Admission Control for Ad-hoc Edge Cloud

Ana Juan Ferrer[a,b], Javier Panadero, Joan-Manuel Marques, Josep Jorba[a]

[a] *Universitat Oberta de Catalunya, Barcelona, Spain*
[b] *Atos, Research and Innovation, Barcelona, Spain*

Abstract

Penetration of connected devices in society offers overwhelming figures over the last decade. Today connected devices are not only available everywhere, but they are quickly gaining complexity in terms of their ability to carry significant compute and storage capacities. This way, computing is no longer confined into certain stationary compute devices but is starting to be embedded and pervasive to virtually everything. This, combined with the fact that AI processes are multiplying the need of timely processing at the Edge, reveals the urgent need of exploiting all compute capacity available at the Edge of the network. Ad-hoc Edge Cloud is a distributed and decentralized Edge Computing system dynamically formed out of IoT Edge computing resources which aims to exploit this capacity. The specific characteristics of the IoT devices which constitute this infrastructure bring specific challenges to resource management in this context due to heterogeneity, dynamicity and volatility of resources, resulting in probability of node churn. In this work, we analyze specific admission control and service placement issues for Ad-hoc Edge Cloud by presenting an specific Admission Control mechanism for which we define an associated resource availability prediction model. For this we focus on the evaluation at scale of implications of resource dynamicity and scale in terms of services and constituent nodes.

Keywords: Edge Computing, Ad-hoc Edge infrastructure, Admission Control, Service placement, Resource availability prediction

2010 MSC: 00-01, 99-00

1. Introduction

Edge Computing is today one of the major IT trends which is developed at the intersection among four main IT developments: Internet of Things (IoT), Cloud Computing, Networks and more recently, Artificial Intelligence (AI). Edge computing (named Fog computing by some authors) emerged to bring Cloud computing capacities at the Edge of the network in order to address latency issues present in IoT scenarios. Edge computing overall purpose is to provide a compute environment located nearby to data generation sources that avoids latency issues detected in accessing Cloud services. Edge Computing brings together networking with distinctive cloud principles to define distributed computing platforms that attend the specific needs of IoT[1].

In fact, this has established an initial step towards the decentralisation of Cloud computing initiating its transformation from the provision of services in dedicated datacentres for which resources were perceived as unlimited to a more decentralised approach in which these cloud services are offered in combination with stationary Edge devices[2]. This approach is today materialised in existing market offerings that provide initial IoT data filtering and pre-processing with integrated synchronization with cloud services by major providers such as AWS Greengrass[3] and Azure IoT Edge[4]. However, Connected IoT devices are today not only available everywhere, but also, they gradually carrying on noteworthy compute resources. Current IoT environments are not only constituted by simple sensors with 8-bit micro-processors [5], but increasingly these IoT environments are founded by complex devices which are assemblies of non-negligible computing and storage resources aggregated together with diverse sensors and actuators, such as robots, drones and autonomous vehicles. Moreover, innovative compute devices are being released to the market carrying on application specific processors for AI processing with the purpose of facilitating embedding compute intelligence into all kinds of IoT devices. While todays Edge computing environments typically solely considers stationary dedicated Edge computing devices, the growth in the complexity of IoT devices is call-

ing to take advantage all compute and storage capacity available in a specific location in all kinds of IoT edge devices.

The aim of Ad-hoc Edge Cloud [2, 6] is to provide an Edge management system that harnesses the increasingly available computing capacity at the Edge of the network to form ephemeral compute infrastructures out of resources available in a certain physical space in a concrete point in time, making use of distributed heterogeneous non-dedicated IoT resources at the Edge of the network.

