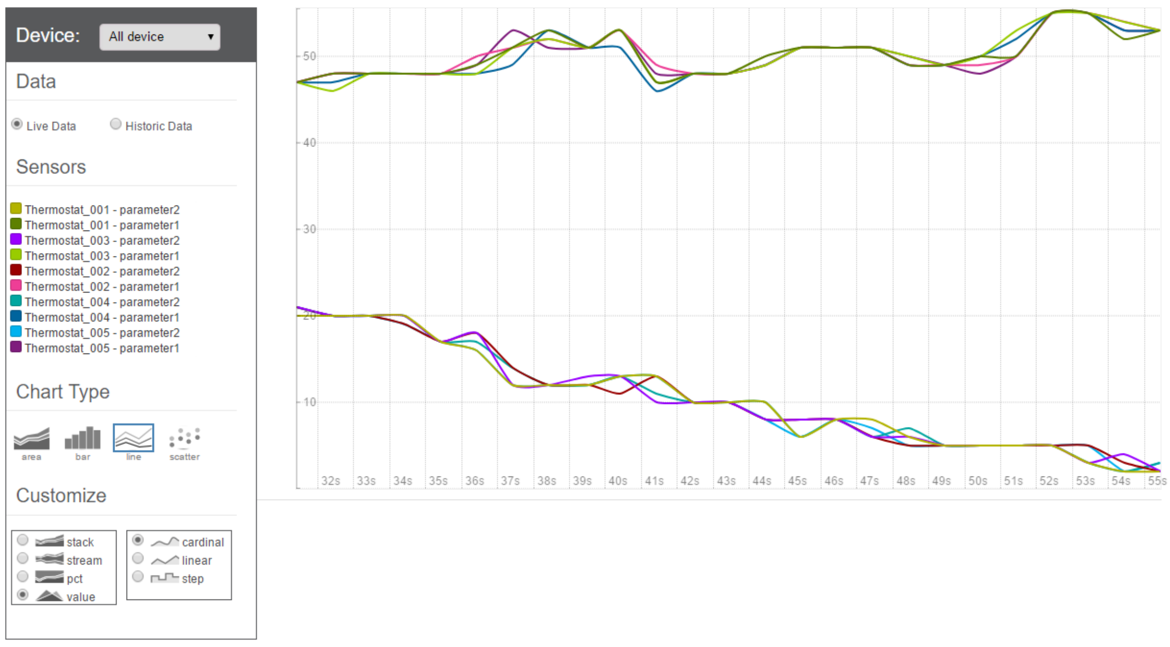
In order to demonstrate the applicability of SUMMON, we used MobIoTSim, our mobile IoT device simulator. It can be easily connected to a private gateway service (to be discussed in the next section). It is capable of receiving and processing sensor data from several devices simultaneously. It has a web-based graphical interface to visualise sensor data coming from MobIoTSim. Messages (defined in JSON format) sent by simulated devices are managed by an MQTT server. It can also be used to send responses (or notifications) back to the simulated IoT devices in MobIoTSim.

The simulation of an IoT usage scenario starts with the filtered JSON data from SUMMON imported into the MobIoTSim simulator. When the user starts a simulation with MobIoTSim, the sensor data are sent in the form of messages to the MQTT Broker of the IoT Gateway service. After receiving these real-life values, the broker forwards them to a gateway application. The gateway application shows a real-time chart with the data, even from several devices simultaneously. In cases when we need specific values from the dataset, or just a specific number or a value of records, SUMMON can generate a form, where the required criteria can be added. The downloaded JSON data from SUMMON can be imported into the MobIoTSim simulator. The process can be done similarly to create a new device, but instead of generating random data, we can select the downloaded file to read the values from it. In this way, the created simulated device will send real-life data to the cloud. Even multiple simulated devices can be added, with the same or different data source imported from SUMMON.

In order to extend the previously mentioned use cases, we designed an automated web crawling service [145] to gather and store additional historical IoT and sensor data in SUMMON. We planned to provide a way to gather and store sensor and IoT data scattered over websites of various projects. To create such a service we used Scrapy [13], which is an opensource python-based framework for data mining. We realised the service as a container-based Docker application, deployed it in the Google Cloud, and connected it to SUMMON through a REST API connection. In this way the data gathered by our crawler service can be regularly updated and stored in SUMMON. To evaluate the operation and usability of our proposed crawler service, we performed measurements for crawling such websites by executing spider jobs in certain time intervals. With our experiments, we observed that the web page layouts of the corresponding smart systems vary highly, therefore no generic spider can be used to crawl them. The results also showed the difficulty and sophistication needed to collect sensor and IoT data from the three different application areas we considered. Detailed evaluation results can be read in [145].

2.3.3 Private Gateways

We have seen our envisioned architecture for simulating IoT-Cloud systems in the beginning of this section. Next, we focus on its fourth component, and we show how to develop specialised, private cloud gateways for IoT application management. In the literature, Kang et al. [90] introduced the main types and features of IoT gateways. Compared to this work, our research did not aim to propose a generic solution for all needs of an IoT system, but to provide a gateway solution that can be used together with MobIoTSim to enable a comprehensive simulation environment for investigating IoT-Cloud systems and applications.


Device: All device

Data: [✓ Live Data] Historic Data

Sensors:
- 🟨 Thermostat_001 - parameter2
- 🟩 Thermostat_001 - parameter1
- 🟪 Thermostat_003 - parameter2
- 🟩 Thermostat_003 - parameter1
- 🟥 Thermostat_002 - parameter2
- 🟥 Thermostat_002 - parameter1
- 🟦 Thermostat_004 - parameter2
- 🟦 Thermostat_004 - parameter1
- 🟦 Thermostat_005 - parameter2
- 🟪 Thermostat_005 - parameter1

Chart Type: area, bar, [✓ line], scatter

Customize: stack, stream, pct, [✓ value]; [✓ cardinal], linear, step

Figure 2.6: *Detailed, parameter-wise data visualization in the private gateway in the IBM Cloud sent by a group of devices in MobIoTSim*

We developed our own, private gateway service (discussed in detail in [143]) that is capable of managing several devices simultaneously, and can send a notification to the MobIoTSim device simulator by responding to critical sensor values. This gateway service is an extended version of the IBM visualisation application [5], which is able to handle up to hundreds of devices at the same time. It has a web-based graphical interface to visualise sensor data coming from MobIoTSim. Messages (in JSON format) received from the simulated devices are managed by the Mosquitto MQTT broker. It can also be used to send responses (or notifications) back to the simulated IoT devices in MobIoTSim. It provides a view in which the data sent by all devices can be seen in a single, real-time chart as shown in Figure 2.6 [143]. It is a Node.js application that is able to manage the chart by subscribing to the device topics with the MQTT protocol, and waits for the messages. To enhance and better visualise the data of many devices at the same time, we introduced device grouping for chart generation. Though the IBM Cloud platform provided some monitoring information for application services, custom Docker containers can be better monitored. For this purpose, we created the Docker version of our private gateway to support portability.

It is easy to connect the MobIoTSim simulator to such a private gateway. The predefined template can be used, we only need to specify an organisation identifier and the connection type to enable connection to the MQTT server. In the case of already available templates, the necessary URL is predefined. The application ID, the authentication key and token can be retrieved by registering to the gateway service, and these parameters can be used later to sign in to the data visualisation site of the gateway. The simulated devices also need to be registered to the MQTT server of the gateway service – just like real devices –, by specifying their device and type identifiers and sensor data thresholds, which reply with their token identifiers (to be used for device setting). We can create advanced scenarios with this private gateway, such as managing more devices and responding to critical sensor data coming from the simulated devices.

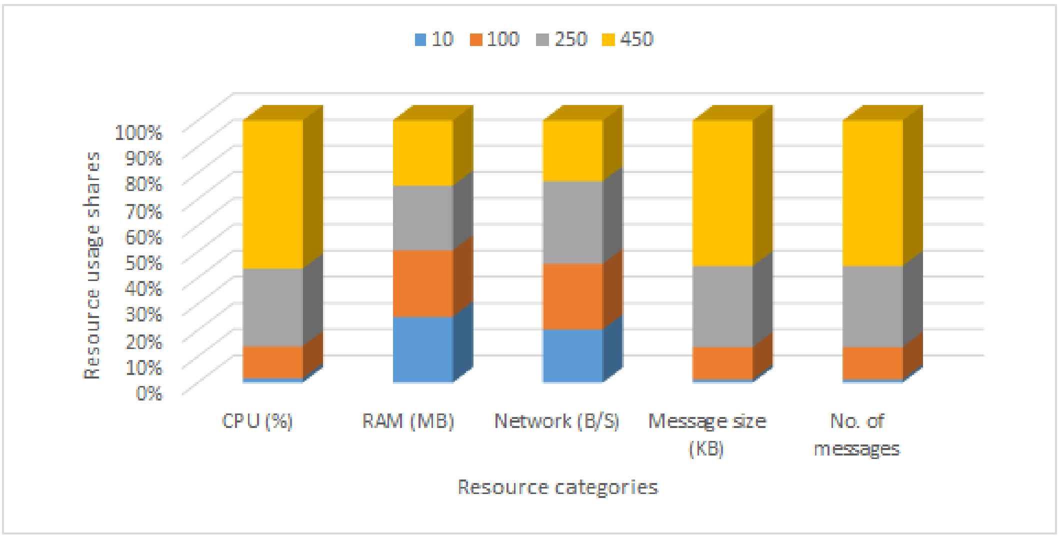


Figure 2.7: *Comparison of different device groups*

Evaluation of the private gateway

We performed various experiments to evaluate the performance of MobIoTSim used together with this private gateway. Here, we show some highlights of this evaluation. To evaluate the grouped device management feature of the gateway, we compared cases with 10, 100, 250, and 450 devices in groups. We deployed the private gateway in a Docker container as a micro application with the following parameters: OS - Ubuntu with Linux kernel 4.4.0; RAM - 256 MBs; CPU - Intel Xeon E5-2690 with 48 cores. Figure 2.7 shows the average results for the device groups and the appropriate resources [143]. It highlights how resource utilisation varies by enlarging the groups of devices to be managed by the gateway.

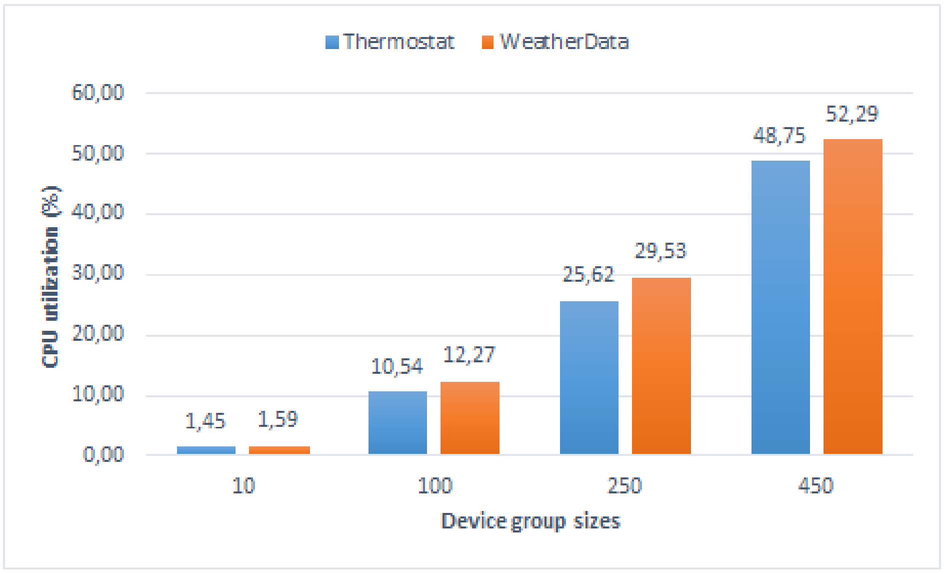


Figure 2.8: *Comparison of CPU utilisation of the two device templates*

Once we got some impression of working with a simple thermostat device, we chose a more complex one. As one of the earliest examples of sensor networks are from the field of meteorology and weather prediction, we chose to model a meteorological service and gathered real data from OpenWeatherMap. We used it with the Weather template of MobIoTSim to create device groups of 100 and 450 simulated devices. For this experiment, we randomly picked weather data of cities (one city for one simulated device) from earlier traces. We can see that as the number of devices grows, the resource utilisation also gets higher, as shown in Figure 2.8, which compares the CPU utilisation of the two device types [143].

Specialized gateway for IoLT applications

Smart farming is a rapidly growing area within smart systems that need to respond to great challenges in the near future. By 2050, it is expected that the global population will grow to 9.6 billion as the United Nations Food and Agriculture Organisation predicts. A recent Beecham Research report [4] also states that food production must respond to this growth by increasing it with 70% until 2050. Plant phenotyping [160] also evolves rapidly and provides high-throughput approaches to monitor plant growth, physiological parameters, and stress responses of plants with high spatial and temporal resolution. Recent advances use the combination of various remote sensing methods that can exploit IoT and cloud technologies. In the past, typical plant phenotyping platforms used very expensive instrumentation to monitor several hundreds, even a few thousands of plants. Although these large infrastructures are very powerful, they have high cost, ever reaching a staggering few million euros per platform, which limits their widespread, everyday use. Due to recent ICT developments, we can apply novel sensor and IoT technologies to provide a promising alternative, called affordable phenotyping.

In a research work performed together with colleagues from the Biological Research Centre [146], we proposed a low cost plant phenotyping platform for small sized plants, in the frame of the Internet of Living Things (IoLT) national project. It enables the remote monitoring of plant growth in a standard greenhouse environment. This so-called IoLT Smart Pot uses IoT and cloud technologies to monitor the effect of various stress factors of plants (drought, nutrition, salt, heavy metals, etc.), as well as the behaviour of various mutant lines.

The biologists designed a hardware platform for hosting 12 small sized plant pots (for Arabidopsis plants) organised in a 4x3 matrix. To monitor plant growth, an RGB camera and a LED-based illumination system for additional lighting are installed above the plant cluster. The relevant environmental parameters are light intensity, air and leaf temperature, relative air and soil humidity, which are monitored by sensors placed above and into the pots. To govern the monitoring processes, a Raspberry Pi board is placed beside the cluster. The monitored sensor data are stored locally on the board, and is accessible through a wired connection on the same network. The initial configuration for performing periodical monitoring was set to five minutes concerning the sensor readings, and one hour to take pictures of the cluster of pots.

To operate the smart pot cluster, we developed a specialised gateway to monitor the plants. Users can access the gateway through a web interface provided by a Node.js portal application. It can be used to group and manage pots and users with projects created by administrators. In this way, registered and connected smart pots can send sensor data to them, which can be visualised in the portal. The gateway stores regularly updated sensor values and periodically taken pictures of the smart pot cluster. The portal can be used to query, visualise, and download a set of sensor values for a certain period, and the created pictures. The portal is also capable of monitoring the growth speed of the plants, which is performed by calculating the projected leaf area visible in the taken pictures.

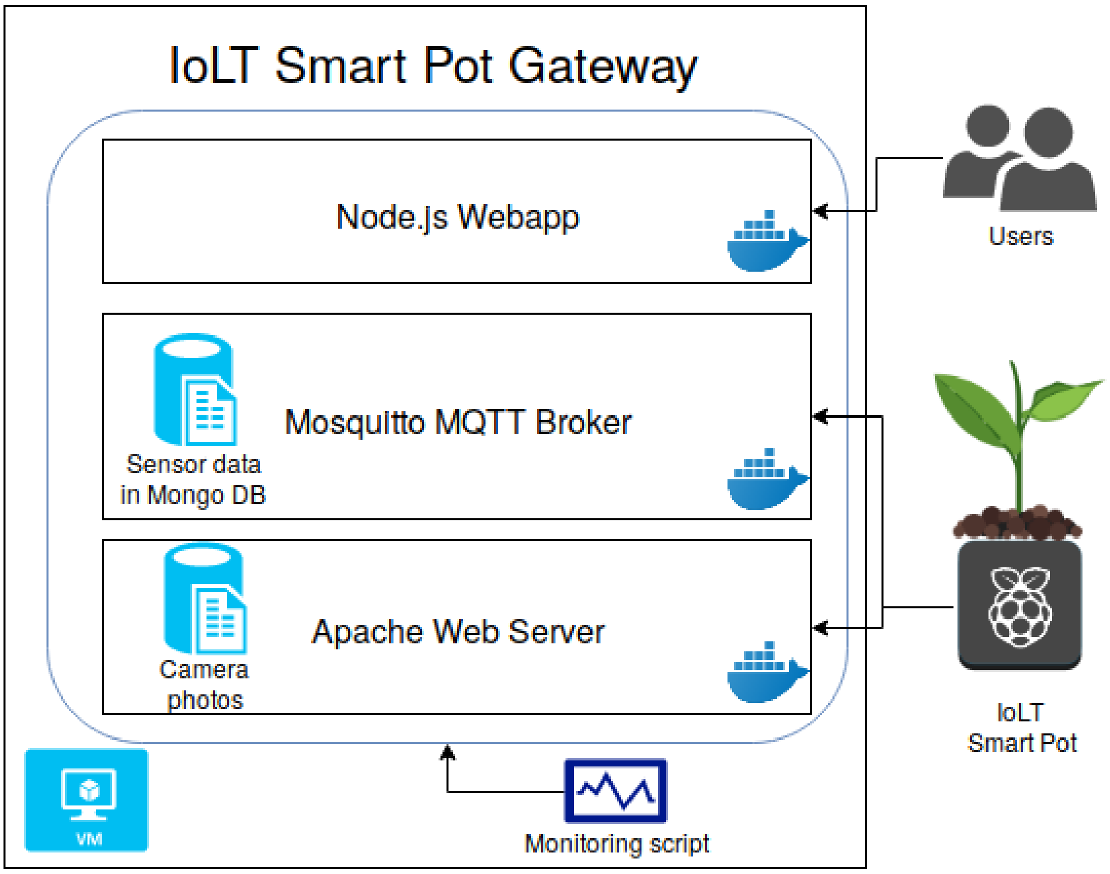


Figure 2.9: *The architecture of the IoLT Smart Pot Gateway*

The architecture of our proposed IoLT Smart Pot Gateway can be seen in Figure 2.9 [146]. It has a modular setup, it consists of three microservices, realised in Docker containers, which compose the gateway application that can be deployed to at a cloud provider. Special monitoring scripts are used to track and log the resource use of containers for performance measurements. In the middle of the architecture, we can find the Mosquitto MQTT Broker service, which is built on the open-source Mosquitto tool to store the sensor values received from the pots using a MongoDB [12] database. The seven monitored sensor types of a pot are described by a JSON document, which should be regularly updated and sent in a message by an MQTT client of a smart pot to the MQTT broker running in this service. Sensor readings on the Raspberry Pi board are performed by a Python script using an MQTT client package configured with a pot identifier, sensor value sampling frequencies, and picture taking frequencies. The third microservice in the bottom is called the Apache Web

Server, which is responsible for saving the pictures of the plants in the pots.

To evaluate the scalability of our gateway solution, we performed three simulation measurements with 50, 100, and 250 clusters (composed of 600, 1200, and 3000 pots respectively). In all cases, we performed the measurements for 30 minutes, and the simulated smart pot platform sent sensor values to our gateway. In the first simulation for 50 clusters, we set the sampling of resource usage (processor and memory usage) to every 10 seconds, while for the second and the third one (100 and 250 clusters) we set it to two seconds (to have a better resolution of resource loads). For the largest experiment, we further increased the number of pot clusters to 250 arriving to a total number of 3000 simulated pots. For this round, we performed the simulation for 30 minutes, again, with the same periods as defined for the lower-scale rounds.

To summarise our investigations, we could observe that by increasing the number of pots to be managed by the gateway service, the utilisation also increased. As expected, CPU utilisation was the highest in the last round for managing 3000 pots at the same time with almost 40 percent. The memory utilisation was also the highest in this case with almost 25 percent. Finally, we could state that these results proved that we can easily serve numerous phenotyping projects monitoring up to thousands of pots with a single gateway instance in a cloud.

Summary of Chapter 2.

Contributions:

- *Mishra and I analysed, evaluated and categorized message brokers using the MQTT protocol;*
- *I designed an approach for semi-simulating IoT-Cloud systems; Pflanzner and I proposed and evaluated MobIoTSim, a mobile IoT device simulator based on this approach;*
- *Pflanzner and I proposed and evaluated a generic and a specialized cloud gateway solutions that can efficiently manage IoT devices by receiving, storing and processing or visualizing their data;*

Project involvement:

Duration	Project name	Type	Role
2013-2017	ACROSS (IC1304)	EU COST Action	MC Member
2017-2018	UNKP-17-4	Postdoc. scholarship	Leader
2017-2021	Internet of Living Things	GINOP	WP Leader
2018-2021	Smart Systems	NKFIH	RG Leader
2019-2022	FK-131793	OTKA	Leader

Resulting publications:

No.	Title	Venue	Rank	IF
1	A taxonomy and survey of IoT ... [142]	EAI IoT	-	-
2	The use of MQTT in M2M and IoT ... [128]	IEEE Access	Q1	3.367
3	Stress-testing MQTT brokers ... [129]	MDPI Energies	Q1	3.252
4	A mobile iot device simulator ... [102]	Springer JoGC	Q1	2.095
5	Traffic management for cloud ... [46]	Springer LNCS	Q2	-
6	A private gateway for investigating ... [143]	CLOSER conf.	-	-
7	A crawling approach to facilitate ... [145]	IOTSMS conf.	-	-
8	Designing an iot-cloud gateway ... [146]	Springer CCIS	Q4	-

Investigating IoT-Fog-Cloud Systems 3

INTERNET of Things (IoT) is estimated to reach over 75 billion smart devices around the world by 2025 [176], which will dramatically increase the network traffic and the amount of data generated by them. The IoT paradigm composes sensors, actuators, and smart devices into a complex system through the Internet, which are utilised in various domains such as smart homes, smart cities, healthcare, agriculture, and transportation. IoT systems often rely on Cloud Computing (CC) [154] solutions, because of their ubiquitous and theoretically infinite, elastic computing and storage resources.

In the previous chapter, we have seen that IoT-Cloud systems can be utilised in many application areas ranging from local smart homes to mid-range smart cities, or wider smart regions. To cope with the possibly huge number of communicating entities, data management operations are better placed close to their origins, resulting in better exploitation of the edge devices of the network. In the distributed computing paradigm called Fog Computing (FC) [114], groups of such edge nodes form a fog, where data processing and analysis can be performed with reduced service latency and improved service quality compared to remote cloud utilisation. As a result, cloud and fog technologies can be used together to aid the data management needs of IoT environments, but their application gives birth to complex systems that still need a significant amount of research.

IoT-Fog-Cloud systems have to deal with various challenges, to ensure reliable IoT services [188]. Emerging issues of the IoT-to-Cloud continuum, such as connectivity problems of IoT devices, task offloading of computing nodes, billing and operation costs of substantial components, and resource provisioning, are typically addressed by researchers. However, hiring physical machines from virtual server parks that fit various IoT scenarios could be very expensive, and the investigation of IoT-enabled service compositions is not always possible with real cloud providers. It is obvious that significant investments, design and implementation tasks are required to create such IoT-Fog-Cloud systems in reality, therefore, it is 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.

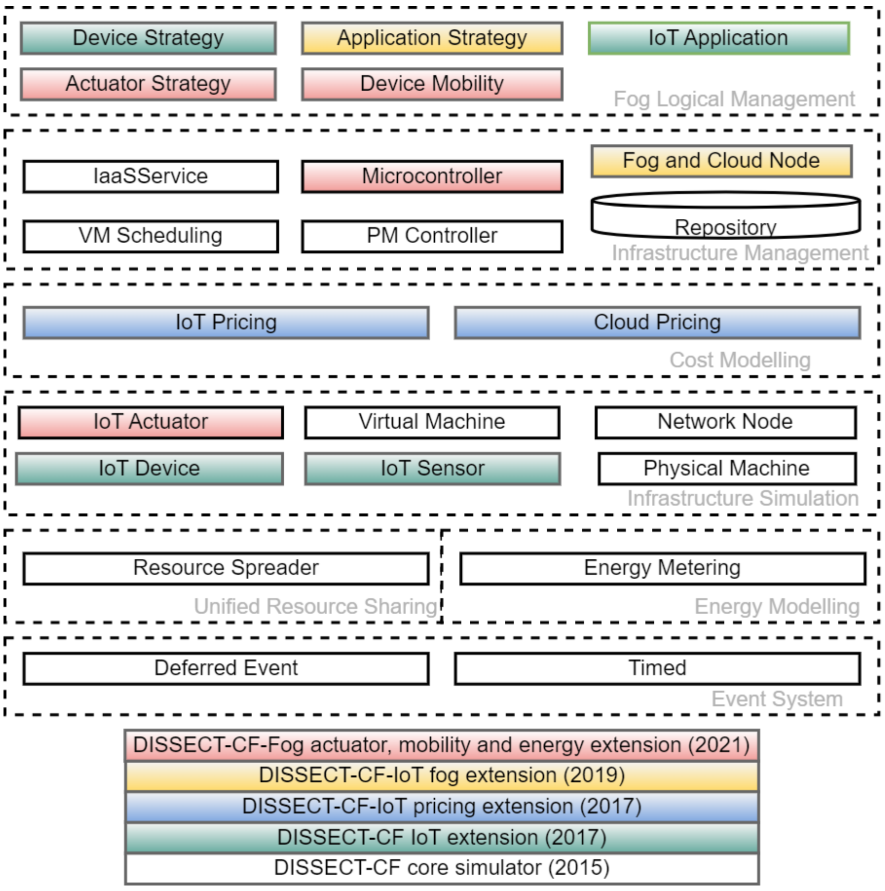


Figure 3.1: *The evolution of DISSECT-CF-Fog through its components*

In this chapter, I present our research efforts towards the development of a general purpose simulator called DISSECT-CF-Fog, which is capable of simulating IoT-Fog-Cloud systems. It is based on the DISSECT-CF discrete event IaaS (Infrastructure-as-a-Service) cloud simulator [92], and extends it in four phases with: IoT sensor and device modelling, cloud and IoT service usage pricing, fog infrastructure modelling, finally mobility, actuation and energy metering features. The basic (uncolored) and the extended (colored) system components are depicted in Figure 3.1 [122].