The specific characteristics of the IoT devices which take part in this infrastructure bring specific challenges to its overall resource management practices, and specifically to its Admission control processes. These challenges relate to the dynamicity on the availability of resources. Resources availability dynamicity is motivated on one side, by the still constrained nature in terms of battery and compute capacities of the heterogeneous IoT devices. But more importantly, it is determined by the mobile nature of some of these IoT devices due to its associated connectivity instability. These result the imperative need for Ad-hoc Edge Cloud of handling heterogeneity, dynamicity and volatility of resources, resulting into probability of node churn. In this context node churn is characterized by the dynamic and volatile behavior of IoT Edge resources being intermittently available and unavailable to the system. Admission Control processes represent the decision that determines whether to accept a service to be executed in the infrastructure and in the affirmative case, to identify the set of the available IoT Edge resources which are most appropriated to place the different service components. The specific considerations the level of Admission control mechanisms in Ad-hoc Edge Cloud are defined taking into account its past behaviour in terms of connections and disconnections, representing the source in which we base the resource availability prediction model for infrastructure that is the main research area of this work.

The remainder of the paper is organized as follows. Section 2 elaborates on the Ad-hoc Edge Computing concept. Section 3 presents the Infrastructure model in which it relies. Section 4 defines the typologies of services to be

executed in Ad-hoc Edge Cloud. Section 5 depicts its proposed Architecture. Section 6 presents the Admission control mechanisms and its defined resource availability prediction model. Section 7 presents the performed evaluation over these. Section 8 illustrates related work in the areas of resource availability prediction and Edge and Fog computing placement while section 9 supplies information regarding future work and conclusions.

2. Ad-hoc Edge Cloud Concept

The overall idea of Ad-hoc Edge Cloud is to exploit available capacity at the Edge in order to create on the fly and in an opportunistic manner these Ad-hoc Edge Cloud compute infrastructures that respond to the need of extracting value out of the overwhelming data availability collected from IoT Edge devices resulting into growing processing demand at the Edge powered by the technological advances in AI and deep leering.

In order to do so, Ad-hoc Edge Cloud takes into consideration the specificities of Edge devices, their characteristics and the non-dedicated inherent quality of the infrastructure by developing mechanisms to handle their expected massive number, the dynamic resources availability (due to mobility and resources constraints) and the heterogeneous nature of IoT Edge resources. Ad-hoc Edge Cloud develops the idea of Decentralised Cloud [2], by creating dynamic ecosystems of IoT Edge Devices in a completed distributed and decentralised manner. This way Ad-hoc Edge Cloud relies in a decentralized management which is distributed among all participant resources, which avoids a single point of failure for the infrastructure and offers inherent mechanisms to scale. Besides, this fact also ensures the autonomy of the Ad-hoc Edge infrastructure, not needing to rely on external management layers, for instance, located at the cloud and which can hinder its operation in case of not reliable connectivity.

Distributed and decentralised management is also key in order to handle the uncertainty on resource availability in the Ad-hoc Edge Cloud. This dynamicity is often named node churn and refers to the volatile behaviour of resources in

relation to their accessibility to be part of the system. This dynamicity is motivated by many different factors, such as a change of edge device locations, they can deplete the battery or punctual loss of connectivity. This enforces the need of specific procedures on admission control and service management in the Ad-hoc Edge Computing infrastructure.

3. Ad-hoc Edge Infrastructure Model

Three distinctive aspects characterise the kind of infrastructure resources Ad-hoc Edge Cloud relies on: their heterogeneity, their potential mobility and their restrictions in terms of the battery and overall capacity. These three characteristics are crucial to understand the dynamic availability of resources considered by this Ad-hoc Edge Cloud and which significantly differs from typical resource management practices in Cloud computing.

Heterogeneity

Cloud computing model is sustained on economies of scale. These are enabled in large public cloud providers by the capacity of automation over standardised and homogeneous huge farms of servers which provide compute, storage and networking resources to the cloud. Homogeneity in these large data centre set-ups drives to reduced operation costs thanks to standardised management practices and automation.

Edge computing environments, on the contrary, are characterised by its heterogeneity. The expected massive growth on connected IoT devices together with the wide variety of use cases in which these can be used, brings diverseness at several levels. Devices at the Edge can range from simple sensors able to capture data (i.e. a temperature sensor) to complex aggregations of sensors and actuators embedded together with high performant compute and storage resources, such as an autonomous car [7]. Edge devices can be stationary wired powered devices for which size is not a critical design issue or can be constrained battery powered mobile devices with strict requirements for optimisation of devices autonomy.

In addition, the increasing computing demands for these devices to provide more intelligent features is driving the emergence of innovative sets of compute devices designed to be embedded into IoT environments and devices. In the recent years use of Raspberry Pi[8] for this purpose gained significant popularity. However, nowadays the rise of AI and its demanding compute requirements, is bringing the appearance of embeddable devices designed specifically for execution of AI at the edge [9], AI accelerators, by means of providing specific purpose hardware micro-processors and computer systems such as Intel movidius[10], NVIDIAs Jetson systems[11] and not long ago, Google Coral [12] to cite some remarkable examples.

In this context, it is crucial for our work to be able to handle all diversity that arises from the diversity of capacities of the edge resources which can take part in an Ad-hoc Edge Cloud including different processor architectures (CPU, GPU, TPU, FPGA) in addition to other considerations such as variety of operating systems edge devices can operate (i.e. Linux, Raspbian, Robot Operating system) as well as connectivity protocols that need to be supported. We aim to adopt Edge devices categorization provided by HEADS [13] project which classifies Edge devices as: Tiny (8 and 16 bit microcontrollers), Small (between 64-128 Kb memory) and Large ("devices running a general operating system like Linux or similar [13]); . The latter are the specific target for this work. These encompass devices such Arduino Yun, Raspberry Pi, Android and iOS. Long term feasibility of this approach is evidenced by increasing support for operating-system-level virtualization, containerization, in typically considered largely constrained environments. This is demonstrated by means of the increased availability of conteinarisation technologies more lightweight than Docker[14], such as Unikernels[15], Kata Container[16] and gVisor[17].

Mobility

Another source of differentiation of Resource management in the context of Ad-hoc Edge Cloud versus the traditional data centre resource management in Clouds, is the potential mobility of the devices taking part in the infrastructure.

Mobility of resources involves using unreliable network links for the Edge

device connectivity and its consequent, resource volatility and lack of reliability. These issues are key aspects of this work and have not been until now analysed in todays Edge and Cloud computing stationary resources environments. Node churn, the term commonly employed to describe the dynamicity in resources appearing and disappearing of the system, is instead an area widely studied in P2P [18] and to less extend in Mobile Cloud Computing [2] areas.

In addition to this, mobility of devices assumes battery powered environments and need to optimise battery lifetime in order to not compromise devices autonomy and their availability to the Ad-hoc Edge computing infrastructure. Hence, Mobility of devices motivates two main matters to be considered as part of this work: volatility in the node availability in the infrastructure, node churn, and resource limitations.

Resource Limitations

As previously presented heterogeneity is a key characteristic of this environment. However, generally speaking target IoT Edge devices for this work are restricted in terms of computational and storage capacity. The motivation for this, is the need of IoT Edge devices producers to achieve the appropriate trade-off among cost, energy consumption and performance in the devices being launched to the market [5]. Providers of intelligent IoT Edge devices such as smartphones, automobiles, robots or drones need to get the right balance among rich functionalities, appropriate energy consumption to optimize devices autonomy and overall profitable solutions costs. In this complex environment, IoT Edge resource providers trend to choose optimal cost solutions with lower performance, reserving the option of higher performant processor architecturesas those described in previous section, for cases in which they derive a significant competitive advantage for their devices. This way, a solution which aims to rely on this kind of devices has to anticipate limitations in terms of available compute and storage resources together with constraints in batteries as energy supply.