There are already existing survey papers highlighting the basic capabilities of simulation tools modelling such complex systems, also comparing them by certain views, e.g. by Ragman et al. [156] or by Puliafito et al. [152]. Nevertheless, 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 remainder of this chapter is organised as follows: in Section 3.1 I summarise the most relevant related work in the field of IoT-Fog-Cloud simulation, then in Section 3.2 I detail our iterative extensions to the base DISSECT-CF simulator to reach the current set of functionalities.

3.1 Modelling and Simulating Fog Computing

There are already a couple of survey papers addressing different aspects of Cloud Computing and Fog Computing techniques supporting IoT systems.

Yousefpour et al. [189] presented a survey on Fog Computing and made comparisons with the following main points of view: computing paradigms such as cloud, fog, mobile, mobile-cloud, and edge, and frameworks and programming models for real applications, software, and tools for simulated applications.

Svorobej et al. [174] examined different fog and edge computing scenarios, and investigated Fog and Edge Computing with various aspects, such as (i) application level modelling, (ii) infrastructure and network level modelling, (iii) mobility and (iv) resource management, and (v) scalability, and made a comparison of seven available fog tools.

Hong and Varghese [80] summarised resource management approaches in Fog and Edge Computing focusing on their utilised infrastructure and algorithms, while Ghanbari et al. [68] investigated different resource allocation strategies for IoT systems. This second work classified the overviewed works into eight categories: QoS-aware, context-aware, SLA-based, efficiency-aware, cost-aware, power-aware, utilisation-aware, and load balancing-aware resource allocation. We found this work particularly interesting, hence it opened the investigation into the field of IoT.

Puliafito et al. [152] investigated the benefits of applying Fog Computing techniques to support the needs of IoT services and devices. In their survey, they described the characteristics of fogs and introduced six IoT application groups exploiting fog capabilities. They also gathered fog hardware and software platforms that support the needs of these IoT applications.

Table 3.1: *Comparison of related surveys according to their main contributions*

Survey	Year	Aim
Yousefpour et al. [189]	2018	Paradigms and research topics for fogs
Hong and Varghese [80]	2018	Resource management in the fog and edge
Svorobej et al. [174]	2019	Fog and edge simulation challenges
Ghanbari et al. [68]	2019	Resource allocation methods for IoT
Puliafito et al. [152]	2019	Fog characteristics for IoT
Our survey	2019	Fog models and software quality of simulators

Table 3.1 summarises the relevant surveys that represent the current state-of-the-art in relation to our research aims [118]. Although cloud solutions are dominating,

some started to open investigations towards fog, edge, and IoT fields. In the previous chapter, I summarised the most important IoT simulators, here I present a summary of the most relevant fog simulators analysed and published in our detailed survey paper [118] in 2020.

In the development process and foundations of simulators offering fog modelling capabilities, we can identify a close relation to clouds: most of them are extensions of cloud or IoT simulators. Nevertheless, they follow different architectural models: some have centralised, while others have decentralised or peer-to-peer communication schemes. As a result, understanding the model elements, functions, and implementation details can be very hard and time-consuming; that was one of the issues our survey aimed to relax.

Brogi et al. [44] presented a novel cost model to deploy applications on a fog infrastructure. The approach is based on a simulation prototype called FogTorchΠ, which determines a deployment in the fog according to actual resource consumption and cost needs. The main capabilities of this simulator are the monthly cost calculation extended with subscription/data transfer cost of IoT devices (using bandwidth and latency parameters), and it takes into account geolocation information.

The PureEdgeSim [53] tool was proposed to design and model cloud, fog, and edge applications focusing on the high scalability of devices and heterogeneous systems. PureEdgeSim is based on CloudSim Plus, and it uses XML descriptions for the simulation, where the users can configure the data and parameters of geographical location, energy model and virtual machine (VM) settings.

The IoTSim-Edge simulation framework [87] extends the CloudSim model towards IoT and Edge systems. It focuses on resource provisioning for IoT applications considering the mobility function and battery usage of IoT devices, and different communication and messaging protocols as well. The IoTSim-Edge contains no dedicated class for the actuator components, nevertheless, the representative class of an IoT device has a method for actuator events, which can also be overridden. This simulation tool also considers the mobility of smart devices. The location of a device is represented by a three-dimensional coordinate system. Motion is influenced by a given velocity and range, where the corresponding device can move, and only horizontal movements are considered within the range by the default moving policy.

One of the most referred fog simulators is iFogSim [73], which is also based on CloudSim. iFogSim can be used to simulate real systems and follows the sensing, processing and actuating model, therefore, the components are separated into these three categories. The main physical components are the following: (i) fog devices (including cloud resources, fog resources, smart devices) with the possibility to configure CPU, RAM, MIPS, uplink- and downlink bandwidth, busy and idle power values, (ii) actuators with geographic location and reference to the gateway connection, (iii) sensors that generate data in the form of a tuple representing information. The main logical components aim to model a distributed application: the (i) AppModule is a processing element of iFogSim, and the (ii) AppEdge realises the logical data flow between the virtual machines. The main management components are as follows: the (i) Module Mapping searches for a fog device to serve a virtual machine, if no such device is found, the request is sent to an upper-tier object, and the (ii) Controller launches the application on a fog device. To simulate fog systems, we first have to define the physical components, then the logical components, and finally the controller entity. Although numerous articles and online source codes are available for the use of this simulator, based on our experiments, we suggest that there is a lack of source code comments for many methods, classes, and variables. As a result, application modelling with this tool requires a relatively long learning curve, and its operations take valuable time to understand. After its appearance, it has been used for many research works and various experiments. For example, in [136] a smart city network architecture is presented for Fog Computing called FOCAN as a case study, and in [23] the authors proposed to combine Fog and the Internet of Everything into the so-called Fog of Everything paradigm.

There are also more advanced extensions of iFogSim. MyiFogSim [111] was designed to manage virtual machine migration for mobile users. The main capability of this simulator is the modelling of user mobility and its connection with the VM migration policy. The evaluation contains a comparison to a simulation without VM migration. MobFogSim [153] aims to model user mobility and service migration, and it is one of the latest extensions of iFogSim, where actuators are supported by default. Furthermore, the actuator model was revised and improved to handle migration decisions, because migration is often affected by end-user motions. To represent mobility, it uses a two-dimensional coordinate system, the users' direction and velocity. The authors considered real datasets as mobility patterns, which describe buses and routes of public transportation. EdgeCloudSim [170] is another CloudSim extension with the main capabilities of the network modelling extension for WLAN, WAN, and device mobility. They aimed to respond to the disadvantage of iFogSim's simple network model, which ignores network load and does not provide content mobility. SpanEdge [162] is another decentralised tool related to edge computing, aiming to model data stream processing. Developers can build up a geographically distributed network by installing parts of a stream processing application near the data source for latency and bandwidth reduction. We can also find simulation solutions in the field of Fog Computing exploiting the use of Matlab, Docker or other real service APIs.

DISSECT-CF-Fog is our proposed solution (to be discussed in the next section) to simulate IoT-Fog-Cloud systems, written in Java programming language, and has several extensions to the core DISSECT-CF simulator.

Table 3.2 compares the works we shortly introduced above [118]. It shows that

Table 3.2: *Comparison of the examined fog simulators with software metrics*

Simulator	Language	Lines of code	Comments (%)	Duplication (%)	Files	Bugs	Vulnerabilities	Code smells
FogTorchΠ	Java, XML	2,748	15.9	8.3	39	21	31	308
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
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
DISSECT-CF-Fog	Java, XML	9,870	33.3	2.0	118	31	192	482

the iFogSim simulator has relatively bad software quality: about one-fourth of its code is duplicated, and it has more than 25 thousand lines of code having more than 1,500 code smells, which is the worst out of all simulators. On the contrary, DISSECT-CF-Fog and PureEdgeSim have the best ratio of duplication, and DISSECT-CF-Fog has the best ratio of comment lines. As it revealed, software re-engineering methods are highly encouraged to be used in the future to arrive at reliable and maintainable extensions.

Table 3.3: *Mobility-related characteristics of simulation tools*

Simulator	Communication direction	Actuator events	Mobility	Position
DISSECT-CF-Fog	- Sensor $\rightarrow$ Fog / Cloud $\rightarrow$ Actuator - Sensor $\rightarrow$ Actuator	- 10 different predefined actions for actuation - Adding new by overriding	- Nomadic - Random Walk	Latitude, Longitude
iFogSim	- Sensor $\rightarrow$ Fog $\rightarrow$ Actuator	- Default, but it can be overridden	-	Coordinates
IoTSim-Edge	- Sensor $\rightarrow$ Fog Device $\rightarrow$ Actuator	- Default, but it can be overridden	- Linear	Coordinates
YAFS	- Sensor $\rightarrow$ Service $\rightarrow$ Actuator	-	- Real dataset	Latitude, Longitude
MobFogSim	- Mobile Sensor $\rightarrow$ Mobile Device $\rightarrow$ - Mobile Actuator	- Migration	- Linear - Real dataset	Coordinates

Considering fog modelling capabilities in general, our survey highlighted that models of Fog Computing were evolving more rapidly, than the ones for IoT, based on the latest publications, which was approved by the fact that we found twice as many solutions for the fog compared to IoT solutions. There were many independently developed tools (half of the considered studies) we found, and six works extended some CloudSim solution in a direct or indirect way. It also seemed that former (purely) network simulation solutions were not considered anymore to be the base of new extensions for Fog Computing.

The actuator and mobility abilities of the considered simulators are summarised in Table 3.3, which is the result of our additional literature review presented in [122].

The second column shows possible directions for transferring the sensor data (usually in the form of messages), in case the actuator interface is realised in the corresponding simulator. The third column highlights actuator events that can be triggered in a simulator. The fourth column shows the supported mobility options (we only listed the ones offered in their source code) and finally, we denote the position representation manner in the last column. It can be observed that there is a significant connection between mobility support and actuator functions, but only half of the investigated simulators applied both of them. Since the actuator has no commonly used software model within the latest simulation tools, developers omit it, or it is left to the users to implement it, which can be time-consuming (considering the need for additional validation). In a few cases, both actuator and mobility models are simplified or just rudimentary handled, thus realistic simulations cannot be performed.

Modelling Fog Computing is the latest direction in simulation features, and it tries to build on previous cloud and IoT system simulators. This area has been under active research over the past decade. Focusing on the currently available fog simulators, we can summarise that though usable solutions can be found for analysing specific fog capabilities or use cases, in general, complex fog environments are still hard to be addressed with a single tool. It is also unlikely that near future solutions would target coming up with a general simulator, rather extensions will appear to cover more fog management-related properties. One direction, which has already been started, is the modelling of container-based fog node behaviour, another one is the location-aware management of fog nodes. Cost modelling and energy-aware management are also missing features in many simulators, hence they still need extensive research. The sensor models are quite simple in most tools, therefore, sensor and device behaviour analysis and modelling should also be a target feature of future research.

For simulating distributed systems using emerging technologies, we can state that CloudSim-based extensions (e.g. iFogSim or EdgeCloudSim) are the most popular choices, 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, more scalable, and more reliable than CloudSim (see [116]). 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. Besides the paper-based evaluation and review of simulators capable of modelling Fog Computing, we also performed experiments on executing and comparing concrete simulations with DISSECT-CF-Fog and iFogSim, which is the most popular simulator in this field. To compare the performance of the two simulators, we performed various experiments with a unified scenario implemented in both systems. A detailed description of the scenario and the evaluation results can be read in [119]. According to results of these measurements, we could observe that DISSECT-CF-Fog is better suited for high-scale simulations, while iFogSim can become intolerably time-consuming by modelling higher than a certain number of entities.

We also used a simple source code metric to compare the implemented scenarios in the simulators. The so-called lines of code (LOC) is a common metric for analysing software quality. It is interesting to see that the same, unified scenario could have been written three times shorter in case of DISSECT-CF-Fog, than in iFogSim. Of course, we tried to implement the code in both simulators with the least number of methods and constructs (in Java language). In general, we managed to model an IoT-Fog-Cloud environment with both simulators and investigated a meteorological IoT application execution on top of it with different sensor and fog and cloud resource numbers. While DISSECT-CF-Fog dealt with these simulations with ease, iFogSim struggled to simulate more than 65 entities of this complex system. Nevertheless, it is obvious that there are only a small number of real-world IoT applications that require only hundreds of sensors and fog or cloud resources; we need to be able to examine systems and applications composed of hundreds of thousands of these components, which can only be simulated by using DISSECT-CF-Fog.

3.2 The Evolution of DISSECT-CF-Fog

In the previous chapter, we have seen that there are already several solutions to enable the investigation of IoT-Cloud systems, and we proposed a semi-simulation framework that supports decision making in the design and development phases of IoT applications. In this chapter I introduce a more generic tool, which is capable of analysing simulations at much higher scales, than the ones we could consider with MobIoTSim. In this case, our research aimed at supporting simulations of up to thousands of IoT devices participating in previously unforeseen/existing IoT scenarios that have not been examined before in detail (e.g. 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 the connectivity and maximum update frequency of their network gateway. Actuators are entities also limited by their network connectivity and reaction time (e.g. how long it takes to actually perform an actuation action). They also have a unique feature that allows changing the location of non-cloud entities. 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 DISSECT-CF-Fog simulator extensions presented in this chapter take into account the following IoT components: sensors, actuators and devices (i.e. gateways or brokers), and applications (deployed in a distributed computing service, i.e. cloud and fog).

3.2.1 Pricing Model for IoT-Cloud Applications

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 [166]. IoT systems are currently expanding rapidly as the amount of smart devices (sensors with networking capabilities) is growing substantially, and the costs of sensors are decreasing at the same time.

IoT solutions are often used a lot within businesses to increase the performance in certain areas, and allow for smarter decisions to be made based on more accurate and valuable data. Businesses have grown to require IoT systems to be accurate, as decisions based on their data are heavily relied on. However, many sensors have different behavioural patterns. For example, a heart rate sensor has a different behaviour compared to a light sensor, since a heart rate sensor relies on human behaviour as well, which is inherently unpredictable, whereas a light sensor could be predicted quite accurately based on the time of day and location. Predicting how a sensor may impact a system is important as companies generally want to leverage the most out of an IoT system. An incorrect estimation of their performance could possibly have a negative impact on the performance of other systems (e.g. using too many sensors could flood the network, potentially causing inaccurate data, slow response times, or even system crashes). As there are many ways a sensor can behave, it is difficult to predict the effects they may have on the overall system, therefore, their interaction and operation must be analysed. Performing such testing could be costly, time-consuming, and high-risk if the infrastructure has to be created and a wide range of sensors are purchased before any information is obtained about the planned system behaviour. It is even more difficult to determine the impact of a prototype system on the network, as there may be limited or no physical sensors available to perform the tests. An example of this issue is the introduction of soil moisture sensors that analyse soil properties in real-time and adjust water sprinklers to ensure crops have the correct conditions to grow. In order to test such an IoT system in the real world, a huge number of sensors would be required, which could be quite costly and difficult to implement.

To represent an IoT device with sensors, the following parameters should be considered: it has a unique identifier and its lifetime is specified with start time and stop time. The cardinality of the supervised sensor set must be defined as well. Alongside the set cardinality, the average data produced by one of the sensors can be specified in the set. Data generation frequency could be set for the sensors (e.g. in milliseconds) to determine the time interval between two measurements. Each device controls its local storage with a predefined capacity and its network connectivity to the outside world is specified by bandwidth and latency values.

IoT applications that receive, process, and store IoT data can be modelled by building on the IaaS cloud representation layer of DISSECT-CF. The following properties are taken into consideration by modelling an IoT application in our DISSECT-CF-Fog extension: the types of virtual machines used by the application and the task size attribute, which aggregates IoT data into a processing unit for the VM.

In our simulation model, we aimed to investigate certain IoT-Cloud applications, therefore, we needed to define and monitor the following parameters: the number of sensors or devices used, the total number of messages (and their data) sent in a certain period of time, and the uptime and capacity of virtual machines used to provide data processing services. Based on these parameters, we can estimate how our application would be charged after operating its system for a certain amount of time at a concrete IoT-Cloud provider.

In a work [89] in 2017, we considered the following most popular providers to create a pricing scheme model: (i) Microsoft and its IoT platform called Azure IoT Hub[1], (ii) IBM's Bluemix IoT platform[2], the services of (iii) Amazon (AWS IoT)[3], and (iv) Oracle's IoT platform[4]. We took into account the prices publicly available on the websites of the providers.

The calculation of the prices depends on different methods. Some providers bill only according to the number of messages sent, while others also charge for the number of devices used. The situation is very similar, if we consider virtual machine rental or application service prices. One can be charged after GB-hour (uptime) or according to a fixed monthly service price. This price also depends on the configuration of the virtual machine or the selected application service, especially the amount of RAM used or the number of CPU cores or their clock signal.

Regarding the IoT-side pricing in 2017, the Azure IoT Hub charged after the chosen edition/tier. This means that there were certain intervals defined for the number of messages used in a month. There were also restrictions for message sizes, which depended on the chosen tier (Free, S1, S2, and S3). Each of them varied in prices, and in the total messages allowed per day. The IBM Bluemix (later IBM Cloud) IoT platform only charged after the MB of data exchanged. They differentiated three categories in terms of data usage, and each of them came with a different price per MB. Amazon's IoT platform used two components for pricing: publishing costs (the number of messages published to AWS IoT) and delivery costs (the number of messages delivered by AWS IoT to devices or applications). Finally, we investigated the pricing of Oracle's IoT solution, which was slightly different from the three providers described before. It defined four product categories regarding the types of used de-

[1]MS Azure IoT Hub (accessed in January, 2017): https://azure.microsoft.com/en-us/services/iot-hub

[2]IBM Bluemix (Accessed at January, 2017): https://www.ibm.com/cloud-computing/bluemix/internet-of-things

[3]Amazon AWS IoT (accessed in January, 2017): https://aws.amazon.com/iot/pricing

[4]Oracle IoT platform (accessed in January, 2017):
https://cloud.oracle.com/en_US/opc/iot/pricing

vices (wearable, consumer, telematics, and business). The categories determined the monthly device price, and the number of messages that could be sent by that particular type of device. In addition, there was a restriction on how many messages a particular type of device could deliver per month. In case the number of messages sent by a device was more, than the device's category permitted, an additional price would be charged according to a predefined price per thousand messages.

Concerning cloud-side costs, most providers have a simple calculation method, which is the following: (i) to run an IoT application one needs at least one virtual machine (VM), container, compute service, or application instance, which has a fixed instance price every month, or (ii) the providers consider the hour per price for every instance the IoT application needs. We considered Azure's application service[5], Bluemix's runtime pricing sheet under the Runtimes section[6], Amazon EC2 On-Demand prices[7], and Oracle's compute service[8] together with the Metered Services pricing calculator[9]. The cloud cost is based on either instance prices (Azure and Oracle), hourly prices (Amazon), or the mix of the two (Bluemix), in which the provider uses both types of price calculation. For example, Oracle charged depending on the daily uptime of our application, as well as the number of CPU cores used by our VMs.

After carefully examining the pricing methods of these providers, we created an XML-based description with the actual categories and prices (for details, see: [121]). This description can be loaded into the DISSECT-CF-Fog simulator and used for automatic usage cost calculation (which can be considered as an estimation), performed by the simulator, based on the utilised resources in the actual simulation experiment.

As one of the earliest examples of sensor networks is from the field of meteorology and weather prediction, we chose to model the crowd-sourced meteorological service of Hungary called Idokep.hu[10], to exemplify the usage of our cost model. It was established in 2004, and it is one of the most popular meteorology websites in Hungary. In 2017, it had more than 400 stations send sensor data to their system (including temperature, humidity, barometric pressure, rainfall, and wind properties), and actual weather conditions were refreshed every 10 minutes. They also offered sensor stations to be installed at buyer-specific locations, in order to extend their sensor network.

[5]MS Azure price calculator (accessed in January, 2017): https://azure.microsoft.com/en-gb/pricing/calculator/

[6]IBM Bluemix pricing sheet (accessed in January, 2017): https://www.ibm.com/cloud-computing/bluemix

[7]Amazon pricing (accessed in January, 2017): https://aws.amazon.com/ec2/pricing/on-demand/

[8]Oracle pricing (accessed in January, 2017): https://cloud.oracle.com/en_US/opc/compute/compute/pricing

[9]Orcale Metered Services pricing calculator (accessed in January, 2017): https://shop.oracle.com/cloudstore/index.html?product=compute

[10]Idokep.hu (accessed in February, 2017): http://www.idokep.hu

Table 3.4: *Basic configuration information of the application*

Devices with sensors	3,864
Device type	Consumer
Message size (KB)	0.05
Messages / month / device	4,464
Total messages / day	556,416
Total messages / month	17,248,896
MB exchanged / month	842.23125
Messages transferred / device / hour	6
Test duration (days)	31
Full uptime (hours)	744

We created a model application of this service, which has the following detailed properties, while operated over a month, as shown in Table 3.4 [121]. It highlights that 483 stations composed the IoT-Cloud system with 3864 sensors, each sending 4464 messages per month by monitoring the environment. To exemplify the use of our cost estimation solution, we performed experiments by simulated operation of this meteorological application for a period of one month. Table 3.5 shows the estimated prices to be paid by using these providers [121]. The results showed that Amazon came with the best price of ∼241 Euros, while Oracle was incredibly expensive compared to the other providers with ∼3862 Euros.

Table 3.5: *Cost estimation for the meteorological case study*

Provider / Cost	Azure	Cost	Bluemix	Cost	Amazon	Cost	Oracle	Cost
IoT fix prices and device side								
Device price / month	-		-		-		+	3,593.52
”Price / message” pricing	-		-		+	78.69	+	
”X messages / month” pricing	+	421.65	-		-		+	0
Data exchange (in MB)	-		+	0.82	-		-	
Message size limit	4		-		0.5		-	
Total messages / day with size limit	556,416				552,960			
Cloud side								
Instance prices	+	188.22	+	245.38	-		+	268
GB-hour prices	-		0.05	39.13	0.01	162.53	-	
TOTAL PRICE / MONTH		609.87		285.33		241.22		3,861.52

After we have introduced these pricing schemes, we turned our research focus on how resource utilisation and management patterns alter, based on the changing sensor behaviour, and how these changes 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). We presented various measurement scenarios in [121]. Here, we highlight only the first scenario of this work, in which we varied the amount of data produced by the sensors of 486 weather stations: we set 50, 100, and 200 bytes for different cases (allowing overheads for storage, network transfer, different data formats, and secure encoding etc.). We also investigated how the costs of the IoT side changed, if we would use one of the four IoT providers defined before. In this experiment, we limited the station lifetimes to a single day. A station used 8 sensors and a sensor measurement was executed every minute. The start-up period of the stations was selected randomly between 0 and 20 minutes in the simulator. The task creator daemon service spawned tasks after the cloud storage received the metering data (i.e. a message), but the size of a task could not exceed 250 KB. This step ensured the estimated processing time of 5 minutes/task. If a task was started with less than 250 KB to process, the execution time was scaled down. Finally, the application was using a cloud configuration based on the IBM Bluemix VM instance having 8 CPU cores and 4 GB memory, and its cost was set as shown in Table 3.5.

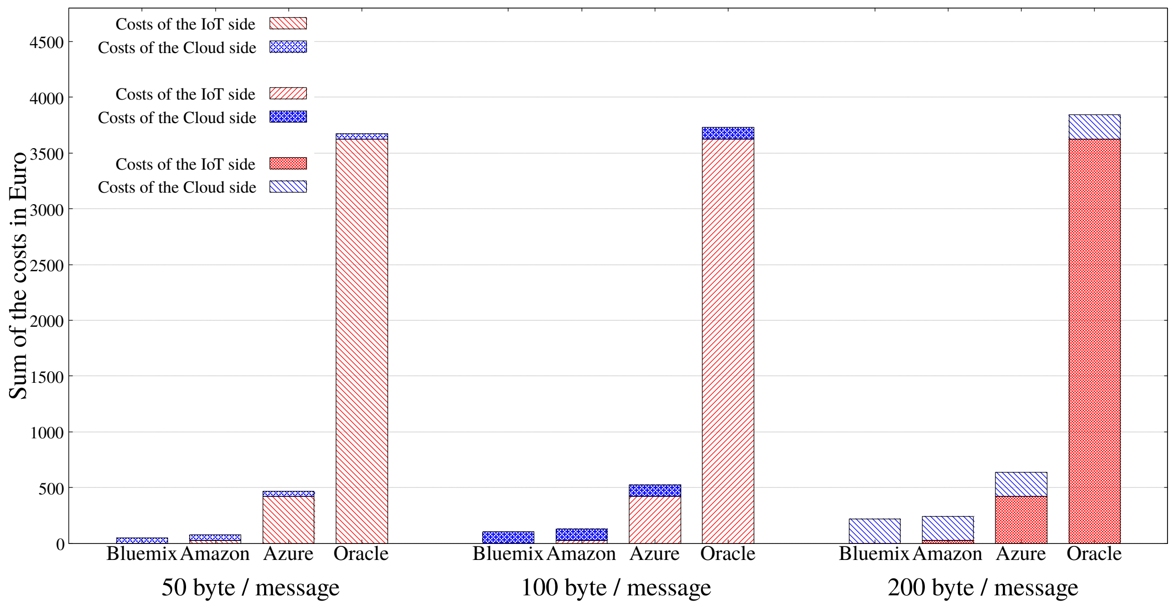


Figure 3.2: *IoT and cloud costs in the experiment*

Table 3.6: *Number of VMs, tasks, and the amount produced data*

Amount of data (byte)	Number of VMs	Number of tasks	Produced data (GB)
50	12	1,153	0.261
100	27	2,299	0.522
200	28	4,486	1.044

For the first case with 50 bytes of sensor data, we measured 0.261 GBs of produced data in total, while in the second case of 100 bytes, we measured 0.522 GBs, and in the third of 200 bytes, we measured 1.044 GBs (showing linear scale up). In the three use cases, we needed 12, 27, and 28 VMs to process all tasks respectively. With the preloaded cloud parameters, the system is allowed to start a maximum of 28 virtual machines, therefore, in the first case of 50 bytes our cloud cost was 48.839 Euros, in the second case of 100 bytes the cloud cost was 103.896 Euros, and finally, in the last case, our cloud cost was 217.856 Euros. The lesson learnt from this scenario was that if we used more than 200 bytes per message, we needed stronger virtual machines (also a larger cloud with stronger physical resources) to manage our application, because in the third case the simulation run for more than 24 hours (despite the fact that the sensors were only producing data for a single day), which increased our costs using time-dependent cloud services. Finally, Table 3.6 shows how many virtual machines were needed to process all of the generated data for all test cases, and how many tasks were generated for the produced data [121].