4. Ad-hoc Edge Service Model

In order to ensure as much as possible interoperability of the Ad-hoc Edge Computing infrastructure with existing Edge and Cloud offerings the resource model is illustrated in a generic manner which could be implemented using available service descriptors in standards and commercial offerings, such as Kubernetes deployments and pods [19] or AWS CloudFormation [20] template. Independently of the language selected to express the workload characteristics, at the end of the day existing languages offer a similar structure which is represented in Figure 1 Ad-hoc Edge Computing Service Model. In more details, the definition of a service it is done based on: a set of service components, number of instances of each component, as well as, optionally a set of scalability rules. More in detail:

- A service is composed by a series of components. In its simplest form a service has a single component.
- Each component belongs to a template, which determines necessary physical resources to allocate it. For each service component its hardware requirements are expressed in its template. Common resources considered are amount of disk, amount of memory and CPUs.
- Multiple instances can exist for a service component.
- The definition of the service indicates how many of component instances to be instantiated when the service it is deployed (minimum value of instances), as well as the maximum number of instances that overall can be created (by means of application of scalability rules). Each instance it is deployed in the resource that has enough capacity to host its hardware requirements defined by its template.
- Component creation it is instantiation of a determined component in an available resource.

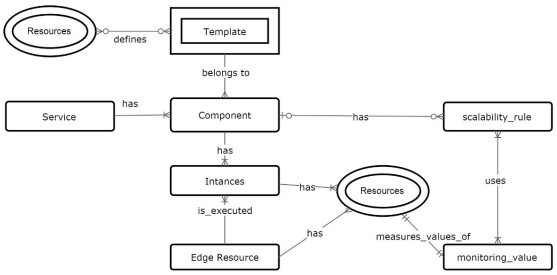

Figure 1: Ad-hoc Edge Computing Service Model

- Deployment determines the necessary steps to have a service component up and running in a concrete infrastructure resource.

Typically, a service will be deployed as a set of containers described as service components. The use of virtualization (OS, hyper-visor based) techniques provides isolation mechanisms among the user data and execution in contributed resources. It also allows transparent allocation of each component of the distributed service inside the Ad-hoc Edge Infrastructure. In addition to this, it makes easier the process of horizontal elasticity, i.e. adding/removing extra capacity for each component during runtime to maintain a certain level of performance for the overall service when there are variations in the workload.

5. Ad-hoc Edge Cloud framework

The architecture for Ad-hoc Edge Infrastructure was presented in [6] and it is depicted in **Figure 2**. The architecture is structured in two main contexts which are present in all devices that take part in the Ad-hoc Edge Infrastructure. Contexts represent separations of concerns: Edge Device Context, entails tools and mechanisms for the management of a particular node of the infrastructure. It enables Edge resources to execute services or parts of them.

- Node Manager allows handling a node as part of the infrastructure. It enables the unified description of the specific resource static characteristics

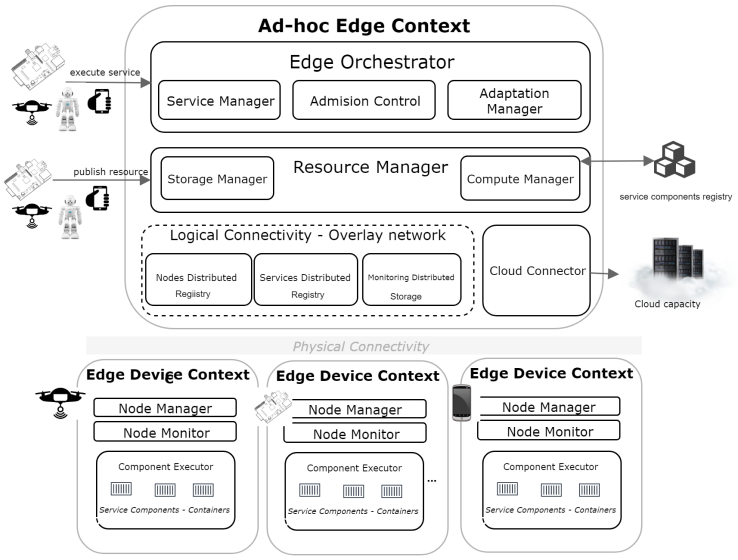

Figure 2: Ad-hoc Edge Cloud Architecture

and with support of Node monitor, of its dynamic characteristics. These are described in detail in Table 1. Finally, the component executor is able to manage the life-cycle of services components to be executed in the Edge node with support of OS workload virtualization tools such as Docker.

- Ad-hoc Edge Cloud Context designates the components devoted to the management of the overall infrastructure cluster distributed among all participant resources. The core component in this layer is the Logical connectivity layer which is based in etcd[21] and handles the distributed indexes to manage nodes in the infrastructure, services and monitoring information as a distributed system with no central management. This allows Resource management, Admission control and Service management to have a unified view of the status of the infrastructure by means of its out of the box replica and data distribution mechanisms.

A request to execute a service raises the Ad-hoc Edge cluster instantiation. The initiating device makes use of nodes distributed registry based on etcd [21]

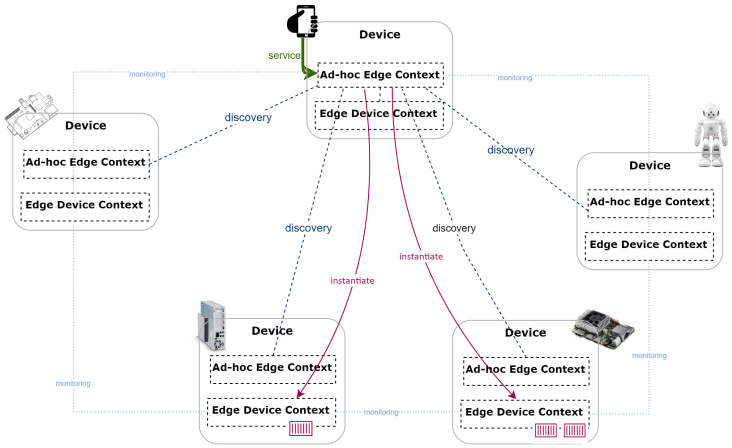

Figure 3: Ad-hoc Edge Cloud Architecture Flow of Events

in order to dynamically discover other available nodes in the specific location. The Admission Control mechanisms determines the acceptance or rejection of the initiating service based on the placement options it can find out of the characteristics of the service to be executed and the current status of resources currently available to the infrastructure. This process is presented in Section 6. In the case a feasible placement option has been identified by the Admission Control process, the different service components are instantiated into the correspondent Edge resources by means of Edge Device Context Components in each node. At that point, Service Management processes take responsibility of monitoring the availability and performance both resources and executing services. In the event of remote node failure for instance, due to node churn, the Service Manager manages the reallocation of the service component to a different available node by executing the Admission control process for the specific part of the service. In addition, it is able to handle additional deployment adaptations such as conducting the addition or removal of services components instances raised by performance observation and elasticity monitoring. This process is illustrated in **Figure 3**.

6. Admission Control/Service placement for Ad-hoc Edge Cloud

[22] describes the admission control problem for Cloud computing as the mechanism for deciding whether or not it is worth to admit a service into a Cloud, and in case of acceptance, obtain the optimum allocation for each of the components that comprise the service. Admission control and resource scheduling in loosely-coupled distributed systems such as Grid and Cloud computing systems has been an area extensively researched [23, 24].

Specifically in Ad-hoc Edge Computing Infrastructure, difficulties in the scope of Admission control for the Edge infrastructure of non-dedicated resources are not anticipated to be based on the characteristics of services to execute but due to volatility of resources, as described by Park [25]. In this work it is recognized that unpredictability of Edge resources (such as mobile devices) is recognized to have the following problems: unstable wireless connection, limitation of power capacity, low communication bandwidth and frequent location changes. These issues significantly increment levels of resource churn continuous process of resource enrolment and un-enrolment that Ad-hoc Edge Computing Infrastructure needs to handle.