Figure 3.2 presents a cost comparison for all considered providers [121]. 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.

In summary, we have shown that with our extended DISSECT-CF-IoT simulator, we can investigate the behaviour and operating costs of these systems and contribute to the development of better design and management solutions in this research field.

3.2.2 Analysis of Multi-layered Fog Environments

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. Sensors are mostly resource-constrained and passive entities with restricted network connection, on the other hand, actuators ensure broad functionality with an Internet connection and enhanced resource capacity [138]. They aim to make various types of decisions by assessing the processed data retrieved from the nodes. These actions can affect the physical environment or refine the configuration of the sensors. Furthermore, the embedded actuators can manipulate the behaviour of smart devices, for instance, restart or shut down a device, and motion-related responses can also be triggered. Systems composed of IoT devices supporting mobility features are also known as the Internet of Mobile Things (IoMT) [133]. 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. Furthermore, using purely cloud services can limit the support for mobility [149].

In order to serve actuator and mobility decisions, first we had to introduce an offloading mechanism for IoT applications in DISSECT-CF-Fog. In case a selected fog node is overloaded, the execution of the appropriate task will be delayed, which has a negative effect on the makespan of its application and/or on the execution costs. To reach the required QoS of an IoT application, the management of IoT-Fog-Cloud systems should also take into account the position and the actual load of the fog nodes.

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. Such functions can take into account the physical location of the entities of the execution environment (i.e., fog or cloud node or IoT devices), the load of the network used for communication and data transfers, and the transfer, storage, and execution costs.

To overcome this problem, we introduced a multi-layer fog node management in DISSECT-CF-Fog by enabling task offloading from (possibly overloaded) nodes to others [123]. A typical fog topology can contain numerous nodes, some of them are grouped as a fog cluster, which restricts the access and visibility of other nodes. These nodes can be ordered into layers, where higher-level fog layers usually contain stronger physical resources. The computing nodes can be heterogeneous and often restricted on certain types and strengths of VMs and applications in order to serve as many users as possible, such restrictions usually concern the processing power and the memory usage of the given VM.

In the simulator the following parallel steps are performed during an experiment: (i) sensors of smart devices generate data in the given frequency, (ii) the unprocessed data are forwarded to a node, (iii) the data are packaged in a compute task, (iv) VMs on a node process as many tasks as they can, and finally (v) if a node is overloaded, then the unprocessed tasks will be forwarded to another node.

To manage the offloading decisions separately for each node, we introduced different application strategies and different task allocation approaches adhering the characteristics of the considered multi-layer fog and cloud topology. We performed and presented detailed experiments in [123], demonstrating these introduced capabilities and new components of the simulator. We defined four basic strategies for task allocation. 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 proposed was a Pliant system-based 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). The Pliant system, which is a kind of fuzzy theory that is similar to a fuzzy system [58]. The difference between the two systems lies in the choice of operators. In fuzzy theory, the membership function plays an important role, but the exact definition of this function is often unclear. In the Pliant system, we use a so-called distending function, which represents a soft inequality. Furthermore, the various operators, which are called conjunction, disjunction, and aggregative operators, are closely related to each other. We also have a generator function, which can be used to create such operators. In the Pliant system, the corresponding aggregative operators of the strict t-norm and strict t-conorm are equivalent, and De Morgan's law is obeyed with the corresponding strong negation of the strict t-norm or t-conorm. For each reachable fog and cloud node the *Pliant* strategy calculates a score number using normalisation, Sigmoid and Kappa functions, and the aggregation operator to choose the appropriate one for the offloading decision. We defined the following three properties for each system node: load, cost, and unprocessed data of a node. In Table 3.7, we can see the exact values of the normalisation functions we used [123]. These properties can be used in the Pliant algorithm to rank fog nodes, from which we choose from in case of offloading. More details on the applied model and the operation of the algorithm can be read in [123].

The stakeholders of IoT applications and resource providers have to prepare in advance for the unforeseen data generated by sensors and often aspire to treat them in real-time, thus a fog topology should be robust enough and be able to handle a vast amount of IoT devices. Such requirements can be investigated by defining application and device strategies. A device strategy is applicable to map any device to any preferred node, according to the policy of the actual device. Similarly to other models, resource needs, energy consumption, latency, bandwidth, and IoT/cloud-

Table 3.7: *Normalisation parameters*

Parameter	Lambda	Shift
Workload of a node	-1.0/8.0	(Maximum workload - minimum workload) / 2
Price of using a node	4.0	Minimum price
Unprocessed data	-1.0/4.0	(Maximum unprocessed data - minimum unprocessed data)

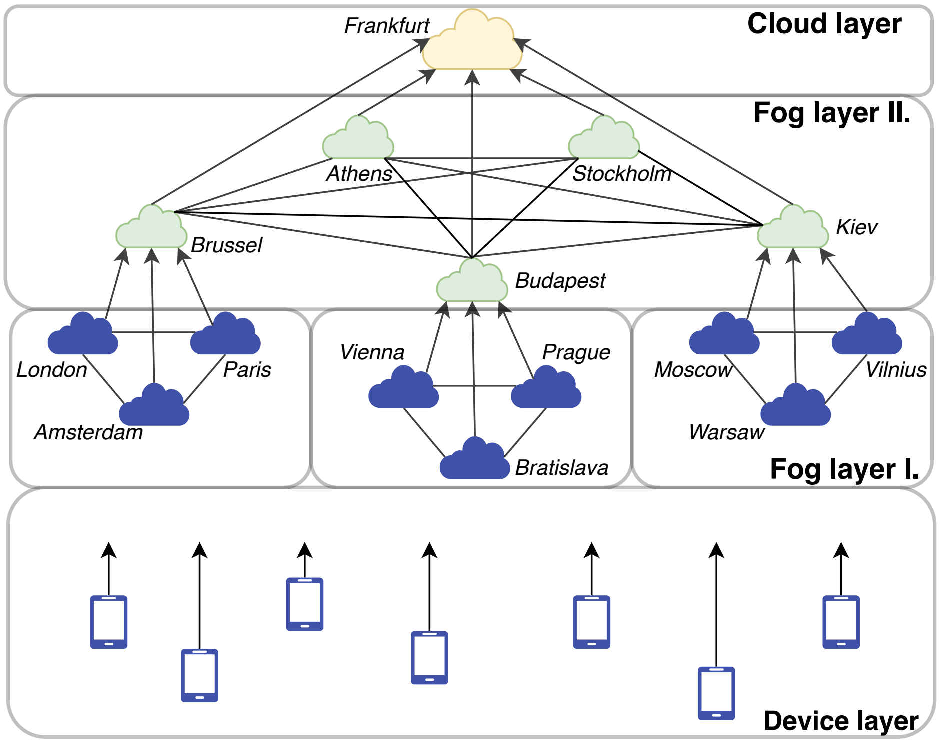


Figure 3.3: *The considered fog topology in the evaluation*

side costs can be considered, in order to ensure appropriate pricing, or to reduce IoT application execution time (i.e. makespan). The *Distance-based* device strategy chooses the geographically closest fog node, to be selected only from the lowest fog layer, for a device to communicate with (i.e. send tasks to). The disadvantage of this strategy lies in data overload in a certain node if many devices were located in its neighbourhood. The *Load-balanced* device strategy always tries to balance the number of connected devices for each node considering the number of available physical machines of the node as well. The disadvantage of this strategy is the increased latency if a device chose a node located farther away from the lowest layer. One can observe that considering only device strategies can cause a bottleneck effect in the topology.

We evaluated the proposed device mapping and task allocation strategies with three different scenarios simulating a European-wide weather forecasting system, as discussed in detail in [123]. Here, I present only the last scenario of this work, where we modelled a dynamic system. In the beginning, only 2,000 stations started to work, and after every four hours we installed 2,000 more, thus at the end of the day, the total number of devices was 12,000. Each station was equipped with five sensors (e.g. measuring weather conditions, such as temperature and humidity, with 50 bytes of sensor data). The time interval between two measurements (i.e. the data generation frequency) was set to one minute.

The topology contained two fog layers with 14 different fog nodes organised into clusters, and one cloud layer had a single cloud node. Each node was represented by a different city, and the latency between them was measured by using WonderNetwork[11]. The defined topology can be seen in Figure 3.3, which also depicts the fog clusters and layers with different colours [123]. The arrows represent routes that are responsible for communication between the layers, while the (undirected) edges are for the message exchanges inside a cluster. The network capability of the smart devices (or stations) was modelled with a 4G network with an average 50 ms of latency. When a connection is created by applying a device policy, the exact latency value is weighted in proportion to the physical distance between the device and the node. The nodes of the topology were modelled with real VM specifications and pricing schemes according to the Amazon Web Services (AWS): the lowest Fog layer has VMs with 2 CPU cores, 4 GB RAM and 0.051$ hourly price, the top fog layer has VMs with 4 CPU cores, 8 GB RAM and 0.101$ hourly price, finally, the cloud layer has VMs with 8 CPU cores, 12 GB RAM and 0.204$ hourly price. Each VM could process only one task (represented by 250 KB of data) at a time. The lowest fog layer of the topology was divided into three clusters, one node of the cluster tackled with 12 CPU cores and 24 GB RAM altogether. We doubled the resource capacity of a node of the upper fog layer, thus it dealt with 24 CPU cores and 48 GB RAM. Our preliminary evaluation showed that applying this scheme made the topology (the system) particularly strong, therefore, some strategies (e.g. *Push Up*) became more beneficial than others. Therefore, the cloud resources were set to 48 GB CPU cores and 96 GB RAM.

Concerning the application management in the simulator during executing a scenario, the ratio of the forwardable data was limited to 50% (hence moving all data to a different node is prohibited). The daemon service of the application decided after every 150 seconds about the task allocation request to a VM of a node. In the evaluation scenario, the locations of all devices were fixed during the simulation, however, those positions were randomly chosen at the beginning of the simulation to enable realistic and unexpected behaviour of such devices. The start time of the devices had a delay randomly set from the 0-20 interval in minutes (to avoid burst operations, and also to be more realistic).

[11]WonderNetwork (accessed in May, 2020): https://wondernetwork.com/pings

Table 3.8: *Evaluation results of a multi-layer fog scenario*

Device strategy	*Distance-based*				
Application strategy	Pliant	Runtime-aware	Random	Hold Down	Push Up
Num. of VMs	90	90	90	60	79
Cost ($)	110.1	111.9	107.8	77.1	105.0
Network utilisation (sec.)	87	131	110	0	69
Data transferred (MB)	2,117	5,471	2,939	0	1,099
Timeout (min.)	**74.8**	144.4	111.6	1,091.0	332.3
Device strategy	*Load-balanced*				
Application strategy	Pliant	Runtime-aware	Random	Hold Down	Push Up
Num. of VMs	90	90	90	51	66
Cost ($)	132.8	138.1	131.5	106.9	134.8
Network utilisation (sec.)	114	204	140	0	40
Data transferred (MB)	4,307	13,599	5,300	0	2,911
Timeout (min.)	195.3	**172.8**	217.8	3,095.0	1,852.3
Device strategy	*Mixed*				
Application strategy	Pliant	Runtime-aware	Random	Hold Down	Push Up
Num. of VMs	90	90	90	60	80
Cost ($)	111.0	112.8	110.2	78.0	107.3
Network utilisation (sec.)	83	121	108	0	55
Data transferred (MB)	2,194	5,126	3,402	0	1,251
Timeout (min.)	**68.7**	112.7	72.3	1,641.0	254.8

Table 3.8 summarises the average results after executing the scenarios with all device and application strategies three times [123]. We also extended our measurements with an additional device strategy called *Mixed*, where half of the IoT devices were managed with the *Distance-based*, and the other half of the devices were managed with the *Load-balanced* strategy. To compare the proposed strategies, we measured how many virtual machines were required to process tasks (and their data) during operating hours. The Network utilisation metric reflects the network load, and it represents the time taken to transfer the sensor data from the source node to the actual processing node. The Data transferred metric represents the total size of all forwarded data. The Data transferred and the Network utilisation metrics were rounded. The Timeout value means the time taken to finish data processing after the last sensor measurement was performed. This metric relates to the makespan of an application. Here, the less time required to process all tasks of an IoT application (after stopping the devices), the better a node selection strategy is. According to the used AWS pricing models, we could calculate the exact costs of the usage of the fog and cloud resources (after executing the IoT applications).

Regarding the results, two of the device strategies (*Distance-based, Mixed*), and the *Plant* application strategy managed the best Timeout values with 74.8 and 68.7

minutes. This shows that an IoT application only handles the data in a fair way, when we applied the *Load-balanced* strategy with the *Runtime-aware* strategy. Concerning the average results for the device strategy, we can see that the best strategy for the operation cost, the network transfer and the timeout value is the *Distance-based* device strategy. In one case, the *Mixed* had a good influence on the network utilisation, which means location independence of the smart devices may require further investigation. We can clearly see that the *Load-balanced* strategy dealt with the least number of virtual machines, and it became the best scenario, but it also dramatically increased the rest of the metric values.

3.2.3 Actuator and Mobility models

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. The actuator interface should facilitate a more dynamic device layer and a volatile environment in a simulation. In the works presented so far we did not exploit actuation and IoT device mobility features, hence they were not represented in detail in the simulator. In our model we aimed to integrate to the DISSECT-CF-Fog simulator, an actuator component is connected to one IoT device for two reasons in particular: (i) it is observing the environment of the smart device and can act based on previously specified conditions, or (ii) it can influence some low-level sensor behaviour, for instance, it changes the sampling interval of a sensor, resets or completely stops the smart device. The latter indirectly conveys the concept of a reinterpreted actuator functionality for simulator solutions.

The actuator component we developed in DISSECT-CF-Fog can also model a low-level software component for sensor devices, which makes the model compound [122]. The actuator model of DISSECT-CF-Fog can operate with compact, well-defined events, that specify the exact influence on the environment or the sensor. The set of predefined events during a simulation provides a restriction to the capability of the actuator, and limits its scope to certain actions that are created by the user. The determination of the exact event executed by the actuator, happens in a separate, reusable, and extendable logic component. This logic component can serve as an actual actuator configuration, but can also be used as a descriptor for environmental changes and their relations to specific actuator events. This characteristic makes the actuator interface thoroughly flexible and adds some more realistic factors to certain simulation scenarios. With the help of the logic component, the actuator interface works in an automatic manner. After a cloud or fog node has processed the data generated by the sensors, it sends a response message back to the actuator, which chooses an action to be executed. This models the typical sensor-service-actuator communication direction.

The proximity of computing nodes is the main principle of Fog Computing and it has numerous benefits, but mobile IoT devices may violate this criterion. These devices can move further away from their processing units, causing higher and unpredictable latency. When a mobile device moves out of range of the currently connected fog node, a new, suitable fog node must be provided. Otherwise, the quality of service would drastically deteriorate and due to the increased latencies, the fog and cloud nodes would hardly be distinguishable in this regard, resulting in losing the benefits of Fog Computing. Another possible problem that comes with mobile devices is service migration. Service migration usually occurs between two computing nodes, but if there is no fog node in an acceptable range, the service could be migrated to the smart device itself, causing lower performance and shorter battery time.

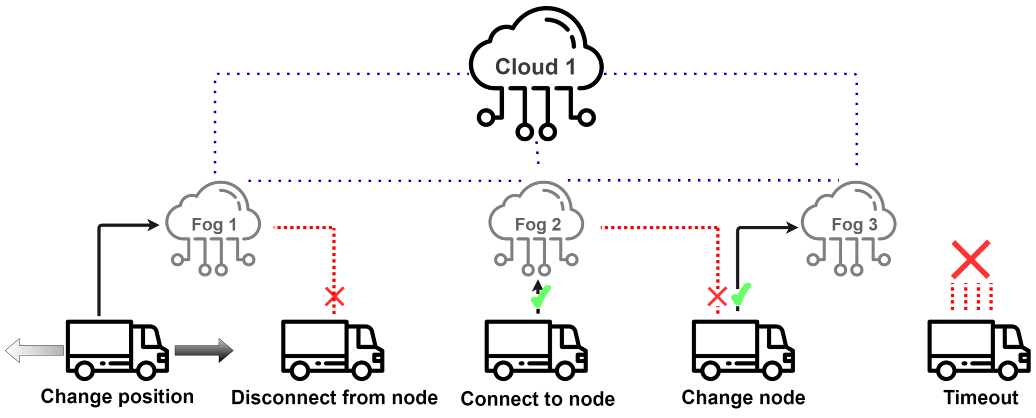


Figure 3.4: *Actuator events related to mobility behaviour*

The physical location of fog nodes in a mobile environment is a major concern. Placing fog nodes too far from each other will result in higher latency or connection problems. In this case, IoT devices are unable to forward their data, hence they are never processed. Some devices may store their data temporarily until they connect to a fog node, but this contradicts the real-time data processing promises of fogs. A slightly better approach would be to install fog nodes fairly dense in space to avoid the problem discussed above. However, there might be some unnecessary nodes in the system, causing a surplus in the infrastructure, which results in resource wastage.

The mobility extension of the DISSECT-CF-Fog was designed to create a precise geographical position representation of computing nodes (fog, cloud) and mobile devices, and simulate the movements of devices based on specified mobility policies. As the continuous movement of these devices could cause connection problems, we considered five actuator events related to mobility, as shown in Figure 3.4 [122]. Position changes are done by the *Change position* event of the actuator. The connection or disconnection methods of a device are handled by the *Disconnect from node* and the *Connect to node* events, respectively. When a more suitable node is available for a device than the already connected one, the *Change node* actuator event is called. Finally, in some cases, a node may remain without any connection options due to its position, or in cases when only overloaded or badly equipped fog nodes are located in its neighbourhood. The *Timeout* event is used to measure the unprocessed data due to these conditions, and to empty the device's local repository, if data forwarding is not possible.

The mobile device movements are based on certain strategies. Currently, two mobility strategies are implemented. We decided to implement one entity and one group mobility model according to [49], but since we provided a mobility interface, the collection of usable mobility models can be easily extended. The goal of the (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 certain use cases. 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 can dynamically change.

Evaluation with a logistics scenario

We evaluated the proposed actuator and mobility extensions of the DISSECT-CF-Fog simulator with two different scenarios in [122], which belong to the main open research challenges in the IoT field [117]. The goal of these scenarios was to present the usability and broad applicability of our proposed simulation extension. We also extended one of the scenarios with larger-scale experiments, in order to determine the limitations of DISSECT-CF-Fog (e.g. determining the possible maximum number of simulated entities).

One of the scenarios we considered was IoT-assisted logistics, where precise location tracking of products and trucks can be realised. It can be useful for route planning (e.g. for avoiding traffic jams or reducing fuel consumption), or for better coping with different environmental conditions (e.g. for making weather-specific decisions). During the evaluation of our simulator extension, we envisaged a distributed computing infrastructure composed of a certain number of fog nodes (hired from local fog providers) to serve the computational needs of our IoT applications.

Besides these fog resources, additional cloud resources could be hired from a public cloud provider. For each of the experiments, we used the cloud schema of LPDS Cloud of MTA SZTAKI [79] to determine realistic CPU processing power and memory usage for physical machines. Based on this schema, we attached 24 CPU cores and 112 GB of memory for a fog node and set at most 48 CPU cores and 196 GB of memory to be hired from a cloud provider to start virtual machines (VMs) for additional data processing.

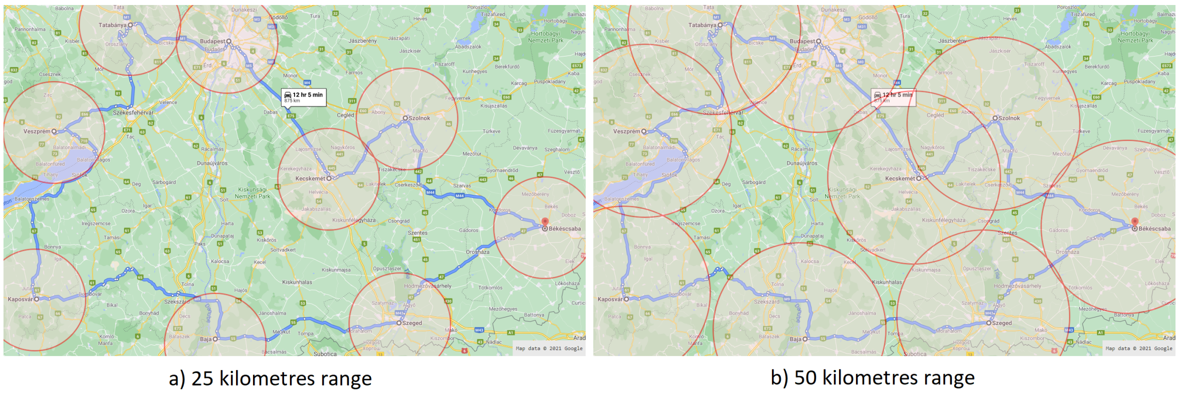


Figure 3.5: *Applied fog ranges in the logistics scenario*

The simulator can also calculate resource usage costs, so we set VM prices according to the Amazon public cloud pricing scheme. For a cloud VM having 8 CPU cores and 16 GB RAM we set a 0.204$ hourly price (*a1.2xlarge*), while for a fog VM having 4 CPU cores and 8 GB RAM we set a 0.102$ hourly price (*a1.xlarge*). This means that the same amount of data is processed twice faster on the stronger cloud VM, however, the cloud provider also charges twice as much money for it. In our experiments, we proportionally scale the processing time of data, for every 50 KB, we model one minute of processing time on the Cloud VM. To perform the experiments, we used a PC with Intel Core i5-4460 3.2GHz, 8GB RAM, and a 64-bit Windows 10 operating system to run the simulations. Since our simulations took into account random factors, each experiment was executed ten times, and the average values are presented below.

In the logistics IoT scenario, 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 we used a single cloud node of a cloud provider located in Frankfurt. Each fog node has a direct connection with the cloud node, the latency between them is set based on the values provided by the WonderNetwork service as before. A fog node forms a cluster with the subsequent and previous fog node on the route as depicted in Figure 3.5

[122]. This figure also presents the first test case (a), when the range of a fog node is considered as 25 kilometres radius (similar to a LoRa network). For the second test case (b), we doubled the range to 50 kilometres radius. The IoT devices (placed in the vehicles to be monitored) were modelled with 4G network options with an average 50 ms of latency.

Table 3.9: *Results of the Transport actuator strategy and number of events during the logistics 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

All vehicles were equipped with three sensors (asset tracking sensor, AIDC (automatic identification, and data capture) and RFID (radio-frequency identification)) generating 150 bytes of data per sensor. In the simulations we set a daemon service on the computational nodes checking the local storage for unprocessed data every five minutes, and allocating them in a VM for processing. Each simulation run dealt with an increasing number of IoT entities, we initialised 2, 20 and 200 vehicles in every twelve hours, which went around on the route. Half of the created objects were intended to start their movements in the opposite direction (selected randomly).

During our experiments, we considered the *Transport* actuator policy, which is a realistic strategy to model asset tracking, which aims to follow objects based on a broadcasting technology (e.g. GPS). A typical use case of this, when a warehouse is prepared for receiving supplies according to the actual location of the truck. In our evaluation, 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 was set to five minutes, and from 10 to 30, the data generation was set to 10 minutes, lastly, if it was farther than 30 kilometres, it informed changes in 15 minutes.

The results are shown in Table 3.9 [122]. Comparisons are based on the following parameters: (i) *VM* reflects the number of created VMs during the simulation on the cloud and fog nodes, which process the amount of generated data. As we mentioned earlier, our simulation tool can calculate the utilisation cost of the resources based on the predefined pricing schemes (*Fog+Cloud cost*). *Delay* reflects the timespan between the time of the last produced data and the last VM operation. *Runtime* is a metric describing how long the simulation runs on the corresponding PC. The rest of the parameters are previously known, it shows the number of the defined actuator and mobility events. Nevertheless, *Timeout data* is highlighting the amount of data lost, which could not be forwarded to any node, because the actual position of a vehicle is too far for all available nodes.

Interpreting the results, we can observe that in the case of the 25-kilometre range, the *Transport* actuator strategy drops about a quarter (around 23,4%) of the unprocessed data losing information. In the case of the 50-kilometre range, there is no data dropped, because the nodes roughly cover the route and the size of gaps cannot trigger the *Timeout* event. In contrast, the ranges do not cover each other in the case of the 25-kilometre range, which results in a zero *Change node* event. Based on the *Fog+Cloud cost* metric, one can observe that the *Transport* actuator strategy utilised 19 cloud and fog VMs, and the average price for a VM was 100$ for managing two vehicles. In case of 20 vehicles, it increased to about 236$, and lastly, operating 200 vehicles the price was increased to about 538$, which means that the more vehicles needed to be managed, the more resources had to be used to process their data. Since the IoT application frequency was set to five minutes, we considered the *Delay* acceptable, when it was equal to or less than five minutes. Based on the results, all test cases fulfilled our expectations. The *Runtime* metric also points to the usability and reliability performance of DISSECT-CF-Fog; less than three minutes was required to evaluate a one-year-long scenario with thousand of entities (i.e. simulated IoT devices and sensors running for a year).

Modelling Energy Consumption

Besides scalability, latency, and resource management issues, reducing the energy consumption of an IoT-Fog-Cloud system is also a great challenge as stated in [22]. Energy-efficient solutions also have a significant impact on reducing the carbon footprint and on slowing down the climate change. The monitoring and measuring of energy consumption entail significant challenges for IoT-Fog-Cloud systems. The initial energy model of the DISSECT-CF simulator covered cloud datacenters by introducing resource consumption modelling for CPU, disk, and network energy utilisation. The energy model of the simulator takes into account: the minimum (min) power (e.g. the machine/device is turned off, but still plugged into the energy source), the maximum (max) power (e.g. if the CPU is fully utilised), and the idle power of a resource (e.g. when the PM is running without executing computational tasks). By default, the simulator has two power models: (i) dynamic power draining behaviour applies linear interpolation between idle and max power values, whilst (ii) constant power draining behaviour can consider any power value (e.g. min). The dynamic model is applied in the case of states with high energy consumption (e.g. running state of a PM), and it handles the idle power with min power values, and the consumption range can be get by subtracting the idle power value from the max power value.

In another work [120], we extended the existing energy model of DISSECT-CF to cover IoT devices to enable complex energy utilisation analysis of IoT-Fog-Cloud systems. To determine a fine-grained energy model for microcontrollers, we measured and collected the energy consumption values of real devices (we chose ESP32 and Raspberry Pi (1 Model A+) microcontrollers). Both devices were equipped with DTH22 temperature and humidity sensors, and a KCX-017 meter was applied to display the voltage and the current of the connected USB port. To measure the general power consumption of IoT applications, we developed a simple Python program for sensor data reading, message creation and sending. Besides our real measurements of a typical use of a microcontroller, we gathered information from related works, such as [115] and [91], since they performed comparative analysis and monitoring of IoT devices. 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 model the power consumption of IoT resources, we had to extend the IoT device representation of DISSECT-CF-Fog to include predefined states for microcontrollers, which allow mapping a certain power consumption to a certain state.

To evaluate the extended, unified energy model for IoT-Fog-Cloud architectures in DISSECT-CF-Fog, we modelled a weather forecasting scenario with numerous weather stations (run by IoT microcontrollers with special sensors). These devices could communicate directly with a fog layer, which contains three different nodes with an equal amount of resources, utilising 40 CPU cores and 40 GB of memory in total. On top of the fog topology, there was one cloud datacenter having 56 CPU cores and 40 GB of memory, furthermore, the devices were not allowed to send messages (unprocessed sensor data) directly to the cloud (they are connected only to the fog). We considered two types of virtual machine images simulating existing Amazon Cloud instances. The *cloud1* node could utilise VMs with 8 CPU cores and 4 GB of memory, their hourly prices were set to 0.202$, while the *fog1*, *fog2* and *fog3* nodes could deploy VMs with 4 CPU cores and 2 GB of memory with 0.101$ hourly price. We also set the IoT-side pricing by applying the IBM Cloud pricing schema, which charged the consumer after the amount of data exchanged (in MB).

In our simulation, the microcontrollers could use either ESP32 or Raspberry Pi energy models, and they were equipped with a temperature-humidity sensor (similar to our previous 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.

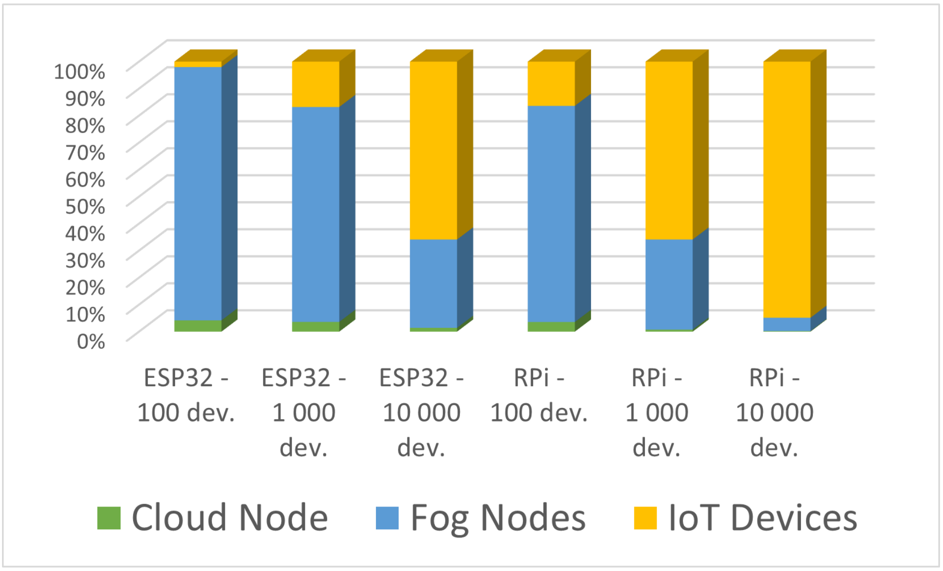


Figure 3.6: *Energy consumption percentage of cloud, fog nodes and IoT devices*

During the evaluation, we modelled a European-wide scenario, where the cloud was located in Frankfurt, whereas the three fog nodes were positioned in London, Budapest, and Vienna. The latency between them was determined based on online ping statistics from WonderNetwork. The delay between a device and a fog node was set to an average of 50 ms weighted with the actual physical distance, and the positions of the devices were randomly generated across Europe. To highlight the energy consumption of the nodes, the number of VMs was scaled up and down dynamically according to the actual load caused by the tasks.

Concerning the evaluation results, the cloud resource utilisation was basically the same in all six simulation cases, because they had to deal with around the same amount of unprocessed data/tasks (coming from the fog layer). Nevertheless, in the case of 1,000 devices, seven VMs could easily handle the scheduled amount of tasks for both microcontrollers. The more data a task contains, the more time it takes for the task to be processed, and additional incoming tasks may trigger new VMs to be deployed (depending on the applied task scheduling policy threshold). In the third case having 10,000 devices, the number of VMs dramatically increased to 29, for both device types.

Figure 3.6 highlights the results by comparing the energy consumption ratio of the utilised cloud node, fog nodes, and IoT devices (i.e. microcontrollers) [120]. As we can see from the diagram, cloud consumption takes only a small part of the total energy consumption in all six scenarios. The fog nodes are mostly capable of handling the vast amount of data with their own resources generated by the IoT layer, and there is no need to involve cloud resources drastically. Nevertheless, when we scale up the number of microcontrollers in the IoT layer, our results show a significant increase in total energy consumption, caused only exclusively by the operation of the IoT devices.

Modelling Dew Computing

As the latest extension of DISSECT-CF-Fog, we demonstrated how to model Dew Computing in the simulator in [124]. The concept of Dew Computing can be applied to various domains [74], 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. 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 behaviours, including service migration to mobile devices and temporary Internet access limitation.

Moving IoT devices may switch service providers at any time 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 [158]. 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 our work a Markov chain was used for proactive migration prediction, because it is a popular method for mobility estimation due to its efficiency and intuitiveness [52]. 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. In our prediction technique a weighted Markov model is used, that 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. Detailed information on the implemented proactive migration strategy can be read in [124].

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.

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 from 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.

To evaluate our latest simulator extension, we used an IoT-assisted weather forecasting scenario utilizing both fog nodes and dew services running on dew devices. In this scenario, drones fly and collect information about temperature, humidity, and wind in the atmosphere in the central European area[12]. The size of the whole area was around 850×230 km^2. At the beginning, the devices were located at the edge of the map (uniformly distributed) and were heading to opposite sides. Such devices were equipped with computing and storing units (8 CPU cores, 8 GB RAMs, 16 SD-card), and flied with an average speed of 70 km/h. To be as realistic as possible, we applied the cloud schema of the ELKH cloud of ELKH SZTAKI [79] to determine

[12]Droneblog.com (accessed in June, 2022): https://www.droneblog.com/how-drones-are-helping-with-weather-forecasting/

the CPU processing power, network operations, and storage capacities of physical machines for experiments.

To evaluate the proposed proactive service migration strategy, the following use cases were defined: (i) without motion prediction and (ii) with motion prediction using the two-order Markov model. In the first case, this meant that a service migration only occurred when the actual position of a device was not covered by any node (i.e., reactive migration); therefore, VMs were 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 hours, and the data sampling period was set to 1 min. A measurement was equivalent to 100 bytes of data generation.

The results of the evaluation scenario [124] showed that, when nodes overlapped each other, a device had multiple choices, and it could process the data locally or remotely. The 100 km long range ensured effective data offloading and device handover. This effectiveness also implied the fastest data processing, with a 1.35 min timeout. Due to the proactive service migration strategy, more VMs were used (i.e. 35 in the case of the 100 km long range). Due to the uncertainty of the proactive strategy, it dealt with the highest energy consumption of 25.89 kWh, and initialized 22,487 migrations. It could also be observed that without motion prediction, around 60% (203 MB) of the data – and with motion prediction, around 50% (121 MB) of the data – were processed by dew devices. These ratios could also be observed, when we take a look at how much time the devices spent processing data besides generating data, which were 405.68 and 243.7 min. Finally, considering the time needed to run the simulation, the average simulation time increased due to the matrix multiplications used by applying the Markov-model.

Summary of Chapter 3.

Contributions:

- *Markus and I designed a classification taxonomy for IoT-Fog-Cloud simulators, and categorized of existing simulators using it;*
- *I designed extensible models for IoT sensors, actuators, provider pricing schemes, and Fog and Dew Computing; Markus and I proposed the DISSECT-CF-IoT and DISSECT-CF-Fog simulators realizing these models, and evaluated them with various scenarios;*
- *I designed a methodology for IoT-Fog-Cloud application behaviour analysis; Markus and I proposed various resource allocation strategies following this methodology;*

Project involvement:

Duration	Project name	Type	Role
2016-2017	UNKP-16-4	Postdoc. scholarship	Leader
2017-2021	Internet of Living Things	GINOP	WP Leader
2022-2023	Smart Systems	TKP2021-NVA-09	RG Leader
2020-2025	CERCIRAS (CA19135)	EU COST Action	MC Member
2019-2022	FK-131793	OTKA	Leader

Resulting publications:

No.	Title	Venue	Rank	IF
1	A survey and taxonomy of sim. ... [118]	Elsevier SIMPAT	Q2	3.272
2	Investigating IoT Application ... [119]	Springer CCIS	Q4	-
3	What Does I(o)T Cost? [89]	ACM ICPE WS	-	-
4	Cost-aware iot extension ... [121]	MDPI FI	Q3	-
5	Location-aware task allocation ... [123]	IEEE PDP conf.	CORE C	-
6	Actuator behaviour modelling ... [122]	PeerJ CS	Q2	2.41
7	Modelling energy cons. ... [120]	CLOSER conf.	Best p.	-
8	Modeling dew computing ... [124]	MDPI AS	Q2	2.7

Investigating IoT - Blockchain-Fog-Cloud Systems 4

THE Blockchain (BC) technology was proposed by Nakamoto [135] in 2008 as a revolutionary concept in the fields of distributed trust, decentralized economy and security. It is a Distributed Ledger Technology (DLT) in the form of a distributed transactional database, secured by cryptography, and governed by a consensus mechanism [35]. This technology was first introduced as the backbone of the Bitcoin ecosystem in 2009 [40]. As BC gained a high reputation and attention among research and industry communities, it has proven robustness against the disadvantages of classical centralised systems. It is capable to provide trusted, immutable, and fully decentralised data management and reliable payment methodologies. These criteria could solve the remaining major challenges of IoT-Fog-Cloud systems, once suitably deployed.

In a very simple BC application scenario, an end-user sends a request to the BC network, which consists of BC nodes, to perform a defined transaction (TX). TXs may be data to be stored in the chain (i.e., payment data, reputation data, identity data, etc.), or they can be smart contracts (SC), whose results can be either saved in a centralised database (e.g. in the case of a cloud) or in a distributed manner (in cases of fogs or BC). Once the TX is performed, the majority of BC nodes should agree that it should be saved on the distributed ledger, and as a result added to the chain saved in all BC nodes. In the operation of a BC network, the saving of TXs is gathered by the so-called consensus algorithms (CA). Several approaches were proposed to reach consensus about the state of the chain among the participating nodes (also called miners). The most famous is the Proof-of-Work (PoW) CA, which was implemented in 2009 in Bitcoin [135], the first BC system, and has been used in other robust BC systems, such as Ethereum [183]. In PoW-based BCs, a BC node proves the validity of its generated block of data by coupling a puzzle solution within the block. The puzzle solution is generally characterised by the difficulty to obtain, while it can easily be validated once it is found. The puzzle is a mathematical problem that requires high computational power to be obtained. Although PoW methods have proven strong security and support for BC systems, they have some drawbacks, such as high energy consumption and high latency, that encouraged R&D communities to search for other trusted methods. The Proof-of-Stake (PoS) algorithm [105] was proposed a couple of years later, in order to solve the high energy consumption problem implied by PoW. However, PoS has some drawbacks, e.g. relatively low reliability [191]. In PoS-based BCs, the BC node that is allowed to generate the next block is chosen randomly by the system. To encourage the system to pick a specific BC node, staking more digital coins in deposit will increase the probability of being chosen. This provides high trust measures, as faulty generated blocks are not tolerated by the system, and the staked coins of the malicious/faulty BC node would be burnt as a penalty. Other approaches were proposed that provide enhanced trust in BCs, like the Proof-of-Elapsed-Time (PoET) [54], and the Proof-of-Authority (PoA) [16]. PoET-based BCs generate randomly selected times for BC nodes. The one node whose randomly picked time elapses first, will be the one to generate the next block. PoA, on the other hand, implies that only blocks signed by authorised members are validated and confirmed by the BC network. Those authorised nodes must be known trusted participants that can be tracked and penalised in case of faulty behaviour. Both of these CAs share the property of being suitable for private and permissioned BCs, while PoW and PoS are known to be suitable for public and permissionless BCs.

Though BC is the base technology behind crypto-currencies, yet it is implemented in a wide range of different applications. The security and reliability, along with the distributed trust management criteria provided by BC, excited the research community to integrate it with Fog Computing (FC), in a step towards reaching a distributed and trusted, data, payment, reputation, and identity management systems. The integration of FC and IoT includes various challenges, such as the security and efficiency of communications. The development of a successful IoT system is usually challenged by security and privacy issues, the need of efficient data management schemes, resource-constrained devices, energy consumption, and connectivity into long distances and periods of time [159]. The advent of IoT-Fog-Cloud systems solved some of these challenges by providing flexible data processing and storage services, leading to more than five billion devices connected over the Internet due to IoT [127]. However, the main challenges remained open even, when FC is integrated with IoT, such as the need for efficient data management schemes and enhanced privacy and security.

In this chapter, I summarise our main contributions in the field of integrating BC to IoT-Fog-Cloud systems. In Section 4.1, I summarise our literature review on BC integration approaches, and in Section 4.2 I introduce a simulation solution for BC-FC systems, and summarise our research results in analysing BC-Fog-Cloud systems and applications. In Section 4.3 I introduce how BC-FC integration can assist task scheduling decision making in clouds. Finally, in Section 4.4 I introduce our proposal for an application of BC-based digital credential verification.

4.1 The Role of Blockchains in IoT-Fog-Cloud Systems

BC applications have been proposed in a wide variety of environments such as distributed voting, eHealth, Mobile Computing, Internet of Vehicles, etc. We believe that integrating Blockchain technology with smart applications for managing data of mobile devices can further enhance the privacy and security requirements of current complex systems. In [30], we proposed a vision and research methodology for BC integration options in smart systems. We envisioned a BC-enabled simulation framework capable of analysing the integration possibilities with fog/edge and cloud infrastructures at different layers of smart systems. Such a framework could be able to model and analyse the behaviour of BC networks in large-scale fog-enhanced smart systems, while using different Artificial Intelligence (AI) methods.

The first step of our research in proposing integrated BC-FC solutions was to perform a detailed literature review. In [25] we presented a survey of the state-of-the-art solutions and approaches in this field, and now I summarise our findings. Our goal was to highlight the roles BC played in IoT-Fog-Cloud systems, and to present how the research community envisions future BC-FC integration solutions.

Table 4.1: *Survey papers in FC-BC integration*

Ref.	Domain	Year	Aim
[150]	Econoour & Politics	2017	Compare Cryptocurrency to national currencies
[161]	CC-BC	2017	Compare Ethereum BC vs. Amazon SWF
[139]	IoT-BC	2018	State-of-the-art Assessment
[179]	FC-BC	2018	Compare Golem, iExec and SONM
[159]	IoT-BC	2018	State-of-the-art Assessment
[65]	Smart environments	2019	Assessment of Smart IoT-BC
[175]	IoT-FC	2019	Security and Privacy
[66]	IoT-FC-BC	2019	Cryptography assisment
[84]	SIoV	2019	Trust factors, challenges, models, and vision
[43]	eHealth-BC	2020	State-of-the-art Identity management systems
[20]	IoT-FC-BC	2020	General concepts

Table 4.1 summarises the corresponding surveys found in the literature in 2020 [25]. A conceptual research surveying the criteria needed to develop a cryptocurrency system that integrates neuron technologies, artificial intelligence, BC, and FC was presented in [150]. It mainly focused on the economical aspects rather than technical details, in a step towards understanding the threats, challenges, benefits,

and expectations of replacing national currencies with cryptocurrencies. A comparison of the cost of computation and storage was performed by using Ethereum BC and Amazon SWF was conducted in [161]. Accordingly, the average cost of executing the same process instance on Ethereum was reported to be two orders of magnitude higher than on Amazon SWF. Uriarte and De Nicola [179] technically surveyed three ongoing Fog-based BC projects, namely Golem, iExec and SONM. The survey concluded that even for those three most mature FC-Based BC solutions, they still lack standardisation since they are mainly based on ad-hoc communications. The three solutions use Ethereum platform with different properties. The PoCot [187] algorithm was used in IExec, while a security deposit is made by providers, just like in the PoS [33]. Reyna et al. [159] extensively surveyed the integration of IoT and BC, by highlighting the challenges of BC technology, and investigating the integration ways of IoT and BC. They stated BC as a key factor when deployed in IoT systems, where information need to be securely shared among participants. They also highlighted the challenges and open issues in BC-IoT integration, such as the security, privacy, resource and scalability limitations. Fernandez-Carames and Fraga-Lamas [65] provided a comprehensive study on approaches of smart campuses and universities. They highlighted the main features, communications architectures, and BC potential applications. According to their finding, using traditional database systems provides more efficient latency and energy consumption than using BC, hence, BC deployment is not always the best choice. Tariq et al. [175] surveyed potential security and privacy challenges in fog-enabled IoT systems, while they briefly discussed how BC may enhance such systems. George and Sankaranarayanan [66] investigated light weight cryptographic solutions that might be suitable for IoT-FC-BC systems. Trust management models for the Social Internet of Vehicles (SIoV) were surveyed and discussed in [84], where the authors analysed the trust factors in such systems, such as the reputation, the environment, system expectations and goals. This survey also reviewed existing trust models, and trending solutions to solve the challenges faced by such models, e.g. how BC and FC can boost the development of trusted SIoV systems. Bouras et al. [43] surveyed decentralised BC-based identity management systems, and the possible scenarios of adopting such systems to improve health-care applications. Alli and Fahadi [20] presented the basic concepts of IoT, FC, BC, the FC-BC general deployment frameworks, opportunities, and challenges. They clarified how the decentralisation property of BC can be applied at the device level, the fog level, or the cloud level, and briefly discussed some deployable CAs.

All surveys approved the advantages of the BC-FC integration, which include enhanced security, integrity, reliability, fault-tolerance, and credibility, thanks to the distribution of processing units of IoT and FC, and the decentralisation and trust management mechanisms deployed within the BC mechanisms. On the other hand, this combination of different technologies raised challenges, such as privacy, latency, legalisation, and standardisation issues.

In addition to reviewing these surveys, we identified 43 corresponding articles that proposed systems benefiting from BC-FC integration, or addressed different challenges faced by such an integration in [25]. We found that most of the papers discuss solutions for IoT-FC-BC integration. This may be due to the fact that FC was initially introduced to specifically increase and enhance IoT applications. However, we found that other papers discuss FC-BC integration when deployed in different environments, such as Smart Mobile Devices (SMDs), Internet of Vehicles (IoV), Industrial Internet of Things (IIoT), and eHealth.

Our key observations based on this throughout review were the following. The vast majority of BC-FC integration approaches used Proof-based algorithms. Most of these deployed a variation of PoW-based CA, despite the fact that PoW-based BCs are the highest energy consuming ones, compared to other algorithms. Although the BC deployment has several advantages, there were several challenges faced by researchers and developers of different BC-FC integrated solutions. These challenges include the lack of standardised methods, informally guaranteed privacy, higher latency compared to centralised solutions, higher energy consumption, and social mistrust of BC-based solutions due to the technology being juvenile. Additionally, some applications in the IoV and the eHealth domains required highly adaptive mobility controls, due to the continuous movement of clients, which had usually been solved by deploying FC mechanisms. Specifically, some articles approached some enhancement of mobility handling while deploying BC, yet this negatively affected other criteria, like latency and privacy. Finally, since crypto-currency concepts are still not accepted nor legalised in many countries around the world, BC technology is ignorantly illegal.

4.2 Modelling and Simulating BC-Fog-Cloud systems

In light of the general tendency towards scepticism around BC systems being reliable, huge research and industrial projects are being encouraged to address issues and vulnerabilities of those systems. This is because it is believed that a successful BC deployment would definitely advance the Internet-of-Everything (IoE) applications. Dubai, for example, has been planning to become the first smart city powered by BC [168]. In 2017, the Liberstad project was launched to establish a private smart city in Norway that adopts City Coin as its official currency. In 2019, China had launched a BC-based smart city ID system [69], planning to have its own official digital currency [169].

The major service providers, such as Linux, IBM, Google, and Microsoft, are also interested in incorporating these technologies to their portfolio. To experiment with complex systems benefiting from this integration before the deployment phase, suitable and accurate simulation environments are needed. As we have seen in the previous chapters, the use of simulators can save significant costs and efforts for researchers and companies aiming at adopting these emerging technologies. As we will see in this section, the available simulation environments in 2020 facilitate FC or BC simulation, but not both.

The research and industry communities have been working hand-in-hand to solve the major challenges discussed earlier, along with other technical issues. Such efforts require reliable and flexible simulation environments that can mimic real-life scenarios at the lowest possible costs. Simulation tools that were initially implemented in the past for classical Peer-to-Peer (P2P) networks, such as PeerSim [131], may not be able to cover all the mechanisms of a modern BC system. Although some recently proposed systems use PeerSim, such as [141], it required a large amount of changes, modifications, and additions to redesign it into a BC simulation tool.

We performed an extensive literature review on FC and BC simulation approaches in [26], here I summarise its results. We found several simulation tools that mimic FC-enhanced cloud systems, IoT-Fog-Cloud scenarios, and some tools that mimic BC scenarios, each with specific constraints on the used CAs. In approaches targeting FC-BC integration applications, general-purpose FC simulators are typically used, where the BC is only implemented as an application case, such as in [107]. The results of such simulation approach can be trusted valid for limited cases, such as providing a proof-of-concept proposal. However, critical issues, such as scalability and heterogeneity in huge networks, need to be simulated in more specialised simulation environments. To mention one critical case, the BC protocols, deployed in different CAs, require a more precise and accurate deployment of the BC entities. Additionally, interoperation in different layers of a FC-enhanced IoT-Cloud system is rather critical. As some simulation scenarios need an event-driven implementation, others need a data-driven implementation, therefore, scenario outputs may differ when simulated using different simulation environments. Such a possibility of fluctuated simulation outputs should normally lead to unreliable simulation results.

Anilkumar et al. [21] have compared different available simulation platforms specifically mimicking the Ethereum BC, namely Remix Ethereum, Truffle Suite, Mist, and Geth. The comparison included some guidelines and properties, such as initialisation and ease of deployment. The authors concluded that truffle suite is ideal for testing and development, Remix is ideal for compilation and error detection and correction, while Mist and Geth are relatively easy to deploy. Alharby and van Moorsel [19] and Faria and Correia [64] proposed a somewhat limited simulation tool, namely BlockSim, implemented in Python, which specifically deploys the PoW algorithm to mimic the Bitcoin and Ethereum systems. Similarly, Wang et al. [184] proposed a simulation model to evaluate what is named Quality of Blockchain (QoB). The proposed model targeted only the PoW-based systems to evaluate the effect of changing different parameters of the simulated scenarios on the QoB. For example, average block size, number of TXs per day, the size of the memPool, etc. affecting the latency measurements. Furthermore, the authors identified five main characteristics that must be available in any BC simulation tool, namely the ability to scale through time, broadcast and multicast messages through the network, be event-driven, so that miners can act on received messages while working on other BC-related tasks, process messages in parallel, and handle concurrency issues. Gervais et al. [67] analysed some of the probable attacks and vulnerabilities of PoW-based BCs through emulating the conditions in such systems. They also classified the parameters affecting the emulation and presented a quantitative framework to objectively compare PoW-based BCs rather than providing a general-purpose simulation tool. Memon et al. [126] simulated the mining process in PoW-based BC using the Queuing Theory, aiming to provide statistics on those, and similar systems. Zhao et al. [192] simulated a BC system for specifically validating their proposed Proof-of-Generation (PoG) algorithm. Therefore, the objective of the implementation was to compare PoG with other CAs such as PoW and PoS. Another limited BC implementation was proposed by Piriou et al. in [148], where only the blocks appending and broadcasting aspects are considered. The tool was implemented using Python, and it aimed at performing Monte Carlo simulations to obtain probabilistic results on consistency and ability to discard double-spending attacks of BC protocols. In [57], the eVIBES simulation was presented, which is a configurable simulation framework for gaining empirical insight into the dynamic properties of PoW-based Ethereum BCs.

Table 4.2: *Blockchain simulation tools and their properties*

Ref.	PL	PoW	PoS	PoA	SC	DM	PM	IDM	FC
Faria and Correia [64]	Python	✓	χ	χ	✓	χ	✓	χ	χ
Wang et al. [184]	Python	✓	χ	χ	χ	χ	✓	χ	χ
Memon et al. [126]	Java	✓	χ	χ	✓	χ	χ	χ	χ
Zhao et al. [192]	Python	✓	✓	χ	χ	✓	χ	χ	χ
Piriou and Dumas [148]	Python	χ	χ	χ	χ	χ	✓	χ	χ
Deshpande et al. [57]	Java	✓	χ	χ	✓	χ	✓	χ	χ

Table 4.2 highlights the relevant properties BC-FC integrated solutions should have, and compares the most mature solutions in the field [26]. PL, DM, PM, IDM and FC are abbreviations for Programming Language, Data Management, Payment Management, Identity Management and Fog Computing support, respectively. As shown in the table, none of the previously proposed BC simulation tools made the PoA algorithm available for simulation scenarios, provided a suitable simulation environment for identity management applications, or, most importantly, facilitated the integration simulation of FC and BC technologies.

Many other works can be found in the literature, in which a part of a BC system, or a specific mechanism is implemented. The simulated part is only used to analyse a specific property in strict conditions, or to validate a proposed technique or mechanism under named (and probably biased) circumstances, such as in [185] and [157]. It is also worth mentioning here that some open-source BC projects are available and can be used to simulate BC scenarios. For example, the HyperLedger [177] projects administered by the Linux Foundation are highly sophisticated and well implemented BC systems. One can locally clone any project that suits the application needs and construct a local network. However, those projects are not targeting simulation purposes, rather provide realised BC services mostly for industrial projects. Additionally, most of these projects, such as Indy, are hard to re-configure and, if re-configured, very sensitive to small changes in their code.

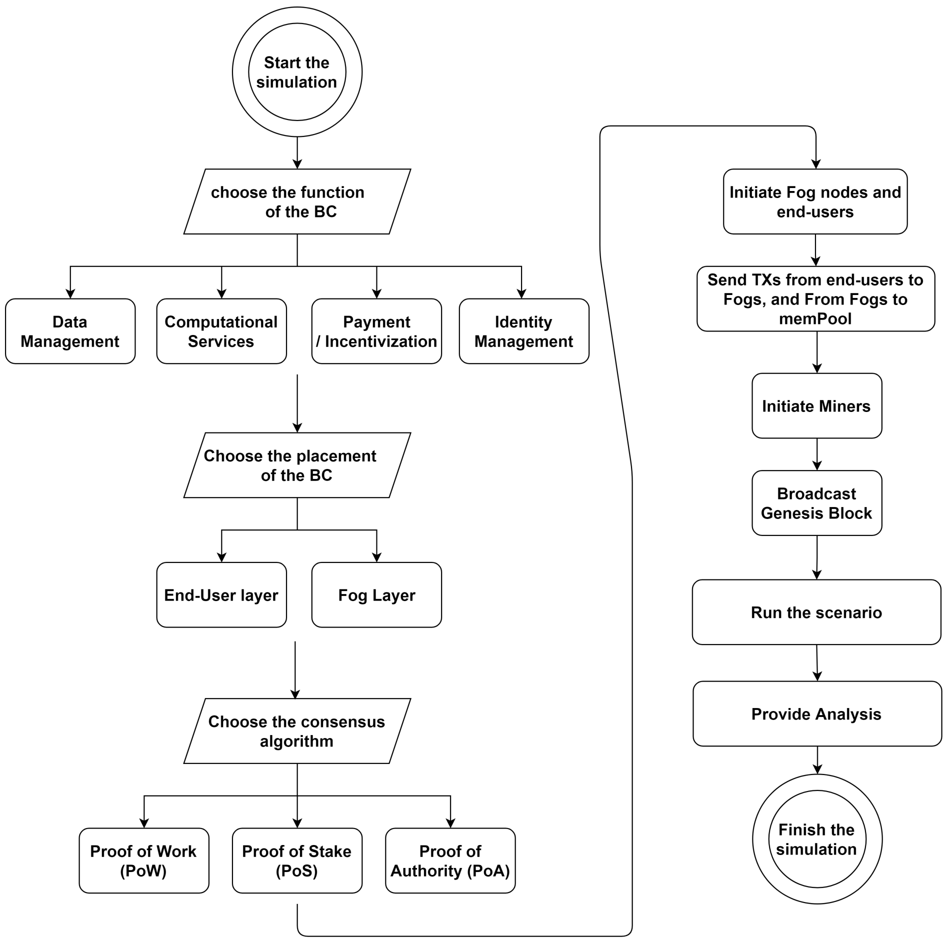


Figure 4.1: *Workflow of a simulation run using the proposed FoBSim tool*

4.2.1 FoBSim

In [26], we proposed FoBSim, a BC-FC simulation environment that can simulate different integration scenarios of FC and BC technologies, filling the gaps identified in our literature survey summarised in the beginning of this section. FoBSim implements different CAs, namely PoW, PoS, and PoA that are ready to be deployed in any scenario. It facilitates the deployment of BC miners in the fog or end-user layer. It allows different services to be reliably provided by the BC network, namely Data Management, Identity Management, Computational Services (through SCs), and Payment/Currency transfer Services. It provides both parallel and non-parallel execution for mining processes. Enhanced communication among the network nodes is supported by gossiping, which can be optionally selected, leading to greater consistency in different possible network topologies. To the best of our knowledge, FoBSim was the first simulation environment released in 2020, whose primary goal was to mimic integration scenarios of FC and BC technologies.

In FoBSim, the fog layer can be configured according to the scenario that needs to be simulated. For example, the number of fog nodes, the communications within the fog layer and with other entities of the simulated system, and the services provided by the fog, can all be modified. The simulator was implemented using Python v3.8, and made available as an open source software. It can easily be extended by adding the needed classes and modules and, hence, cover necessary proposed scenario entities. Its architecture and conceptual workflow is shown in Figure 4.1 [26]. It was designed in line with the general architecture of a reliable BC simulation tool presented in [110]. As shown in the workflow, different CAs can be used, different services of the BC network can be declared, and different placement scenarios of the BC network can be chosen. When the BC network is located in the fog layer, the number of BC nodes does not need to be selected, because each fog node is also a BC node. Nevertheless, number of task requester end-users connected to each fog node needs to be added, while some fog nodes in a PoA-based scenario might not be authorised to mint new blocks. Once the network is built, the system model can be run and tested.

Network connectivity characteristics are a major and critical concern in any BC system. To facilitate a network architect's job, FoBSim allows to define the number of nodes in each layer, the number of neighbours of each BC node, and the general topology of the network. Additionally, all BC nodes are connected into one giant component by default, whether they were deployed in the fog layer or end-user layer. Consequently, the effect of manipulating the topology of simulated networks can be easily captured.

Concerning the operation of the simulator, the first block added to the chain in each simulation run is the most important block of the chain. Different scenarios imply different formats of this block, and different methods to broadcast it among, and be accepted by, miners. In the current version of FoBSim, a genesis block is initiated with a list of miners available when this block was generated. The block number, previous hash, and nonce are all set to 0. The timestamp of the genesis block indicates when the chain was launched, and hence all blocks shall have later timestamp values than this first timestamp.

Currently, there are three available CAs ready to be used in different simulation scenarios, which are not abstracted away, but actually executed during the simulations. The use of gossiping can be activated regardless of the selected CA. A Gossip Protocol [41] is usually deployed in P2P systems for maintaining the consistency of distributed data saved in decentralised networks. Specifically in BC systems, miners regularly, yet randomly, gossip with their neighbours about current versions of the chain, aiming to reach consensus finality as soon as possible. According to specific characteristics of the BC, locally saved chains are updated so that all confirmed chains will become equivalent after some delay [78]. The equivalency (that any BC system is seeking) is defined by the content similarity of the chains (i.e. TXs, hashes, etc.), and the order similarity of the confirmed blocks. Gossip protocols are usually fault tolerant as many failing nodes do not affect the protocol, and they can adapt to the dynamics of the network as several solutions have been proposed in the literature to manage nodes joining and leaving the network. If gossiping was activated in the simulation, each miner will gossip once a new block is received by default. A miner, who is gossiping, requests information about the longest chain, and adopts it if its contents were agreed on by the majority of the network. Additionally, if a miner receives a new valid block, and the resulting local chain was longer than the global chain, the miner updates the global chain instantly.

In a simplified scenario of a PoW-based BC in the simulator, miners collect TXs from the mempool (which is a shared queue in FoBSim), and accumulate them in blocks that they mint. Specifically, all available miners compete to produce the next block that will be added to the chain. The fastest miner producing the next block is the miner whose block is accepted by all other miners of the BC. Synchronously, all blocks that are being minted by other miners are withdrawn, and all TXs within are sent back to the mempool. To mimic this scenario in FoBSim, we deployed the multiprocessing package of Python and trigger all miners to work together on the next block. Each miner then works within an isolated core of the device on which the simulation is conducted. When a Miner receives a new block, it checks whether the hash of the block (in which the nonce or the puzzle solution is included) is in line with the acceptance condition. Furthermore, the receiver miner checks whether sender end-users have sufficient amount of digital coins to perform the TX (in the case of payment functionality). Any miner is authorised to produce a block and there is no miner verification required. More details on the inner workings of the simulator, and on the implementation of other CAs can be read in [26].

Following the validation and verification methods of the simulation models presented in [163], I presented the technologies and paradigms that exist within our proposed FoBSim simulation environment, and highlighted its novelty compared to other related works. Next, I show, how we validated it usability through various case studies. We performed detailed experiments, validations, and evaluations in the following papers: [26], [96] and [95]. In the next subsections, I present three use cases of these works. During performing all the experiments, we deployed FoBSim on the Google Cloud Platform, using a C2-standard-16 VM (clocked at 3.8 GHz, 16 vCPUs, 64 GB memory), running Ubuntu 20.04 LTS OS.

4.2.2 Comparing Time Consumption of PoW, PoS and PoA

When PoW, PoS and PoA are compared in terms of average block confirmation time, PoW is expected to present the highest rates. This is due to the mathematical puzzle that each miner needs to solve, in order to prove its eligibility to mine the next block. In PoS, the network algorithm randomly chooses the next miner, while it slightly prefers a miner with a higher amount of staked coins. Once a miner is chosen, all miners are informed about the generator of the next block and, thus, the miner needs to perform no tasks other than accumulating TXs in a new standard block. Other miners then accept the new block, if it was generated by the expected generator. The verification process takes nearly no time (assuming that the transmission delay between miners is negligible).

Using PoA, all authorised miners mine new blocks, verify newly mined blocks, and maintain the chain locally. Meanwhile, unauthorised nodes verify new blocks and maintain the chain, but do not mine new blocks [56]. Every BC node has a regularly updated list of authorised miners. This implies that the more authorised entities, the more complex the verification on the receiver side can be. Consequently, it is recommended to reduce the number of authorised miners to decrease the complexity of verification [37]. Meanwhile, the more maintainers in a PoA-based BC, the higher the overall security level of the system.

Table 4.3: *Simulation parameters configuration for Case 1*

Simulation parameter\CA	PoW	PoS	PoA
no. of miners	5–500	5–500	5–500
neighbours per miner	4	4	4
puzzle difficulty	5–20	–	–
Authorized miners	All	Random choice	2–25
Initial wallet	–	1000	–
BC functionality	Data Management	Data Management	Data Management
BC deployment	end-user layer	end-user layer	end-user layer

To exemplify the use of FoBSim, we performed several experiments deploying different CAs under similar conditions. The simulation runs specifically focused on the measurement of the average time consumed by each CA, from the moment when a miner is triggered to mine a new block, until this mined block is confirmed by at least 51% of other BC miners.

As shown in Table 4.3 [26], we changed the difficulty of the puzzle during the PoW-based BC simulation runs from an easy level (i.e. 5), to a harder level (10), and finally to very hard levels (15) and (20). During the runs where PoA was used, we changed the number of authorised miners from 2/5 (2 authorised out of a total of 5 miners), 5/10, 10/20, and 25 authorised miners for the rest of the runs. We set the Data Management functionality, and stabilised the total number of TXs delivered to the mempool unchanged, which gives equivalent input for all simulation runs. However, we changed the number of TXs generated by each end-user to be equal to the number of miners in each run. Concerning the runs where a PoS is deployed, miner nodes were initiated with a wallet that has 1000 coins, allowing miners to stake random amounts of coins (with a block reward of 5 coins). We placed the BC in the end-user layer for all runs.

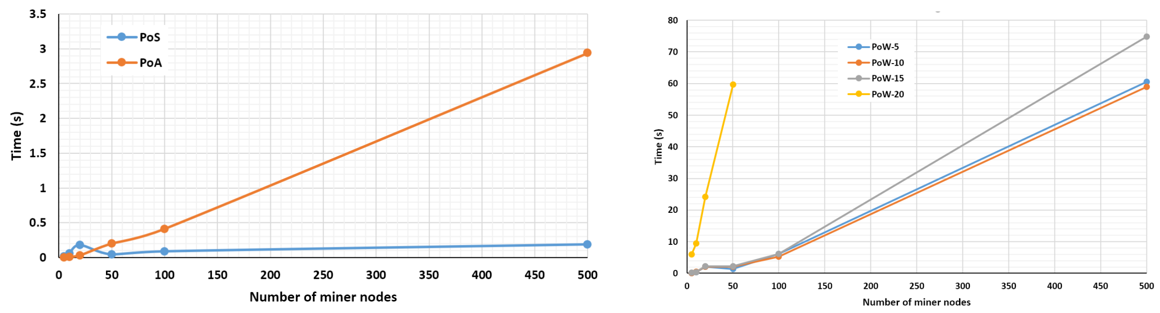


Figure 4.2: *Average block confirmation time for: (a) PoS-based vs. PoA-based BC; and (b) PoW-based BC with difficulty 5, 10, 15, and 20*

Some of the results obtained are depicted in Figures 4.2.a and 4.2.b [26]. We can see from the results that PoW-based BCs consume much more time to confirm a block, than PoA and PoS-based BCs, as expected. Additionally, the average block confirmation time, in PoW-based and PoA-based BCs, seems to be directly proportional to the BC network size, which complies with the results presented in related works, such as [130]. Comparatively, an average block confirmation time in a PoS-based BC seems unaffected by the network size, which complies with the results presented in [50].

4.2.3 Analysing Distributed Ledger Consistency

A major component of a BC-based system is the Distributed Ledger (DL), whose consistency is a problem that describes the unreliability of DLs in dense and highly dynamic networks [51]. This problem concerns maintaining exact copies of the DL, as the appearance of different DL versions is generally expected in realised scenarios. Reasons for such issue include both the transmission delay between network entities, and the continuous and concurrent alteration of DL data. The concept of finality is usually related to the DL consistency, which is the state of the BC, under which TXs cannot be cancelled, reversed or changed by any member of this network under any circumstances [32]. Although Nakamoto's model did not perfectly solve the consistency problem, it proposed a highly accurate, probabilistic solution. Specifically, the next block of data is introduced into the network, when its previous block had, most likely, sufficient time to be confirmed on the DL (i.e., synchronised between the majority of network entities). The occurrence of two different data blocks at the same time would mostly lead to a temporary inconsistent state of the DL, which is termed forking [85].

The enforced delay between blocks depends mainly on the ability of the system entities to find a puzzle solution. The difficulty of the puzzle is indeed updated through time according to the design requirements (e.g. Bitcoin's predefined delay is 10 mins). Such a model is classified as a Probabilistic Finality System. That is, the system continuously reduces the probability of concurrent DL updates, but the probability never reaches zero. Examples of algorithms belonging to this category are the PoW and the PoET. The other category is the Absolute Finality System, where the system allows its entities to produce the next block, only when the previous block is confirmed. Examples of algorithms belonging to this class include the PBFT and some versions of PoS [86].

The mechanisms deployed in BCs to decrease the probability of forking can be concluded by three main approaches. First, a gossiping protocol can be used to synchronise confirmed blocks and longer DLs. Second, the puzzle difficulty can be continuously incremented, which gives a window to network entities to gossip, leading to increased total energy consumption of the system and/or decreased total throughput. Third, the use of full and light nodes in the network can also be a solution [39], so that gossiping is performed by fewer network entities. Consequently, data propagation through the network shall consume less time.

In [96], we analysed the forking phenomenon in BC-based systems using a probabilistic finality mechanism. We defined various experiments in FoBSim, and executed them by using the PoW algorithm to represent a probabilistic finality system. As we performed our analysis under different conditions, we evaluated the consistency of the DL, by finding the ratio of forks appearing within the DL to the maximum possible number of chain versions. Here, we present the main result achieved and summarise the lessons learnt.

By deactivating the gossiping functionality in FoBSim, we could detect the appearance of a fork during a simulation run. That is, more forks appearing in the

Table 4.4: *Scenarios for observing the forking effect of BC parameter changes*

Scenario	Oscillated factor	run-1	run-2	run-3	run-4	run-5	Average δ	Effect
	$M = 100$	1	1	1	1	3	**1.4**	
1	$M = 500$	5	1	4	2	2	**2.8**	↑
	$M = 1000$	37	11	4	2	9	**12.6**	↑
	$M = 1500$	26	57	54	28	24	**37.8**	↑
	$N = 2$	91	105	78	69	91	**86.8**	
	$N = 3$	87	66	42	65	79	**67.8**	↓
2	$N = 5$	45	50	53	65	64	**55.4**	↓
	$N = 8$	117	73	71	45	71	**75.4**	↑
	$N = 15$	417	418	409	374	413	**406.2**	↑
	$\Omega = 5$	138	125	142	134	144	**136.6**	
	$\Omega = 10$	143	123	128	126	142	**132.4**	↓
3	$\Omega = 15$	129	135	125	140	136	**133**	↑
	$\Omega = 20$	22	14	20	9	31	**19.2**	↓
	$\Omega = 25$	1	1	1	8	1	**2.4**	↓
	β=2	3	3	3	3	3	**3**	
	β=3	4	3	4	4	3	**3.6**	↑
4	β=5	66	72	42	52	89	**64.2**	↑
	β=8	38	51	39	62	58	**49.6**	↓
	β=12	393	376	389	405	379	**388.4**	↑
	β=18	459	461	469	466	460	**463**	↑
	$\tau = 0$ ms.	2	5	2	1	17	**5.4**	
5	$\tau = 5$ ms.	26	47	7	24	6	**22**	↑
	$\tau = 10$ ms.	21	46	2	6	40	**23**	↑
	$\tau = 15$ ms.	31	72	37	11	103	**50.8**	↑

ledger indicate higher levels of DL inconsistency under the simulated scenario conditions. Manipulating simulation conditions facilitates the analysis of the direct effect of those conditions on DL consistency. In each simulation scenario, we oscillated the configuration of one condition, and stabilised the others. We ran each simulation scenario five consequent times under the same conditions, and computed the average number of chain versions. The five parameters that were oscillated are the number of miners M, the number of neighbours per miner N, the puzzle difficulty Ω, the number of simultaneously mined blocks β, and the average Round Trip Time (RTT) between neighbours τ. We conducted experiments on the Google Cloud Platform using an E2-standard-32 VM instance (up to 3.8 GHz, 32 vCPUs, 128 GB memory) running a Ubuntu 20.10 OS. Table 4.4 shows the applied configuration parameters and the measured results for each simulation scenario [96]. The table also concludes the general observed effect of each factor oscillation on δ.

Concerning Scenario 1, the results showed that δ is proportional to the number of miner nodes participating in the BC network. For Scenario 2, the table shows that δ is inversely proportional to the number of neighbours per miner, as long as the ratio $N : M \leq 1\%$. This result is similar to the results presented in [109], because the impact of N was studied for a maximum of four, and with a network size of 500. However, we found that for the case where the ratio $N : M > 1\%$, δ is directly proportional to N. This observation is somewhat consistent with the observations presented in [164]. That is, it was argued that $N < \log M$ may increase the number of forks in a given BC, due to several weak links acting as bottlenecks. Consequently, it was recommended that N shall be set to $\geq \frac{M-1}{M} \log M$. Taking the Bitcoin network as an example, the authors argued that it is safe, with regards to N, since it operates within a stable range of 22–99 connections (per full miner nodes). However, we argue that although such range is safe to guarantee partition tolerance, it is not optimised in terms of consistency. As [164] sets the recommended lower bound of N for safety, our results recommend an upper bound of N, for optimisation, to be $N \leq \lceil M/100 \rceil$.

For Scenario 3, simulation results showed that δ was inversely proportional to the puzzle solution difficulty. Such result is naturally expected for a system with a probabilistic finality. Increasing Ω is the BC solution to decrease the probability of $\beta > 1$. Additionally, the increment of Ω provides sufficient window for miners to gossip, and compensates for the continuous enhancement of mining machines. Such continuous enhancement (predicted by Moore's law [165]) may lead to faded effect of a static Ω through time. From another point of view, compensating for the advancement of computational capacities of mining machines only by increasing Ω implies higher energy consumption through time as discussed earlier. In Scenario 4, the δ was directly proportional to the number of blocks simultaneously mined and broadcast in the BC network. In the case where $M \leq 500$, simultaneous blocks appearing in the network are tolerated in terms of Y up to $\beta = 8$. Finally, the results for Scenario 5 showed that δ was directly proportional to the average transmission delay between neighbouring miners. These results conform with the proportionality characteristic presented in [112], where higher transmission delays predicted higher forking probability. Furthermore, it agrees with [81], where it was shown that lower delay between BC miners implies higher efficiency in terms of consistency.

The individual, relative effect of each analysed, oscillated factor can also be drawn from the results [96]. That is, increasing M can indeed decrease the ledger consistency level, yet it is not as effective as increasing the number of neighbours per miner (e.g. adding 500 miners to the network increases Y with about 1%, while changing the number of neighbours per miner from 8 to 15 increases Y with about 60%).

Optimizing Neighbour Selection in BC

I mentioned before that gossiping can be used to regain a consistent state of a BC faster, and to reduce the time needed for inter-miner communication during the operation of a BC. Based on the results we obtained in this section so far, we further proposed a Dynamic Optimized Neighbour Selection (DONS) protocol in [31], to enhance neighbour selection (NS) for the BC network nodes, which could further enhance these aims. More neighbours per miner, and higher delivery time rates between neighbours lead to lower levels of DL consistency [63, 164]. Our proposed DONS protocol requires the minimum number of neighbours per miner for a BC, by directing the miners to communicate with a globally optimised selection of neighbours. DONS decreases the number of cycles within a network path that shared data walks from any peer to any other peer (i.e. if there are no cycles, we get a Spanning Tree, which is an optimal solution [151]). In this work, we showed how DONS decreases the maximum time spent from generating data by any peer, till it reaches all the peers of the network. Additionally, we discussed how DONS can address the scalability issues of the network, leading to adaptive optimisation of NS, despite continuous change in network topology.

The proposed protocol includes a privacy-preserving leader election method, allowing one of the peers within the BC network to compute the Minimum Spanning Tree (MST) without previous knowledge of network peers identities (e.g. IP address). Using one of the famous MST algorithms (e.g. Prim's or Kruskal's), the leader computes the MST and broadcasts it to all the network. Every recipient of the MST then can read only its identity and its neighbours' identities, leading to each peer of the network communicating with the optimised selection of neighbours. We also evaluated the DONS protocol against other approaches, using two randomised network models, namely Erdos-Renyi model [60] and Barabasi-Albert model [155]. The DONS protocol was analytically evaluated in terms of security and privacy, and was experimentally evaluated in terms of propagation time and message overhead against the currently used RNS and Round Trip Time based Neighbour Selection (RTT-NS) methods. The leader election method used is theoretically and experimentally evaluated, in terms of time and message complexity [76], against a recent solution proposed in [140]. The evaluation results showed high levels of security, privacy and efficiency.

Overpowering PoW-based BC Networks

Upon the proposal of Bitcoin, several main security-related assumptions were made depending on the postulated high security of the utilised cryptography methods. As long as these main assumptions were not violated, the claimed high security of PoW-based BCs shall remain dependable. One of these assumptions is that PoW-based BCs will always be secure against a dishonest miner, as long as its computational power proportion is less than 50% of the total computational power of the network. Hence, controlling a minority of the computational power of a network by a dishonest party is claimed to be tolerated in Nakamoto's model. In our most recent work [28], we discussed probable alternative PoW mining methods, other than those presented in the original paper [135].

We reviewed several approaches that can be used to undermine or overpower PoW-based BCs. We discussed, how a dishonest miner can take over the network using improved Brute-forcing, AI-assisted mining, Quantum Computing, Sharding, Partial Pre-imaging, Selfish mining, among other approaches. As a result, we found that several practical approaches are available in the literature to undermine PoW-based BCs under some circumstances, even with minority. We showed that a function that generates random values, whose randomness level matches the randomness level of SHA-256, can be used to improve PoW mining. Machine Learning can also be used to conditionally enhance PoW mining, in terms of block finality, up to 26.6%. An attacker that controls 35.4% or more of the network can unfairly mine PoW-based Blockchains using our proposed method using Machine Learning.

4.2.4 Comparing BC Deployment Options for COVID Tracking

Following the outbreak of the COVID-19 virus in early 2020, a new pandemic has spread across the world, changing our lives. Emerging technologies have played an important role in providing various solutions to prevent virus spreading. Responding to the COVID-19 pandemic, smart applications [61] have started to be developed related to prevention of virus spreading or management of related societal problems, such as travel restrictions caused by the pandemic. The vast majority of these applications are mainly centralised and non-smart, which makes them carry single-point-of-failure, privacy, high latency, and legal issues, along with the lack of efficient handling of mobile devices [147]. In addition, the challenges in real-life scenarios include different healthcare institutions, different stakeholders within the supply-chain, heterogeneous networks, multicultural and highly distributed and dynamic system entities [108].

The adoption and mass acceptance of such COVID-19-related applications are greatly hindered by the general lack of trust associated with the nature of tracing apps, and the reluctance of people to share their personal data. To overcome this issue, we need to revise current solutions, and design methods addressing privacy-preserving, privacy-awareness, trust, explainability and interoperability [70].

To address this challenge, we envisioned a solution for gathering, storing, validating and analysing COVID-19-related data, including infections and vaccinations of citizens of a region. In this research, we proposed VACFOB [95], a general architecture for VACcination information validation and tracking with a FOg and cloud-based Blockchain system. In this vision, we merged FC and BC technologies to provide a privacy-aware and scalable approach for interoperable and effective vaccination information management. We derived three scenarios from real world vaccination reports, and used their requirements to evaluate the scalability of various BC-based systems with the FoBSim tool. The results can serve as recommendations for possible implementations of our proposal, which could contribute to a better and more efficient fight against COVID-19. Here, I summarise the most relevant findings of this study.

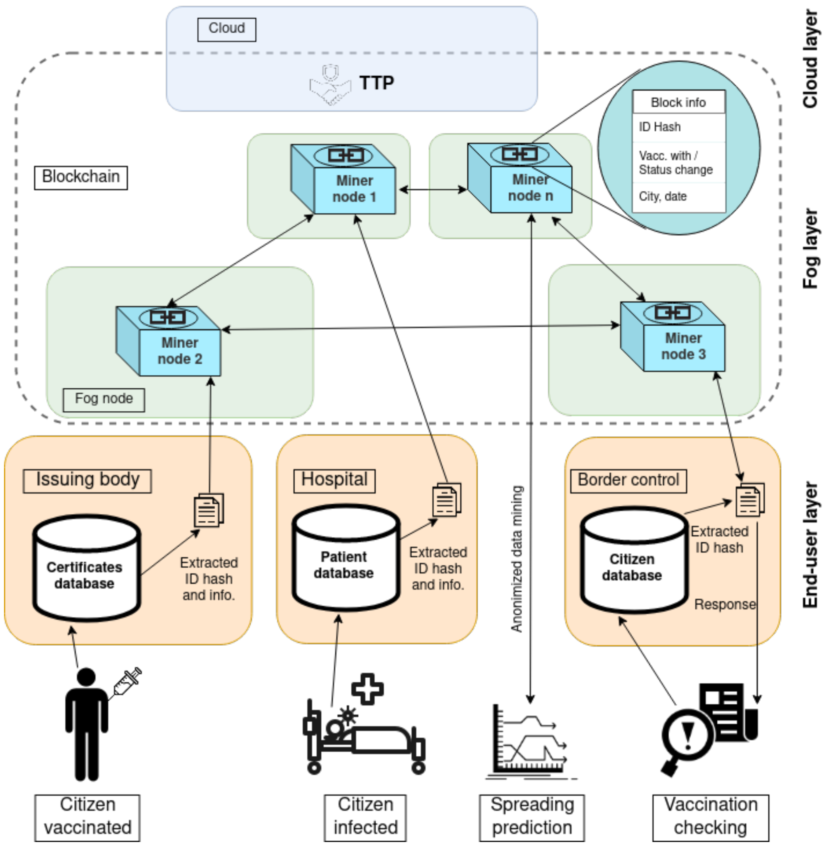


Figure 4.3: *The proposed VACFOB architecture for COVID-19 vaccination verification and immunity analysis*

The architecture of our VACFOB solution is depicted in Figure 4.3 [95]. As shown in the figure, the used BC infrastructure is installed in the fog and possibly backed up at the cloud layer. Fog nodes can host one or more miners and serve several COVID-related information providing bodies close to them. There are four different end-user types of the system (see end-user layer in Figure 4.3): (i) issuing bodies, who can issue vaccination certificates for citizens of a country; (ii) hospitals or private testing centers that are allowed (certified) to perform testings or to report health status of citizens; (iii) border control officers or other legal parties, who have to verify vaccination of citizens; and finally (iv) anyone, who wishes to make public queries for analysing a certain virus spreading. The means for this latest type may depend on the actual type of BC infrastructure utilised. In the first three cases, citizens should identify themselves, and the corresponding issuing and reporting bodies need to retrieve the ID hash, and append the status change for the BC transaction or query. In these processes the required private data are stored in a secure, centralised governmental database (i.e. off-chain).

In case we consider only vaccination verification as the sole role of the system, a public-permissioned BC would be suitable to allow anyone to validate vaccination of a citizen. To enable privacy and comply with GDPR, the issuing body should sign the vaccination or testing document with its digital certificate (stored off-chain), in this way any validator (who wish to validate a citizen vaccination) can perform the verification on the data queried from the BC. The border control entity performs such action in our figure. The extracted citizen ID will be the ID hash stored in a block, which is considered personal data, therefore, the collection and management of user consent must be ensured by the participating bodies (having permission to save new blocks) [77].

In case we would like to restrict access to the system, we may close verification for the public, and allow only specific bodies to perform queries, so the ID hash is not shared with the citizens and others through them. To this end, a Trusted Third Party (TTP) can be introduced and placed in a central cloud, who governs BC participation. In this case, digital signing of the documents is not necessary; any pseudonymised identifier in the form of a hash can be generated for all citizens (during first registration) and stored within the local private off-chain database together with other related personal and health data. From now on we consider this case, and suppose that only a certified body (permissioned) can get the citizen ID hash from the local private governmental or regional database, so not everybody can verify vaccination. Nevertheless, by allowing public queries for pseudonymised information can still support gathering statistics and predictions, e.g. on virus spreading or immunity success rates.

To allow vaccination verification, each transaction should contain the following properties (as shown in Figure 4.3): (i) a digitally signed document extract or a pseudonymised identifier of a citizen (ID hash), (ii) the actual status change (e.g. got vaccinated with some type of vaccine, or recovered/negatively tested), and (iii) a timestamp and location information (exact date and city). Note that this solution does not restrict its use to COVID-19 virus only; we can use it to handle simultaneously other types of virus infections/vaccinations as well (such as SARS, MERS or flu variants). In case we consider the border control use case for vaccination checking (as shown in the right bottom of Figure 4.3), the officer at the border first checks the personal ID of a citizen to pass. Then it enters the ID to the local governmental database to retrieve the corresponding hash ID, that should be used to make a query (look up a transaction) in the BC. If a transaction block is found with the ID hash containing vaccination information with a timestamp within the accepted range (e.g. 6 months), the citizen can pass without quarantine restriction. A concrete implementation of our VACFOB proposal could be analysed and evaluated according to different metrics, such as security and privacy, scalability, or operation cost. Concerning privacy-awareness, our proposal is GDPR-compliant, and by utilising the fog layer, its resources provide lower response times and better scalability, compared to a purely cloud setup.

The exact performance values of a future implementation will depend on the actual BC implementation we use. Therefore, we defined three different scenarios based on real world data with different scalability needs. In a nation-wide scenario, each BC miner in the fog can serve one or more end-user nodes that handle requests of certain bodies (concerning vaccination certificate issuing, updates, health state changes or information requests). In our experiments, we relied on information shared on the website of OurWorldInData.org [125], where we can find the statistics for 2020 on the coronavirus pandemic for more than 200 countries. To estimate the number of daily transactions for a region for our proposed system, we gathered information on daily performed vaccinations (Vacc.), new cases confirmed (Cases), and performed tests (Tests) for two randomly selected dates. From these data we could see that for a small country, such as Hungary (HUN) we may consider around 10 to 200 thousand daily transactions (status updates for citizens). For a big country, such as Germany (GER) we may expect around 1 to 3 million daily transactions, while for continents (Europe - EU, United States - US) they may go up to 5 million. Based on this, we investigated different BC systems that could be used for VACFOB implementations to serve different regions. We use the following scenarios with detailed parameter settings in FoBSim. Scenario 1: 100 thousand daily transactions for COVID-19 status updates for citizens; Scenario 2: 1 million daily transactions for COVID-19 status updates for citizens; Scenario 3: 10 million daily transactions for COVID-19 status updates for citizens.

We assumed that transactions are performed during working hours (e.g. eight hours per day), so we can roughly estimate up to 5 transaction per second (TPS) for Scenario 1, up to 50 TPS for Scenario 2, and finally, up to 500 TPS for Scenario 3. A concrete BC infrastructure can be characterised: by the number of fog nodes and its miners in the system; the maintained block size, which is proportional to the preset number of transactions per block (TPB); and the applied consensus algorithm. By investigating different parameter settings, we could analyse how to meet the abovementioned required TPS values.

The FoBSim simulation experiments using PoA and the PoS consensus algorithms were performed locally, on an Intel i5-8265U CPU (8 cores, 3.8 GHz, 12 GB memory) running a Windows-10 OS. The PoW experiments were run on an HP Synergy 480 Gen10 server node having 2 Intel Xeon Gold 5118 CPUs (2.3 GHz / 12-core, each) and 384 GB memory, running a Ubuntu-20.10 OS. We have fixed the number of transactions to be processed in all simulation runs to 10,000 transactions. We performed each simulation run five times, and took the average value for TPS as they marginally fluctuated.

Table 4.5: *Selected performance results for BC deployment options*

	Parameter settings						Results	
Sim. No.	No. of nodes	No. of miners	No. of neigh.	TPB	Delay (ms)	Cons. alg.	TPS	Target
9	10	10	2	100	15.2	PoA	238	✓
10	50	50	6	100	15.2	PoA	46	✗
17	10	10	2	100	63.4	PoA	129	✗
21	10	10	2	1000	63.4	PoA	1205	✓
1	10	10	2	100	15.2	PoS	206	✓
8	50	50	6	100	15.2	PoS	28	✗
16	10	10	2	1000	63.4	PoS	860	✓
17	50	50	6	1000	63.4	PoS	173	✗
1	100	100	10	100	63.4	PoW-10	263	✗
2	100	100	10	1000	63.4	PoW-10	1025	✓
3	100	100	10	100	63.4	PoW-15	246	✗
4	100	100	10	1000	63.4	PoW-15	599	✓

By determining the parameters to investigate certain BC infrastructures, we made the following restrictions. We varied the number of fog nodes and miners from 10 to 100, and the number of miner neighbours from 2 to 10. Concerning the block size in the BC system, Ethereum stores around 70 TPB, while Bitcoin stores around 2000 TPB on average; therefore, we decided to use 100 and 1000 TPB values for the experiments. To set the delay between neighbours in the fog layer we used the WonderNetwork service. We counted network latency between big cities corresponding to the scenarios defined before (HUN: Vienna-Budapest (7.4 ms), GER: Munich-Amsterdam (15.2 ms), EU: Warsaw and Porto (63.4 ms)). For simplicity, we kept these numbers constant even for a higher number of fog nodes as well. Concerning the settings of the consensus algorithms, we varied the difficulty of the puzzle during the PoW-based BC simulation runs by changing the hardness level from 10 to 20. This value basically represents the number of zeros at the beginning of the hashes to be mint. During the runs where PoA was used, we fixed the number of authorised miners to 3/5 (3 authorised out of a total of 5 miners). The measured TPS value results are shown in the eighth column of the tables, while we denoted in the ninth column, whether the experiment meets the required target threshold or not.

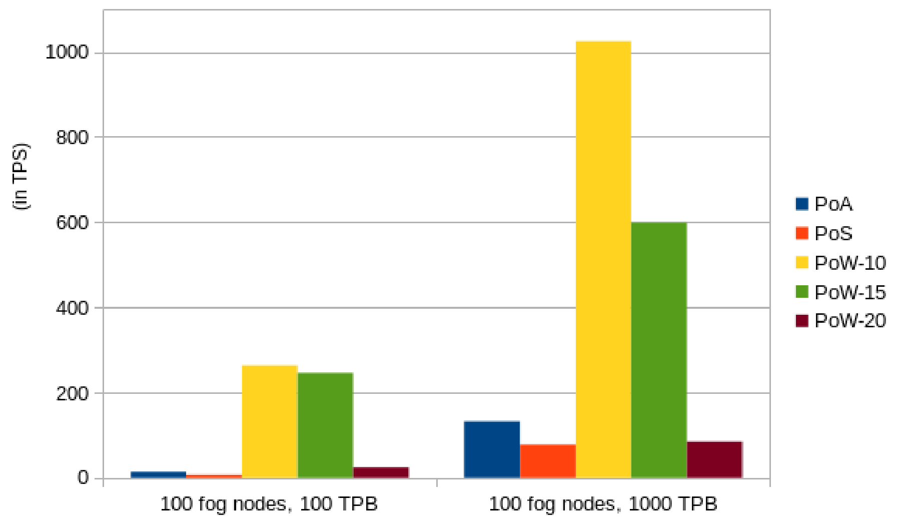


Figure 4.4: *Comparison of different consensus algorithms for Scenario 3*

First, we performed simulations with the PoA algorithm, then with the PoS algorithm, and finally, with the PoW algorithm. Simulation no. 1–8 aimed to cover the needs of Scenario 1 (with 5 TPS target value), no. 9–16 cover Scenario 2 (with 50 TPS), and no. 9–16 cover Scenario 3 (with 500 TPS). Selected evaluation results are summarised in Table 4.5 [95]. From this table we can see that by using PoA, simulation no. 10 and 17 failed to provide the required TPS values, while simulation no. 9 and 21 succeeded to provide the necessary performance. By using PoS, simulation no. 1 and 16 managed to meet the target TPS, but simulation no. 8 and 17 resulted in a failure to meet the expectation. Compared to the PoA algorithm, we can see that a BC using PoS can perform up to 50% less transactions within the same timeframe. Since simulations with PoW are generally very compute intensive, we selected only the largest parameter settings of Scenario 3 to be evaluated in the third set of experiments. To perform these experiments, we used an HP Synergy 480 Gen10 server node with 24 CPU cores. The results show that simulation no. 2 and 4 provided successful results concerning the target TPS.

Finally, we compared the performance of the utilised consensus algorithms. In Figure 4.4, we compared the performance of BCs using different CAs for Scenario 3 of our VACFOB proposal [95]. We can see that PoW with difficulty levels 10 and 15 can provide the best performance for the applied settings and requirements, but these variants can only meet the target TPS when 1000 TPB is applied.

In summary, we can state that determining the number of fog nodes and the number of transactions per block to be stored in the BC are crucial. For smaller scale systems (e.g. in Scenario 1) we can keep these numbers low, but in case we need to cover a larger region, it is inevitable to scale up the number of fog nodes, which implies that the TPB value needs to be raised as well to have the necessary performance. We believe that our systematic evaluation can provide a general overview on the behaviour of BC-based systems, and in case of a possible implementation of our VACFOB proposal, it can serve as a guideline to ease parameter selection.

4.3 Enhancing Task Scheduling in BC-Fog-Cloud Systems

We have seen in Chapter 1, task scheduling in Cloud Computing is challenging, since it is an NP-hard problem Ala'a Al-Shaikh et al. [18]. Taking a look at the literature, there are many scheduling algorithms addressing optimisation issues of task allocation to Virtual Machines (VM) or Virtual Resources (VR) in cloud datacenters [24]. Ant Colony Optimisation (ACO) [59] was introduced as a Computational Intelligence meta-heuristic technique for optimising a wide set of different problems, including task allocation. ACO takes inspiration from the social behaviour of some ant species, who deposit pheromone on the ground in order to mark some favourable path that should be followed by other members of the colony. In ACO, a number of artificial ants build solutions to the considered optimisation problem at hand, and exchange information on the quality of these solutions via a communication scheme that is reminiscent of the one adopted by real ants. Several works have proposed the use of ACO or new methods to improve its results, e.g. Hussein and Mousa [82] proposed two improved variants of ACO and Particle Swarm Optimisation for task scheduling.

With the advent of the BC technology, new possibilities arose to realise automated scheduling solutions to improve the effectiveness of data management in IoT-Fog-Cloud environments. In a research in this direction, we proposed an ACO algorithm in a fog-enabled BC-assisted scheduling model, called PF-BTS [29]. The protocol and algorithms of PF-BTS exploit BC miners for generating efficient assignment of tasks to be performed in cloud VRs using ACO, and award miner nodes for their contribution in generating the best schedule. It also allows the fog to process, manage, and perform tasks to enhance latency measurements. Meanwhile, the fog is enforced to respect the privacy of system components and to ensure that data, location, identity, and usage information are not exposed. We evaluated and compared PF-BTS performance, with a related BC-based task scheduling protocol, in a simulated environment. Our evaluation and experiments showed reliable privacy awareness of PF-BTS, along with noticeable enhancement in execution time and network load. Here, I present and summarise the main contributions of this work.

In terms of related work, Srikanth et al. [186] developed a Proof-of-Schedule

(PoSch) algorithm in their proposed BC-based scheduling approach to solve the same problem discussed before. In their proposed approach, task schedulers in the cloud layer are treated as BC miners, which were categorised into four different groups, and each group finds its optimal solution using a different algorithm. The algorithms used in the four groups are FCFS, Shortest-Job-First (SJF), Round-Robin (RR), and Hybrid Heuristic based on Genetic Algorithm (HSGA). Finally, their approach chooses the least time/energy consuming assignment. In another work, Srikanth et al. [178] used an ACO algorithm to achieve a feasible assignment of tasks to heterogeneous processors. In their research, it was also experimentally proven that optimising task scheduling using ACO guarantees better task assignment, than the results of the FCFS approach.

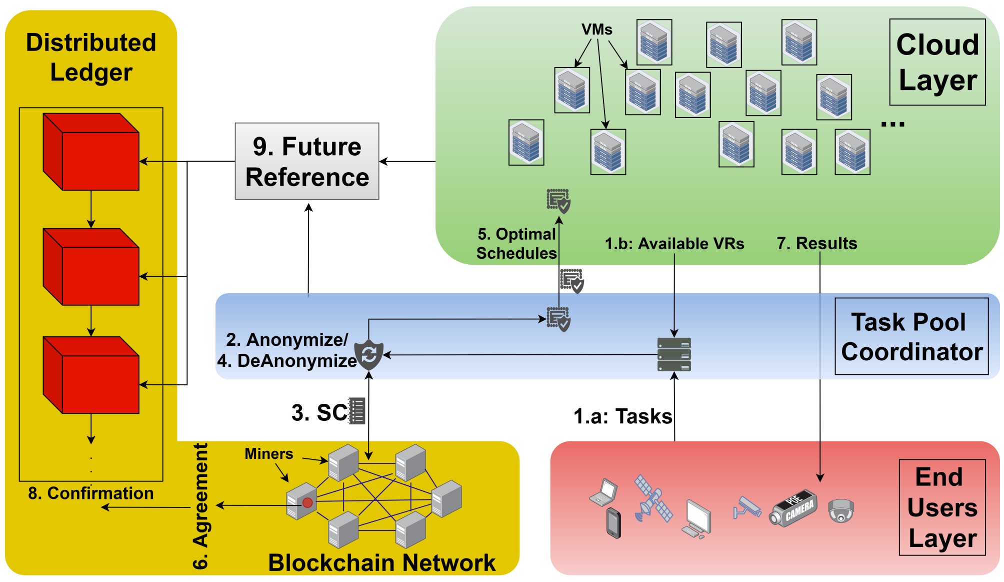


Figure 4.5: *The proposed BC-Fog-Cloud architecture for PF-BTS*

In our approach, we proposed exploiting BC miners to compute schedules with Smart Contracts (SC), who usually perform computational puzzle solving, to receive digital coins from the BC network. As shown in Figure 4.5, we deploy fog nodes to control the communication among end users, the BC network, and the cloud [29]. The Task Pool Coordinator (TPC) in a fog node of our proposed system is responsible for choosing the best schedule according to specific criteria, which reduces the latency compared to the case, where the BC network generates the best schedule by consensus. That is, a BC node picks the SC generated by the fog node, and sends its proposed schedule back to the fog node. The fog node then chooses the best schedule among the received ones and sends it to the cloud. The cloud accordingly performs the tasks using the assigned VRs, and returns the results to the fog node, which forwards them to end users. The direct incentive for BC miners who run the SCs is provided in the form of GAS, while incentives for generating new blocks are given by the BC network itself. Notably, the results are saved on the chain by BC peers consensus (e.g. PoW) for future reference and analysis. However, the time needed to reach a consensus, in order to save the data on the chain, is not included in the time required to generate the optimal schedule. Therefore, the type of CA deployed in the BC network does not affect latency. If some related information is needed, by the system administrators, the cloud, or the fog, it can be easily retrieved as the result of each SC is immutably saved on the chain.

To evaluate our proposed system, we adopted and simulated the BC scheduling system (BS) proposed in [186], which was also used to compare and validate our experimental results. We performed the experiments on eight VRs, five of them ran the ACO assignment optimisation, while the other three performed SJF, FCFS, and RR assignments. We excluded the fourth group studied in [186], since it presented the worst performance in most cases that were originally evaluated in the BS proposal. For our ACO assignment, we adopted a Multi-Objective ACO with Global Pheromone Evaluation, improved by a greedy optimal assignment approach to get even better solutions. Further technical details of the ACO version we created, its equations, the proposed improvement, and the final solution algorithm can be found in the corresponding paper in [29].

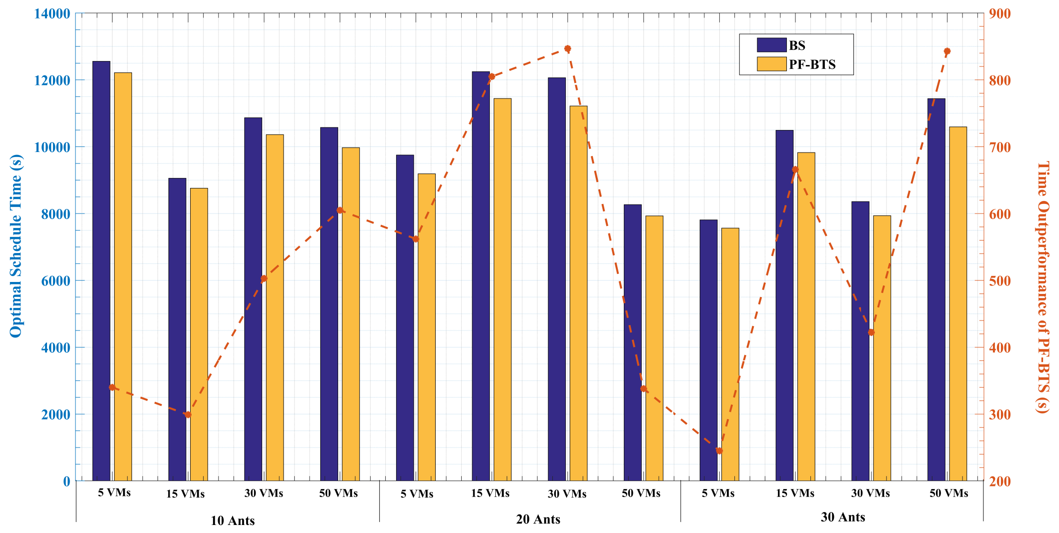


Figure 4.6: *Time consumption of BS and PF-BTS for 300 tasks*

Next, I present the experiment setup and some of the results. To evaluate our proposal, we implemented a simulation environment using Python. We ran it on Google Cloud Platform (GCP) using a C2-standard-8 (8 vCPUs, 3.8 GHz, 32 GB memory). We dedicated the first core for the FCFS computations, the second for the SJF computations, the third for the RR computations, and the remaining five cores were dedicated to the ACO computations. We conducted 33 simulation runs in groups of 12, 12, and 9 runs, for simulating 30, 300, and 3000 tasks, respectively, that had to be allocated to 5, 15, 30, and 50 VRs. In each simulation run, the same tasks' length and VRs' computing capacities were set in all generated SCs. Each VR was assigned a random computing capacity that ranged from 4 to 48 MFLOPS. Each task was assigned a random length that ranged from 100 to 1000 MFLO. The TPC was assigned a computing capacity of 20 MFLOPS. Accordingly, a maximum total execution time of generated tasks in TPC was configured to be 80 seconds. If this condition was met, the tasks were performed in the fog layer (i.e. the BC miners, the SCs, and the Cloud are not exploited). The evaporation factor was set to 0.3. One minute cost of VR was set to 0.1 cent/minute.

The experimental results for 300 tasks are presented in Figure 4.6 [29]. The time consumption of BS and PF-BTS (blue and yellow bars, respectively) is correlated with the primary y-axis on the left. The outperformance of PF-BTS over BS (dotted red line) is correlated with the secondary y-axis on the right. To further highlight the time needed from sending the tasks and VRs capacities to the BC network, until getting the schedules back from them, we conducted a simulation run, in which eight tasks were needed to be assigned into four VRs, and the assignment job was distributed to 16 BC miner nodes (16 SCs were generated) with each running 10 ants. The 16 miners sent their assignment suggestions to the TPC within 0.03 second. Obviously, this is a better result, than the one gained by BS (0.07 second), despite that there were 16 miner nodes involved instead of 4, and the schedules provided are optimal rather than random. The results showed that our proposed PF-BTS protocol outperformed several previously proposed approaches for solving similar problems, in terms of privacy, efficient scheduling of the VRs, and minimal exchanged messages. Moreover, PF-BTS allows the fog node, as an extension of the cloud, to perform the computational tasks at the edge of the network.

4.4 PriFoB: An Application for BC-FC Integration

The unprecedented pace of technological development in smart systems, incorporating sensing, actuation, and control functions, have the following properties and needs: (i) they are interconnected and need scalable, virtualized resources to run, store and process data, (ii) they are mobile and can potentially access and build on user data made available by smartphones and tablets, and (iii) they are getting smarter, so they may get access to user data provided by connected smart devices. As the number of smart devices in smart systems grows, the vast amount of data they produce requires high-performance computational and storage services for processing and analysis, among other novel techniques and methods that enhance these services and their management. BC applications have been proposed in a wide variety of environments such as distributed voting, eHealth, Mobile Computing, Internet of Vehicles, Self-Sovereign-Identity, etc. Integrating BC technology with such smart applications for managing data of mobile devices can further enhance the privacy and security requirements of current complex systems.

Specifically, trusted online credential management solutions are needed for instant and practical verification. Most of the available frameworks targeting this field violate the privacy of end-users or lack sufficient solutions in terms of security and QoS. To address this issue, we proposed a Privacy-aware FC-enhanced BC-based online credential solution, called PriFoB in [27]. This solution adopts a public permissioned BC model with different reliable encryption schemes, standardised Zero-Knowledge-Proofs (ZKPs) and Digital Signatures (DSs) within a FC-BC integrated framework, which is also GDPR compliant. We deployed both the PoA and the Signatures of Work (SoW) Consensus Algorithms (CAs) for efficient and secure handling of Verifiable Credentials (VCs) and global accreditation of VC issuers, respectively. Furthermore, we proposed a novel Three-Dimensional DAG-based model of the Distributed Ledger (3DDL), and developed a ready-to-deploy PriFoB implementation.

Here, I present the main architecture of PriFoB, and highlight its latency performance compared to related solutions in the literature [27]. Credential recognition is the process where a(n) (inter)national body, called Verifier, validates the legitimacy of a document that was issued by another body, also called as Issuer. A credential is issued upon an event occurrence to certify that this event has indeed happened, such as educational credentials, vaccination certificates, governmental passports/IDs, etc. Within a country, area, or continent, one may find agreed-on regulations to recognise a named type of credentials for purposes like governmental treatment, hiring, travelling, etc. However, once a person/entity, who has been issued a legitimate credential, needs to approve it abroad, a painfully lengthy and costly process needs to be carried out. This is because credential documents generally include different types of stamps, proofs, identification numbers, and other data that have to be verified individually and carefully for each credential referring to distinct, centralised, locally maintained databases. As a result, no global credentialing standard is used by issuers and no constant way to prove different credentials is guaranteed. Generally, the more sensitive data in a credential, the more complicated and costly it is to be validated.

Several previous works approached BC-based solutions for realising digital credential verification. The recently launched BC-based credential verification projects, namely BlockCerts [14], OpenCerts [15], and trustED [34] were surveyed in [36], where the most mature and suitable consensus methods, BC architectures, and BC platforms were presented and discussed. As these systems save all credentials information, along with relevant personal identifiers of students on the immutable chain, they are non GDPR-compliant.

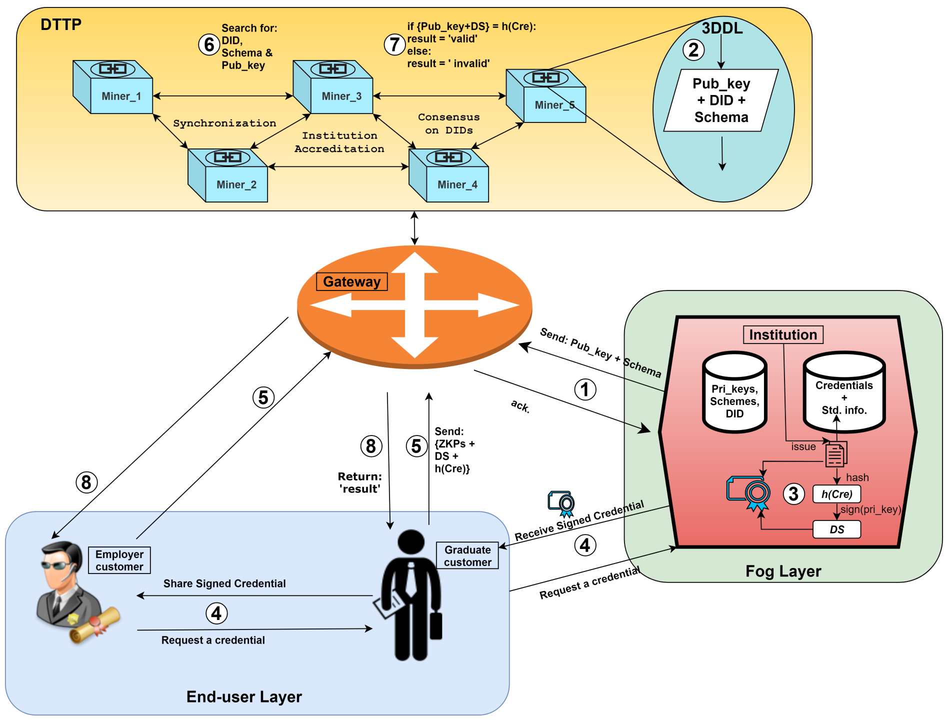


Figure 4.7: *The PriFoB architecture and framework*

The general architecture of PriFoB, and its simplified control flow scheme are depicted in Figure 4.7 [27]. The circled numbers indicates the order of steps to remotely accredit an issuer, publish new schemes, issue a new VC by an accredited issuer and verify that VC. PriFoB can simultaneously provide two major services: i) Institution Accreditation and ii) Credential Verification. The system is privacy-preserving by design, meaning that the deployed communication protocols and interaction/processing methods shall allow no window for private data leakage. PriFoB consists of three major layers: the DTTP layer (i), which consists of the Gateway (GW) and Miners. The GW connects the BC network with the issuers and end-users. Furthermore, it is responsible for bootstrapping new miners with randomly selected peers. Miners, on the other hand, are responsible for verifying new blocks and maintaining the consistency of the DL. Furthermore, miners are responsible for validating VCs using DSs and ZKPs. The End-user layer (ii), which consists of regular end-users requesting to be issued VCs. Those end-users can request, validate, share, or rename their issued VCs. Additionally, those end-users can download the whole or part of the BC. Entities belonging to this layer are not allowed to write on-chain. Finally, the Fog layer (iii), which includes the issuer(s). An issuer is an extended end-user entity, which is responsible for issuing new VCs to other qualifying end-users. To do so, an issuer is initially required to publish its unique Decentralised Identifier (DID) and a schema(s) (which is a VC template) into the DTTP through the GW. Once both are published, it can issue as many VCs as it needs.

As shown in Figure 4.7, the first step for an issuer to be able to issue new VCs is to be accredited. To do that, the issuer generates a public key and sends it to the DTTP along with non-private issuer information (e.g. official name, IP-address, etc.). The corresponding private key is then saved and managed locally by the issuer. The combination of data sent by an issuer to request accreditation is called a DID request. Once the request is accepted and the DID is published on-chain (i.e. as a DID block), we say the issuer is accredited and can issue VC schemes. Miners perform the SoW CA on DID blocks to maintain the DL consistency.

A schema is a VC template that is saved on-chain to refer to later when a VC is to be issued/verified. The second step is publishing a new schema upon issuer accreditation, which includes sending non-private information (e.g. schema title, public key, etc.). Miners perform the PoA CA on schema blocks to maintain DL consistency. Once the issuer is accredited and its schemes have been published, it can generate new VCs for its clients.

The third step of the PriFoB protocol, after publishing a schema, is issuing new VCs with its clients' private data (e.g. name, grade, birth-info, etc.), and perhaps with its own shareable private data as well (e.g. courses, professors' names, admin and registrar's signatures, etc.). Thus, this VC is only saved locally. An end-user (e.g. student or hospital patient, we also interchangeably use the terms client, customer, or agent) sends a VC request to the issuer, which also includes the client's public key. The request shall include some private identifying information (e.g. full name, SSN, year of credential issuing, etc.), so that the issuer can share a VC representing the original credential with high confidence that the requester is the client herself. The client connects with the DTTP to ask for the issuer's public key (which was initially saved on-chain in the first step of the protocol), and the mandatory data that needs to be submitted. Using this public key, the client can encrypt her VC request that includes the mandatory information. Consequently, no entity but the issuer can read the private data within the request, as only its private key can decipher the request. The issuer can then use the client's public key to encrypt its response. If a new type of VCs to be issued, a new schema needs to be submitted and, only after saved on-chain, the new type of VCs can be issued. In our proposed PriFoB system, we solve these issues by utilising a novel 3DDL.

Mainly, the issuer response shall include two parts (assuming that the requested credential was indeed issued): the digital credential, and a signature (Sig) (using the schema's private key). The encrypted response, which is, in fact, the VC, can only be then read by the client, as only her private key can decipher it. This is the fourth step of the protocol. Once the response is decrypted, the client is free to share and verify the obtained VC.

The client may send a verification request to the DTTP, as shown in the fifth step. The verification request includes only non-private data, including: the DID block identifier and index (e.g. issuer official name and its index on-chain), the schema block identifier and index (which might be similar for different issuers but unique for each issuer), the hash of the credential to be validated, and the signature originally provided by the issuer within the VC.

Once the BC receives the verification request, a miner, randomly selected by the GW according to the implemented load-balancing criteria, performs a VC verification (steps 6 and 7). The output of this step defines the response from the DTTP to the client. That is, the response is either Valid or Invalid, yet the reason for considering a VC invalid shall be also provided. Reasons for considering a VC invalid include: the DID or Schema has not been published on-chain, the hash of the credential is not equivalent to the decrypted Sig, or the VC has been revoked by the issuer. Note that no private data are saved on-chain, or provided for validation, because we use the ZKPs scheme to validate VCs without any knowledge requirement.

In our evaluation, we addressed the privacy-awareness of PriFoB, and compared real measurements of PriFoB deployed in the cloud, against well documented implementations of Ethereum and different Hyperledger platforms, including Indy, Besu and Fabric. Some of these works evaluated their solutions using simulation (all miner nodes run on a single machine), while others evaluated their solutions using real test-beds (each miner is allocated a different machine). We compared the latency of PriFoB with all of them to prove PriFoB outperformance, and we ran several test scenarios to comprehensively compare them with PriFoB. The performance of PriFoB was evaluated using a Proof of Concept (PoC) system composed of a single GW, a network of miners with different sizes (2, 4 and 6 miners), and a script we implemented that emulated several issuers and agents simultaneously communicating with the DTTP. We deployed each of those entities on a separate VM at the Google Cloud Platform. According to our evaluation results depicted in detail in [27], PriFoB outperformed all the compared related systems.

Summary of Chapter 4.

Contributions:

- *Baniata and I designed methods for modelling integrated Fog Computing and Blockchain systems, and proposed an open, extensible simulator called FoBSim;*
- *Baniata and I proposed a method for analysing Blockchain performance with consistency measurements, and evaluated it with FoBSim;*
- *I designed and proposed a vaccination information validation and tracking approach with a fog- and cloud-based Blockchain system;*
- *Baniata and I proposed a methodology for designing privacy-aware, Fog-enhanced and Blockchain-based applications;*

Project involvement:

Duration	Project name	Type	Role
2020-2023	MILAB	national	Participant
2022-2023	Smart Systems	TKP2021-NVA-09	RG Leader
2020-2025	CERCIRAS (CA19135)	EU COST Action	MC Member
2021-2022	TruBlo subgrant	EU H2020	Leader
2019-2022	FK-131793	OTKA	Leader

Resulting publications:

No.	Title	Venue	Rank	IF
1	A survey on blockchain-fog ... [25]	IEEE Access	Q1	3.367
2	Towards blockchain-based ... [30]	CEUR WS	-	-
3	FoBSim: an extensible ... [26]	PeerJ CS	Q2	2.41
4	Consistency analysis of ... [96]	EuroPar WS	-	-
5	Block the chain: Software ... [95]	IEEE Computer	Q2	2.2
6	DONS: Dynamic optimized ... [31]	Elsevier FGCS	D1	7.5
7	Approaches to Overpower ... [28]	IEEE Access	Q1	3.9
8	PF-BTS: A privacy-aware fog ... [29]	Elsevier IPM	D1	7.466
9	PriFoB: A privacy-aware fog ... [27]	Elsevier JNCA	D1	8.7

Summary of New Scientific Results

This dissertation presents a selection of my results all originating from a period after defending my PhD dissertation in the year 2011. The four theses are closely related to Chapters 1-4, and include the results, in which my contributions were essential.

Thesis 1: Cloud Federation Approaches

This thesis summarises our theoretical results concerning methods for creating Cloud Federations, and enhancing their operation performance:

1.1. I proposed a federated cloud management approach, and an integrated monitoring solution for Cloud Federations, which are able to perform efficient cloud selection for applications.

1.2. I proposed a classification of data protection issues in Cloud Federations and in IoT-Fog-Cloud systems that serves as a guideline for identifying legal responsibility.

1.3. I proposed and evaluated Pliant-based algorithms for energy-efficient datacenter management in a Cloud Federation. I implemented and evaluated the algorithms with an extension of the CloudSim simulator.

1.4. I proposed an SLA-based service virtualisation approach exploiting meta-brokering in a Cloud Federation to maintain service quality.

During the research works leading to these results, I performed international collaborations with outstanding researchers, among others: Ivona Brandic (Vienna University of Technology), Xavier Franch (Universitat Politecnica de Catalunya), Vlado Stankovski (University of Ljubljana). Related publications in Chapter 1.: [93], [94], [97], [98], [99], [101], [104], [106], [180], [181].

Thesis 2: Investigating IoT-Cloud systems

This thesis summarises our theoretical results concerning methods for analysing IoT-Cloud systems, and enhancing their operation performance:

2.1. Mishra and I analysed, evaluated and categorised message brokers using the MQTT protocol.

2.2. I designed an approach for semi-simulating IoT-Cloud systems. Pflanzner and I proposed and evaluated MobIoTSim, a mobile IoT device simulator based on this approach. The tool was implemented by Pflanzner, which is capable of generating and sending real sensor data over the network to cloud gateways.

2.3. Pflanzner and I proposed and evaluated a generic cloud gateway solution that can efficiently manage IoT devices by receiving, storing and processing or visualising their data. We also proposed a specialised IoLT gateway application on this approach, used for remote monitoring of plant growth. The gateway was implemented by Pflanzner.

During the research works leading to these results, I performed international collaborations with outstanding researchers, among others: Rob van der Mei (Vrije Universiteit Amsterdam) and Burkhard Stiller (University of Zurich). Related publications in Chapter 2.: [46], [102], [128], [129], [142], [143], [145], [146].

Thesis 3: IoT-Fog-Cloud systems

This thesis summarises our theoretical results concerning methods for analysing IoT-Fog-Cloud systems, and enhancing their operation performance:

3.1. Markus and I designed a classification taxonomy for IoT-Fog-Cloud simulators, and proposed an analysis and categorisation of existing simulators using it.

3.2. I designed extensible models for IoT sensors, actuators, and provider pricing schemes, and Fog and Dew Computing. Markus and I proposed the DISSECT-CF-IoT and DISSECT-CF-Fog simulators realising these models, and evaluated them with various scenarios. Markus implemented the simulators and the scenarios.

3.3. I designed a methodology for IoT-Fog-Cloud application behaviour analysis. Markus and I proposed various resource allocation strategies following this methodology. The proposed algorithms can be used to reduce application execution time and resource utilisation costs, and to optimise IoT-Fog-Cloud infrastructure management. We evaluated the algorithms in the DISSECT-CF-Fog simulator through various use cases. These algorithms were implemented by Markus.

During the research works leading to these results, I performed international collaborations with outstanding researchers, among others: Karolj Skala (Ruder Boskovic Institute). Related publications in Chapter 3.: [89], [118], [119], [120], [121], [122], [123], [124].

Thesis 4: IoT-BC-Fog-Cloud systems

This thesis summarises our theoretical results concerning methods for analysing IoT-BC-Fog-Cloud systems, and enhancing their operation performance:

4.1. Baniata and I designed methods for modelling integrated Fog Computing and Blockchain systems, and proposed an open, extensible simulator called FoBSim. We evaluated it by analysing various Blockchain deployment scenarios. The simulator was implemented by Baniata.

4.2. Baniata and I proposed a method for analysing Blockchain performance with consistency measurements, and evaluated it with FoBSim. Baniata and I proposed and evaluated an optimised data propagation protocol with enhanced neighbour selection. The corresponding algorithms were implemented by Baniata.

4.3. I designed and proposed a vaccination information validation and tracking approach with a fog- and cloud-based Blockchain system. I evaluated the approach in FoBSim by comparing different blockchain deployment options with use cases modelling real world data.

4.4. Baniata and I proposed a methodology for designing privacy-aware, Fog-enhanced and Blockchain-based applications, and evaluated it with a simulated fog-enabled Blockchain-assisted task scheduling application in clouds, and with a real privacy-aware institution accreditation and credential validation application. The applications were implemented and validated by Baniata.

During the research works leading to these results, I performed international collaborations with outstanding researchers, among others: Radu Prodan (University of Klagenfurt). Related publications in Chapter 4.: [25], [26], [27], [28], [29], [30], [31], [95], [96].

Bibliography

[1] MQTT Version 3.1.1. OASIS Standard. Edited by Andrew Banks and Rahul Gupta. http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html, 2014. Accessed: October 11, 2020.

[2] Eclipse Paho MQTT Library. https://www.eclipse.org/paho/, 2015. Accessed: August 4, 2020.

[3] Atomiton IoT Simulator. http://atomiton.com/simulator.html, 2017. Accessed: December 8, 2017.

[4] Beecham Research, Smart Farming Sales Brochure. http://www.beechamresearch.com, 2017. Accessed: March 19, 2017.

[5] IBM Watson IoT Platform visualization application. https://github.com/ibm-watson-iot/rickshaw4iot, 2017. Accessed: March 30, 2017.

[6] IBM IoT Foundation message format. https://docs.internetofthings.ibmcloud.com/, 2017. Accessed: March 19, 2017.

[7] OpenWeatherMap. http://www.openweathermap.org, 2017. Accessed: March 19, 2017.

[8] QualNet communications simulation platform. http://web.scalable-networks.com/content/qualnet, 2017. Accessed: March 8, 2017.

[9] SimpleIoTSimulator, SimpleSoft. https://www.smplsft.com/SimpleIoTSimulator.html, 2017. Accessed: December 8, 2017.

[10] Boson NetSim network simulator. http://www.boson.com/netsim-cisco-network-simulator, 2017. Accessed: March 5, 2017.

[11] Open Data documents of the city of Surrey. http://data.surrey.ca/dataset/open-data-documents, 2018. Accessed: 2018-02-08.

[12] MongoDB website. https://www.mongodb.com/what-is-mongodb, 2019. Accessed: June 30, 2019.

[13] The Scrapy tool. https://scrapy.org/, 2019. Accessed: May 12, 2019.

[14] Blockcerts webpage. https://www.blockcerts.org/, 2021. Accessed: December 3, 2021.

[15] Opencerts webpage. https://opencerts.io/, 2021. Accessed: December 4, 2021.

[16] A. A Avasthi and A. Saxena. Two hop blockchain model: Resonating between proof of work (pow) and proof of authority (poa). *International Journal of Information Systems & Management Science*, 1(1), 2018.

[17] Y. Ai, M. Peng, and K. Zhang. Edge computing technologies for internet of things: a primer. *Digital Communications and Networks*, 4(2):77–86, 2018.

[18] H. K. Ala'a Al-Shaikh, A. Sharieh, and A. Sleit. Resource utilization in cloud computing as an optimization problem. *Resource*, 7(6), 2016.

[19] M. Alharby and A. van Moorsel. Blocksim: a simulation framework for blockchain systems. *ACM SIGMETRICS Performance Evaluation Review*, 46(3): 135–138, 2019.

[20] A. A. Alli and M. Fahadi. Chapter four blockchain and fog computing: Fog-blockchain concept, opportunities, and challenges. *Blockchain in Data Analytics*, page 75, 2020.

[21] V. Anilkumar, J. A. Joji, A. Afzal, and R. Sheik. Blockchain simulation and development platforms: Survey, issues and challenges. In *2019 International Conference on Intelligent Computing and Control Systems (ICCS)*, pages 935–939. IEEE, 2019.

[22] 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.

[23] E. Baccarelli, P. G. V. Naranjo, M. Scarpiniti, M. Shojafar, and J. H. Abawajy. Fog of everything: Energy-efficient networked computing architectures, research challenges, and a case study. *IEEE Access*, pages 9882–9910, 2017.

[24] G. Bacso, T. Kis, A. Visegradi, A. **Kertesz**, and Z. Nemeth. A set of successive job allocation models in distributed computing infrastructures. *Journal of Grid Computing*, 14(2):347–358, 2016.

[25] H. Baniata and A. **Kertesz**. A survey on blockchain-fog integration approaches. *IEEE Access*, 8:102657–102668, 2020.

[26] H. Baniata and A. **Kertesz**. FoBSim: an extensible open-source simulation tool for integrated fog-blockchain systems. *PeerJ Computer Science*, 7:e431, 2021.

[27] H. Baniata and A. **Kertesz**. PriFoB: A privacy-aware fog-enhanced blockchain-based system for global accreditation and credential verification. *Journal of Network and Computer Applications*, 205:103440, 2022.

[28] H. Baniata and A. **Kertesz**. Approaches to overpower proof-of-work blockchains despite minority. *IEEE Access*, 11:2952–2967, 2023.

[29] H. Baniata, A. Anaqreh, and A. **Kertesz**. PF-BTS: A privacy-aware fog-enhanced blockchain-assisted task scheduling. *Information Processing & Management*, 58(1):102393, 2021.

[30] H. Baniata, D. Kimovski, R. Prodan, and A. **Kertesz**. Towards blockchain-based smart systems. In *CEUR Workshop Proceedings*, volume 3145, 2021.

[31] H. Baniata, A. Anaqreh, and A. **Kertesz**. DONS: Dynamic optimized neighbor selection for smart blockchain networks. *Future Generation Computer Systems*, 130:75–90, 2022.

[32] S. Bano, A. Sonnino, M. Al-Bassam, S. Azouvi, P. McCorry, S. Meiklejohn, and G. Danezis. Consensus in the age of blockchains. *arXiv preprint arXiv:1711.03936*, 2017.

[33] A. Barhanpure, P. Belandor, and B. Das. Proof of stack consensus for blockchain networks. In *International Symposium on Security in Computing and Communication*, pages 104–116. Springer, 2018.

[34] K. Batzavalis, R. Bala, A. Norta, and O. Norta Partners. A platform for leveraging blockchain technology for the storage, issuance and authentication of academic credentials. `https://www.trusteducation.io/`, 2021. Accessed: December 16, 2021.

[35] R. Beck, M. Avital, M. Rossi, and J. B. Thatcher. Blockchain technology in business and information systems research. *Business & Information Systems Engineering*, 59(6):381–384, Dec 2017.

[36] K. Bhumichitr and S. Channarukul. Acachain: Academic credential attestation system using blockchain. In *Proceedings of the 11th International Conference on Advances in Information Technology*, pages 1–8, 2020.

[37] Binance Academy. Proof of authority explained. `https://academy.binance.com/en/articles/proof-of-authority-explained`, 2020. Accessed: October 27, 2020.

[38] S. Bischof, A. Karapantelakis, C.-S. Nechifor, A. P. Sheth, A. Mileo, and P. Barnaghi. Semantic modelling of smart city data. 2014.

[39] Bitcoin Association. Simplified payment verification. `https://wiki.bitcoinsv.io/index.php/Simplified_Payment_Verification`. Accessed: November 27, 2020.

[40] Bitcoin.org. Bitcoin is an innovative payment network and a new kind of money. `https://bitcoin.org/en/`, 2009. Accessed: October 27, 2020.

[41] B. Blywis, M. Günes, F. Juraschek, O. Hahm, and N. Schmittberger. A survey of flooding, gossip routing, and related schemes for wireless multi-hop networks. Technical report, Freie Universitat, Berlin, Germany, 2011.

[42] A. Botta, W. De Donato, V. Persico, and A. Pescape. Integration of cloud computing and internet of things: a survey. *Future generation computer systems*, 56:684–700, 2016.

[43] M. A. Bouras, Q. Lu, F. Zhang, Y. Wan, T. Zhang, and H. Ning. Distributed ledger technology for ehealth identity privacy: State of the art and future perspective. *Sensors*, 20(2):483, 2020.

[44] A. Brogi, S. Forti, and A. Ibrahim. Deploying fog applications: How much does it cost, by the way? *Proceedings of the 8th International Conference on Cloud Computing and Services Science (CLOSER)*, pages 68–77, 2018.

[45] D. Bruneo, S. Distefano, F. Longo, G. Merlino, and A. Puliafito. *Turning Messina into a Smart City: The #SmartME Experience*, pages 135–152. Springer International Publishing, Cham, 2017.

[46] W. Burakowski, A. Beben, H. van den Berg, J. W. Bosman, G. Hasslinger, A. **Kertesz**, S. Latre, R. van der Mei, T. Pflanzner, P. G. Poullie, M. Sosnowski, B. Spinnewyn, and B. Stiller. *Traffic Management for Cloud Federation*, pages 269–312. Springer LNCS, Cham, 2018.

[47] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic. Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation Computer Systems*, 25(6):599–616, 2009.

[48] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. De Rose, and R. Buyya. Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and experience*, 41(1):23–50, 2011.

[49] T. Camp, J. Boleng, and V. Davies. A survey of mobility models for ad hoc network research. *Wireless Communications and Mobile Computing*, 2, 2002.

[50] B. Cao, Z. Zhang, D. Feng, S. Zhang, L. Zhang, M. Peng, and Y. Li. Performance analysis and comparison of pow, pos and dag based blockchains. *Digital Communications and Networks*, 2020.

[51] G. R. Carrara, L. M. Burle, D. S. Medeiros, C. V. N. de Albuquerque, and D. M. Mattos. Consistency, availability, and partition tolerance in blockchain: a survey on the consensus mechanism over peer-to-peer networking. *Annals of Telecommunications*, pages 1–12, 2020.

[52] B. Chang, L. Yang, M. Sensi, M. A. Achterberg, F. Wang, M. Rinaldi, and P. V. Mieghem. Markov modulated process to model human mobility. In R. M. Benito, C. Cherifi, H. Cherifi, E. Moro, L. M. Rocha, and M. Sales-Pardo, editors, *Complex Networks & Their Applications X*, pages 607–618, Cham, 2022. Springer International Publishing.

[53] Charafeddine Mechalikh. PureEdgeSim simulator. `https://github.com/CharafeddineMechalikh/PureEdgeSim`, 2018. Accessed: December 9, 2019.

[54] L. Chen, L. Xu, N. Shah, Z. Gao, Y. Lu, and W. Shi. On security analysis of proof-of-elapsed-time (poet). In P. Spirakis and P. Tsigas, editors, *Stabilization, Safety, and Security of Distributed Systems*, pages 282–297, Cham, 2017. Springer International Publishing.

[55] M. Chernyshev, Z. Baig, O. Bello, and S. Zeadally. Internet of things (iot): Research, simulators, and testbeds. *IEEE Internet of Things Journal*, 5(3):1637–1647, 2017.

[56] S. De Angelis, L. Aniello, R. Baldoni, F. Lombardi, A. Margheri, and V. Sassone. Pbft vs proof-of-authority: Applying the cap theorem to permissioned blockchain. 2018.

[57] A. Deshpande, P. Nasirifard, and H.-A. Jacobsen. evibes: Configurable and interactive ethereum blockchain simulation framework. In *Proceedings of the 19th International Middleware Conference (Posters)*, pages 11–12, 2018.

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

[59] M. Dorigo, M. Birattari, and T. Stutzle. Ant colony optimization. *IEEE computational intelligence magazine*, 1(4):28–39, 2006.

[60] P. Erdos and A. Renyi. On random graphs. *Publicationes Mathematicae*, 6: 290–297, 1959.

[61] European Commission. Mobile contact tracing apps in eu member states. https://ec.europa.eu/info/live-work-travel-eu/coronavirus-response/travel-during-coronavirus-pandemic/mobile-contact-tracing-apps-eu-member-states_en. Accessed: May 22, 2021.

[62] European Commission. Guidelines on FAIR Data Management in Horizon 2020. Version 3.0. http://ec.europa.eu/research/participants/data/ref/h2020/grants_manual/hi/oa_pilot/-h2020-hi-oa-data-mgt_en.pdf, 2016.

[63] M. Fang and J. Liu. Toward low-cost and stable blockchain networks. In *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*, 2020.

[64] C. Faria and M. Correia. Blocksim: Blockchain simulator. In *2019 IEEE International Conference on Blockchain (Blockchain)*, pages 439–446. IEEE, 2019.

[65] T. M. Fernandez-Carames and P. Fraga-Lamas. Towards next generation teaching, learning, and context-aware applications for higher education: A review on blockchain, iot, fog and edge computing enabled smart campuses and universities. *Applied Sciences*, 9(21):4479, 2019.

[66] G. George and S. Sankaranarayanan. Light weight cryptographic solutions for fog based blockchain. In *2019 International Conference on Smart Structures and Systems (ICSSS)*, pages 1–5. IEEE, 2019.

[67] A. Gervais, G. O. Karame, K. Wüst, V. Glykantzis, H. Ritzdorf, and S. Capkun. On the security and performance of proof of work blockchains. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 3–16, 2016.

[68] Z. Ghanbari, N. J. Navimipour, M. Hosseinzadeh, and A. Darwesh. Resource allocation mechanisms and approaches on the internet of things. *Cluster Computing*, 22:1253–1282, 2019.

[69] Global Times. China launches blockchain-based smart city identification system. https://www.globaltimes.cn/content/1168878.shtml, 2019. Accessed: October 27, 2020.

[70] Good Health Pass Collaborative. Good health pass: A safe path to global reopening. https://www.goodhealthpass.org/wp-content/uploads/2021/02/Good-Health-Pass-Collaborative-Principles-Paper.pdf. Accessed: July 7, 2021.

[71] Google corp. VM instances pricing. `https://cloud.google.com/compute/vm-instance-pricing`, 2021. Accessed: December 1, 2021.

[72] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami. Internet of things (iot): A vision, architectural elements, and future directions. *Future generation computer systems*, 29(7):1645–1660, 2013.

[73] H. Gupta, A. Vahid Dastjerdi, S. K. Ghosh, and R. Buyya. iFogSim: A toolkit for modeling and simulation of resource management techniques in the internet of things, edge and fog computing environments. *Software: Practice and Experience*, 47(9):1275–1296, 2017.

[74] M. Gushev. Dew computing architecture for cyber-physical systems and iot. *Internet of Things*, 11:100186, 2020.

[75] S. N. Han, G. M. Lee, N. Crespi, K. Heo, N. Van Luong, M. Brut, and P. Gatellier. DPWSim: A simulation toolkit for iot applications using devices profile for web services. In *2014 IEEE World Forum on Internet of Things (WF-IoT)*, pages 544–547. IEEE, 2014.

[76] A. B. Haque and B. Bhushan. Blockchain in a nutshell: State-of-the-art applications and future research directions. In *Blockchain and AI Technology in the Industrial Internet of Things*, pages 124–143. IGI Global, 2021.

[77] A. B. Haque, A. K. M. N. Islam, S. Hyrynsalmi, B. Naqvi, and K. Smolander. GDPR compliant blockchains–a systematic literature review. *IEEE Access*, 9: 50593–50606, 2021.

[78] X. He, Y. Cui, and Y. Jiang. An improved gossip algorithm based on semi-distributed blockchain network. In *2019 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)*, pages 24–27. IEEE, 2019.

[79] M. Heder, E. Rigo, D. Medgyesi, R. Lovas, S. Tenczer, F. Torok, A. Farkas, M. Emodi, J. Kadlecsik, G. Mezo, A. Pinter, and P. Kacsuk. The past, present and future of the ELKH cloud. *Informacios Tarsadalom*, 22:128, 2022.

[80] C. Hong and B. Varghese. Resource management in fog/edge computing: A survey on architectures, infrastructure, and algorithms. *ACM Computing Surveys*, 52:5, 2019.

[81] Q. Hu, M. Xu, S. Wang, and S. Guo. Sync or fork: Node-level synchronization analysis of blockchain. In *International Conference on Wireless Algorithms, Systems, and Applications*, pages 170–181. Springer, 2020.

[82] M. K. Hussein and M. H. Mousa. Efficient task offloading for iot-based applications in fog computing using ant colony optimization. *IEEE Access*, 8: 37191–37201, 2020.

[83] A. Iosup, H. Li, M. Jan, S. Anoep, C. Dumitrescu, L. Wolters, and D. H. Epema. The grid workloads archive. *Future Generation Computer Systems*, 24(7):672–686, 2008.

[84] R. Iqbal, T. A. Butt, M. Afzaal, and K. Salah. Trust management in social internet of vehicles: Factors, challenges, blockchain, and fog solutions. *International Journal of Distributed Sensor Networks*, 15(1):1550147719825820, 2019.

[85] F. Jameel, M. Nabeel, M. A. Jamshed, and R. Jantti. Minimizing forking in blockchain-based iot networks. In *2020 IEEE International Conference on Communications Workshops (ICC Workshops)*, pages 1–6. IEEE, 2020.

[86] Z. Jaroucheh, B. Ghaleb, and W. J. Buchanan. Sklcoin: Toward a scalable proof-of-stake and collective signature based consensus protocol for strong consistency in blockchain. In *2020 IEEE International Conference on Software Architecture Companion (ICSA-C)*, pages 143–150. IEEE, 2020.

[87] D. N. Jha, K. Alwasel, A. Alshoshan, X. Huang, R. Naha, S. Battula, S. Garg, D. Puthal, P. James, A. Zomaya, S. Dustdar, and R. Ranjan. IoTSim-Edge: A simulation framework for modeling the behavior of internet of things and edge computing environments. *Software: Practice and Experience*, 50:844–867, 2020.

[88] N. Jones. Top strategic iot trends and technologies through 2023. `https://www.gartner.com/en/documents/3890506`, 2018.

[89] E. E. Kalmar and A. **Kertesz**. What does i(o)t cost? In *Proceedings of the 8th ACM/SPEC on International Conference on Performance Engineering Companion*, ICPE '17 Companion, page 19–24, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450348997.

[90] B. Kang, D. Kim, and H. Choo. Internet of everything: A large-scale autonomic iot gateway. *IEEE Transactions on Multi-Scale Computing Systems*, 3(3):206–214, 2017.

[91] F. Kaup, P. Gottschling, and D. Hausheer. Powerpi: Measuring and modeling the power consumption of the raspberry pi. *39th Annual IEEE Conference on Local Computer Networks*, pages 236–243, 2014.

[92] G. Kecskemeti. DISSECT-CF: a simulator to foster energy-aware scheduling in infrastructure clouds. *Simulation Modelling Practice and Theory*, 58:188–218, 2015.

[93] A. **Kertesz**. Characterizing cloud federation approaches. In *Cloud Computing*, pages 277–296. Springer, 2014.

[94] A. **Kertesz**. Interoperable data management using personal and infrastructure clouds. *IEEE Cloud Computing*, 2(1):22–28, 2015.

[95] A. **Kertesz**. Block the chain: Software weapons of fighting against COVID-19. *Computer*, 55(9):43–53, 2022.

[96] A. **Kertesz** and H. Baniata. Consistency analysis of distributed ledgers in fog-enhanced blockchains. In R. Chaves, D. B. Heras, A. Ilic, D. Unat, R. M. Badia, A. Bracciali, P. Diehl, A. Dubey, O. Sangyoon, S. L. Scott, and L. Ricci, editors, *Euro-Par 2021: Parallel Processing Workshops*, pages 393–404, Cham, 2022. Springer International Publishing.

[97] A. **Kertesz** and S. Varadi. *Legal Aspects of Data Protection in Cloud Federations*, pages 433–455. Springer Berlin Heidelberg, Berlin, Heidelberg, 2014.

[98] A. **Kertesz**, G. Kecskemeti, M. Oriol, P. Kotcauer, S. Acs, M. Rodriguez, O. Merce, A. C. Marosi, J. Marco, and X. Franch. Enhancing federated cloud management with an integrated service monitoring approach. *Journal of Grid Computing*, 11(4):699–720, 2013.

[99] A. **Kertesz**, G. Kecskemeti, and I. Brandic. An interoperable and self-adaptive approach for sla-based service virtualization in heterogeneous cloud environments. *Future Generation Computer Systems*, 32:54–68, 2014.

[100] A. **Kertesz**, F. Otvos, and P. Kacsuk. A case study for biochemical application porting in european grids and clouds. *Concurrency and Computation: Practice and Experience*, 26(10):1730–1743, 2014.

[101] A. **Kertesz**, J. D. Dombi, and A. Benyi. A pliant-based virtual machine scheduling solution to improve the energy efficiency of iaas clouds. *Journal of Grid Computing*, 14:41–53, 2016.

[102] A. **Kertesz**, T. Pflanzner, and T. Gyimothy. A mobile iot device simulator for iot-fog-cloud systems. *Journal of Grid Computing*, 17(3):529–551, 2019.

[103] A. M. Khan, L. Navarro, L. Sharifi, and L. Veiga. Clouds of small things: Provisioning infrastructure-as-a-service from within community networks. In *2013*

IEEE 9th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), pages 16–21. IEEE, 2013.

[104] D. Kimovski, A. Marosi, S. Gec, N. Saurabh, A. **Kertesz**, G. Kecskemeti, V. Stankovski, and R. Prodan. Distributed environment for efficient virtual machine image management in federated cloud architectures. *Concurrency and Computation: Practice and Experience*, 30(20):e4220, 2018.

[105] S. King and S. Nadal. Ppcoin: Peer-to-peer crypto-currency with proof-of-stake. *self-published paper, August*, 19:1, 2012.

[106] K. Kritikos, B. Pernici, P. Plebani, C. Cappiello, M. Comuzzi, S. Benrernou, I. Brandic, A. **Kertesz**, M. Parkin, and M. Carro. A survey on service quality description. *ACM Comput. Surv.*, 46(1), 2013.

[107] T. Kumar, E. Harjula, M. Ejaz, A. Manzoor, P. Porambage, I. Ahmad, M. Liyanage, A. Braeken, and M. Ylianttila. Blockedge: Blockchain-edge framework for industrial iot networks. *IEEE Access*, 2020.

[108] P. K. Lahiri, R. Mandal, S. Banerjee, and U. Biswas. An approach towards developments of smart covid-19 patient's management and triaging using blockchain framework. *Research Square*, 2020.

[109] J. Lan, X. Liu, P. Shenoy, and K. Ramamritham. Consistency maintenance in peer-to-peer file sharing networks. In *Proceedings the Third IEEE Workshop on Internet Applications. WIAPP 2003*, pages 90–94. IEEE, 2003.

[110] S. Liaskos, T. Anand, and N. Alimohammadi. Architecting blockchain network simulators: a model-driven perspective. In *2020 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, pages 1–3. IEEE, 2020.

[111] M. M. Lopes, W. A. Higashino, M. A. Capretz, and L. F. Bittencourt. 10.1145/3147234.3148101. *Proceedings of The 10th International Conference on Utility and Cloud Computing*, page 47–52, 2017.

[112] Z. Ma, Q. Zhao, J. Yuan, X. Zhou, and L. Feng. Fork probability analysis of pouw consensus mechanism. In *2020 IEEE International Conference on Smart Internet of Things (SmartIoT)*, pages 333–337. IEEE, 2020.

[113] G. S. Machado, D. Hausheer, and B. Stiller. Considerations on the interoperability of and between cloud computing standards. In *27th open grid forum (OGF27), G2C-Net workshop: from grid to cloud networks*, 2009.

[114] R. Mahmud, R. Kotagiri, and R. Buyya. Fog computing: A taxonomy, survey and future directions. In B. Di Martino, K.-C. Li, L. T. Yang, and A. Esposito, editors, *Internet of Everything: Algorithms, Methodologies, Technologies and Perspectives*, pages 103–130. Springer Singapore, Singapore, 2018.

[115] A. Maier, A. Sharp, and Y. Vagapov. Comparative analysis and practical implementation of the esp32 microcontroller module for the internet of things. *International Conference on Internet Technologies and Applications*, pages 143–148, 2017.

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

[117] M. Marjani, F. Nasaruddin, A. Gani, A. Karim, I. A. T. Hashem, A. Siddiqa, and I. Yaqoob. Big iot data analytics: Architecture, opportunities, and open research challenges. *IEEE Access*, 6:5247–5261, 2017.

[118] A. Markus and A. **Kertesz**. A survey and taxonomy of simulation environments modelling fog computing. *Simulation Modelling Practice and Theory*, 101:102042, 2020.

[119] A. Markus and A. **Kertesz**. Investigating iot application behaviour in simulated fog environments. In D. Ferguson, C. Pahl, and M. Helfert, editors, *Cloud Computing and Services Science*, pages 258–276, Cham, 2021. Springer International Publishing. ISBN 978-3-030-72369-9.

[120] A. Markus and A. **Kertesz**. Modelling energy consumption of IoT devices in DISSECT-CF-fog. In *Proceedings of the 11th International Conference on Cloud Computing and Services Science*, Best poster award. SCITEPRESS, 2021.

[121] A. Markus, A. **Kertesz**, and G. Kecskemeti. Cost-aware IoT extension of DISSECT-CF. *Future Internet*, 9(3):47, 2017.

[122] A. Markus, M. Biro, G. Kecskemeti, and A. **Kertesz**. Actuator behaviour modelling in IoT-fog-cloud simulation. *PeerJ Computer Science*, 7:e651, 2021.

[123] A. Markus, J. D. Dombi, and A. **Kertesz**. Location-aware task allocation strategies for iot-fog-cloud environments. In *2021 29th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)*, pages 185–192, 2021.

[124] A. Markus, M. Biro, K. Skala, Z. Sojat, and A. **Kertesz**. Modeling dew computing in DISSECT-CF-Fog. *Applied Sciences*, 12(17), 2022.

[125] E. Mathieu, H. Ritchie, L. Rodés-Guirao, C. Appel, C. Giattino, J. Hasell, B. Macdonald, S. Dattani, D. Beltekian, E. Ortiz-Ospina, and M. Roser. Coronavirus pandemic (covid-19). *Our World in Data*, 2020. `https://ourworldindata.org/coronavirus`.

[126] R. A. Memon, J. Li, J. Ahmed, A. Khan, M. I. Nazir, and M. I. Mangrio. Modeling of blockchain based systems using queuing theory simulation. In *2018 15th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP)*, pages 107–111. IEEE, 2018.

[127] R. A. Memon, J. P. Li, M. I. Nazeer, A. N. Khan, and J. Ahmed. Dualfog-iot: Additional fog layer for solving blockchain integration problem in internet of things. *IEEE Access*, 7:169073–169093, 2019.

[128] B. Mishra and A. **Kertesz**. The use of MQTT in M2M and IoT systems: A survey. *IEEE Access*, 8:201071–201086, 2020.

[129] B. Mishra, B. Mishra, and A. **Kertesz**. Stress-testing MQTT brokers: A comparative analysis of performance measurements. *Energies*, 14(18), 2021.

[130] J. Misic, V. B. Misic, and X. Chang. Performance of bitcoin network with synchronizing nodes and a mix of regular and compact blocks. *IEEE Transactions on Network Science and Engineering*, 2020.

[131] A. Montresor and M. Jelasity. Peersim: A scalable p2p simulator. In *2009 IEEE Ninth International Conference on Peer-to-Peer Computing*, pages 99–100. IEEE, 2009.

[132] I. A. Moschakis and H. D. Karatza. Towards scheduling for internet-of-things applications on clouds: a simulated annealing approach. *Concurrency and Computation: Practice and Experience*, 27(8):1886–1899, 2015.

[133] 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.

[134] N. Naik. Choice of effective messaging protocols for iot systems: Mqtt, coap, amqp and http. In *2017 IEEE international systems engineering symposium (ISSE)*, pages 1–7. IEEE, 2017.

[135] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Decentralized Business Review*, page 21260, 2008.

[136] P. G. V. Naranjo, Z. Pooranian, M. Shojafar, M. Conti, and R. Buyya. Focan: A fog-supported smart city network architecture for management of applications in the internet of everything environments. *Journal of Parallel and Distributed Computing*, 132:274–283, 2018.

[137] S. Nastic, S. Sehic, D.-H. Le, H.-L. Truong, and S. Dustdar. Provisioning software-defined iot cloud systems. In *2014 international conference on future internet of things and cloud*, pages 288–295. IEEE, 2014.

[138] E. C. H. Ngai, M. R. Lyu, and J. Liu. A real-time communication framework for wireless sensor-actuator networks. *IEEE Aerospace Conference*, 9, 2006.

[139] A. Panarello, N. Tapas, G. Merlino, F. Longo, and A. Puliafito. Blockchain and iot integration: A systematic survey. *Sensors*, 18(8):2575, 2018.

[140] G. Pandurangan, P. Robinson, and M. Scquizzato. A time-and message-optimal distributed algorithm for minimum spanning trees. *ACM Transactions on Algorithms*, 16(1), 2019.

[141] I. Petri, M. Barati, Y. Rezgui, and O. F. Rana. Blockchain for energy sharing and trading in distributed prosumer communities. *Computers in Industry*, 123: 103282, 2020.

[142] T. Pflanzner and A. **Kertesz**. A taxonomy and survey of IoT cloud applications. *EAI Endorsed Transactions on Internet of Things*, 3(12), 2017.

[143] T. Pflanzner and A. **Kertesz**. A private gateway for investigating iot data management. In V. M. Munoz, D. Ferguson, M. Helfert, and C. Pahl, editors, *Proceedings of the 8th International Conference on Cloud Computing and Services Science, CLOSER 2018, Funchal, Madeira, Portugal, March 19-21, 2018*, pages 526–532. SciTePress, 2018.

[144] T. Pflanzner, K. Z. Leszko, and A. **Kertesz**. SUMMON: Gathering smart city data to support iot-fog-cloud simulations. In *2018 Third International Conference on Fog and Mobile Edge Computing (FMEC)*, pages 71–78. IEEE, 2018.

[145] T. Pflanzner, Z. Feher, and A. **Kertesz**. A crawling approach to facilitate open iot data archiving and reuse. In *2019 Sixth International Conference on Internet of Things: Systems, Management and Security (IOTSMS)*, pages 235–242. IEEE, 2019.

[146] T. Pflanzner, M. Hovari, I. Vass, and A. **Kertesz**. Designing an IoT-Cloud gateway for the internet of living things. In *International Conference on Cloud Computing and Services Science*, pages 23–41. Springer, 2019.

[147] A. L. Phelan. COVID-19 immunity passports and vaccination certificates: scientific, equitable, and legal challenges. *The Lancet*, 395(10237):1595–1598, May 2020.

[148] P.-Y. Piriou and J.-F. Dumas. Simulation of stochastic blockchain models. In *2018 14th European Dependable Computing Conference (EDCC)*, pages 150–157. IEEE, 2018.

[149] 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.

[150] N. Pokrovskaia. Tax, financial and social regulatory mechanisms within the knowledge-driven economy. blockchain algorithms and fog computing for the efficient regulation. In *2017 XX IEEE International Conference on Soft Computing and Measurements (SCM)*, pages 709–712. IEEE, 2017.

[151] P. C. Pop. The generalized minimum spanning tree problem: An overview of formulations, solution procedures and latest advances. *European Journal of Operational Research*, 283(1):1–15, 2020.

[152] C. Puliafito, E. Mingozzi, F. Longo, A. Puliafito, and O. Rana. Fog computing for the internet of things: A survey. *ACM Transactions on Internet Technology*, 19:2, 2019.

[153] C. Puliafito, D. M. Goncalves, M. M. Lopes, L. L. Martins, E. Madeira, E. Mingozzi, O. Rana, and L. F. Bittencourt. MobFogSim: Simulation of mobility and migration for fog computing. *Simulation Modelling Practice and Theory*, 101: 102062, 2020.

[154] L. Qian, L. Z, Y. Du, and L. Guo. Cloud computing: An overview. *IEEE International Conference on Cloud Computing*, page 626–631, 2009.

[155] A. R. and B. A.L. Statistical mechanics of complex networks. *Reviews of Modern Physics*, 74(1):47–97, 2002.

[156] U. U. Rahman, K. Bilal, A. Erbad, O. Khalid, and S. U. Khan. Nutshell—simulation toolkit for modeling data center networks and cloud computing. *IEEE Access*, 7:19922–19942, 2019.

[157] R. K. Raman, R. Vaculin, M. Hind, S. L. Remy, E. K. Pissadaki, N. K. Bore, R. Daneshvar, B. Srivastava, and K. R. Varshney. A scalable blockchain approach for trusted computation and verifiable simulation in multi-party col-

laborations. In *2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, pages 277–284. IEEE, 2019.

[158] Z. Rejiba, X. Masip-Bruin, and E. Marín-Tordera. A survey on mobility-induced service migration in the fog, edge, and related computing paradigms. *ACM Comput. Surv.*, 52(5), 2019.

[159] A. Reyna, C. Martin, J. Chen, E. Soler, and M. Diaz. On blockchain and its integration with iot. challenges and opportunities. *Future Generation Computer Systems*, 88:173–190, 2018.

[160] M. Reynolds and U. Schurr. The 4th international plant phenotyping symposium. *Plant science: an international journal of experimental plant biology*, 282: 1, 2019.

[161] P. Rimba, A. B. Tran, I. Weber, M. Staples, A. Ponomarev, and X. Xu. Comparing blockchain and cloud services for business process execution. In *2017 IEEE International Conference on Software Architecture (ICSA)*, pages 257–260. IEEE, 2017.

[162] H. P. Sajjad, K. Danniswara, A. Al-Shishtawy, and V. Vlassov. Spanedge: Towards unifying stream processing over central and near-the-edge data centers. *EEE/ACM Symposium on Edge Computing (SEC)*, pages 168–178, 2016.

[163] R. G. Sargent. Verification and validation of simulation models. *Journal of simulation*, 7(1):12–24, 2013.

[164] Y. Shahsavari, K. Zhang, and C. Talhi. A theoretical model for fork analysis in the bitcoin network. In *2019 IEEE International Conference on Blockchain (Blockchain)*, pages 237–244. IEEE, 2019.

[165] J. Shalf. The future of computing beyond moore's law. *Philosophical Transactions of the Royal Society A*, 378(2166):20190061, 2020.

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

[167] I. Silva, R. Leandro, D. Macedo, and L. A. Guedes. A dependability evaluation tool for the internet of things. *Computers & Electrical Engineering*, 39(7):2005–2018, 2013.

[168] Smart Dubai Department. Blockchain. `https://www.smartdubai.ae/initiatives/blockchain`, 2020. Accessed: October 27, 2020.

[169] Smartcity Press. China taking a big leap with blockchain. https://www.smartcity.press/blockchain-technology-china/, 2019. Accessed: October 27, 2020.

[170] C. Sonmez, A. Ozgovde, and C. Ersoy. Edgecloudsim: An environment for performance evaluation of edge computing systems. *Second International Conference on Fog and Mobile Edge Computing (FMEC)*, pages 39–44, 2017.

[171] S. Sotiriadis, N. Bessis, N. Antonopoulos, and A. Anjum. Simic: Designing a new inter-cloud simulation platform for integrating large-scale resource management. In *2013 IEEE 27th International Conference on Advanced Information Networking and Applications (AINA)*, pages 90–97. IEEE, 2013.

[172] S. Sotiriadis, N. Bessis, E. Asimakopoulou, and N. Mustafee. Towards simulating the internet of things. In *2014 28th International Conference on Advanced Information Networking and Applications Workshops*, pages 444–448. IEEE, 2014.

[173] H. Sundmaeker, P. Guillemin, P. Friess, and S. Woelffle. Vision and challenges for realising the internet of things. *Cluster of European Research Projects on the Internet of Things, European Commision*, 3(3):34–36, 2010.

[174] S. Svorobej, P. T. Endo, M. Bendechache, C. Filelis-Papadopoulos, K. M. Giannoutakis, G. A. Gravvanis, D. Tzovaras, J. Byrne, and T. Lynn. Simulating fog and edge computing scenarios: An overview and research challenges. *Future Internet*, 11:3, 2019.

[175] N. Tariq, M. Asim, F. Al-Obeidat, M. Zubair Farooqi, T. Baker, M. Hammoudeh, and I. Ghafir. The security of big data in fog-enabled iot applications including blockchain: a survey. *Sensors*, 19(8):1788, 2019.

[176] 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.

[177] The Linux Foundation. What is hyperledger? https://www.hyperledger.org/, 2020. Accessed: October 27, 2020.

[178] G. Umarani Srikanth, V. U. Maheswari, P. Shanthi, and A. Siromoney. Tasks scheduling using ant colony optimization. *Journal of Computer Science*, 8(8): 1314–1320, 2012.

[179] R. B. Uriarte and R. De Nicola. Blockchain-based decentralized cloud/fog solutions: Challenges, opportunities, and standards. *IEEE Communications Standards Magazine*, 2(3):22–28, 2018.

[180] S. Varadi, A. **Kertesz**, and M. Parkin. The necessity of legally compliant data management in european cloud architectures. *Computer Law Security Review*, 28(5):577–586, 2012.

[181] S. Varadi, G. G. Varkonyi, and A. **Kertesz**. Legal issues of social iot services: The effects of using clouds, fogs and ai. In A. E. Hassanien, R. Bhatnagar, N. E. M. Khalifa, and M. H. N. Taha, editors, *Toward Social Internet of Things (SIoT): Enabling Technologies, Architectures and Applications: Emerging Technologies for Connected and Smart Social Objects*, pages 123–138. Springer International Publishing, Cham, 2020.

[182] A. Varga and R. Hornig. An overview of the omnet++ simulation environment. In *Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops*, page 60. Institute for Computer Sciences, Social Informatics, 2008.

[183] D. Vujicic, D. Jagodic, and S. Ranic. Blockchain technology, bitcoin, and ethereum: A brief overview. In *2018 17th international symposium infoteh-jahorina (infoteh)*, pages 1–6. IEEE, 2018.

[184] B. Wang, S. Chen, L. Yao, B. Liu, X. Xu, and L. Zhu. A simulation approach for studying behavior and quality of blockchain networks. In *International Conference on Blockchain*, pages 18–31. Springer, 2018.

[185] B. Wang, Q. Wang, S. Chen, and Y. Xiang. Security analysis on tangle-based blockchain through simulation. In *Australasian Conference on Information Security and Privacy*, pages 653–663. Springer, 2020.

[186] A. Wilczynski and J. Kolodziej. Modelling and simulation of security-aware task scheduling in cloud computing based on blockchain technology. *Simulation Modelling Practice and Theory*, 99:102038, 2020.

[187] T. Xue, Y. Yuan, Z. Ahmed, K. Moniz, G. Cao, and C. Wang. Proof of contribution: A modification of proof of work to increase mining efficiency. In *2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC)*, volume 1, pages 636–644. IEEE, 2018.

[188] S. Yi, C. Li, and Q. Li. A survey of fog computing: Concepts, applications and issues. *In Proceedings of the Workshop on Mobile Big Data*, page 37–42, 2015.

[189] A. Yousefpour, C. Fung, T. Nguyen, K. Kadiyala, F. Jalali, A. Niakanlahiji, J. Kong, and J. P. Jue. All one needs to know about fog computing and related edge computing paradigms: A complete survey. *Journal of Systems Architecture*, 98:289–330, 2019.

[190] X. Zeng, S. K. Garg, P. Strazdins, P. P. Jayaraman, D. Georgakopoulos, and R. Ranjan. IOTSim: A simulator for analysing iot applications. *Journal of Systems Architecture*, 72:93–107, 2017.

[191] R. Zhang and W. K. V. Chan. Evaluation of energy consumption in block-chains with proof of work and proof of stake. In *Journal of Physics: Conference Series*, volume 1584(1), page 012023. IOP Publishing, 2020.

[192] F. Zhao, X. Guo, and W. K. V. Chan. Individual green certificates on blockchain: A simulation approach. *Sustainability*, 12(9):3942, 2020.