Specifically, two factors are considered to significantly influence resource management in the context of Ad-hoc Edge Cloud:

- Stability in the resource availability: Placement decisions in Ad-hoc Edge Cloud has to take into account analysis of historical information on resource availability in order to determine if a service can be accepted, and which is be the most adequate assignment among available resources for this service taking into account its requirements and the available resources characteristics. Both Admission control and Service management are designed to support churn and subsequent resource volatility with the aim of ensuring as much as possible reliability of the overall infrastructure. With this aim we make use in this work of the Node quality concept defined in [26]. Node Quality concept defines the predicted probability of a node be available in a certain time slot based on its historic behaviour of

connections and disconnections to the Ad-hoc Edge Infrastructure.

- Available battery levels in the contributed resources: Energy scarcity in IoT Edge devices is a problem largely studied in the context of IoT [27, 28] and mobile cloud computing [2]. Specifically in Mobile Cloud Computing, devices energy optimisation often used as a motivation for off-loading computational loads to external clouds in order to produce energy savings in the mobile device. Although this is not the main focus approach in this work, it is clear that the available level of energy in the resource is a factor that cannot be neglected in the services component placement decision to take place in the admission control and service management.

In order to understand in detail this problem we will first analyse the Resource availability protocol which makes resources accessible to the Ad-hoc Edge Infrastructure. Then we will analyse Service life-cycle in Ad-hoc Edge infrastructure to introduce the phases of this lifecycle in which allocation decisions take place. Finally, we will present Ad-hoc Edge Cloud admission control mechanism and its associated Node Quality prediction model.

6.1. IoT Device Availability Protocol

In this context we identify the following phases for host availability in the Ad-hoc Edge infrastructure. This protocol describes a particularization of the general protocol for resource collaboration in P2P described in [29]. The defined phases are defined below and depicted in Figure 4:

6.1.1. Publication

In this phase an IoT Edge Device is made available to the Ad-hoc Edge infrastructure. Devices are described by means of a defined resource specification language that is able to support heterogeneity on the typologies of resources to take part in the infrastructure. A minimal resource description provides information with regards to characteristics and resources of the host in the following terms:

Static Device characteristics: These are device characteristics that do not change over time. It includes elements such as description of the processors architecture and their amount and characteristics, memory capacity, Operating System and supported application middleware, such as container execution support.

Dynamic Device Characteristics: These are device related to the device capacity related to its load, which will change over resources operation time. They describe the resource on terms of available different processors usage, memory and internal storage, active network protocol, upload and download network transmission rate, as well as, battery levels in a certain point of time. This way in the Ad-hoc Edge Cloud infrastructure an IoT Edge Device host is described as two collections of attributes, static and dynamic attributes, which characterise an available node in the infrastructure. $h = \{static_{characteristics} = (s_1 = v_1, s_2 = v_2, .s_i = v_i), dynamic_{characteristics} = (d_1 = v_1, d_2 = v_2, .d_i = v_i)\}$.

In addition to these, depending on the specific usage scenario other characteristics could be of interest such as GPS coordinates or readiness to execute a specific programming language.

6.1.2. Registration

This phase refers to the need of keeping track on all available resources to a certain Ad-hoc Edge infrastructure. Given the envisaged distribution of the proposed architecture in Section 5 this process involves the registration of the published node into the Logical connectivity layer Nodes Distributed Registry. The selected mechanism for this module is a Distributed Hash Table implemented by means of etcd distributed storage. This allows the distributed management and automated replication among all nodes in the infrastructure relying on a well-known and widely used P2P system. At this point of time, the registered node starts to be monitored, gathering information about its dynamic characteristics and also connection and disconnections to the Ad-hoc

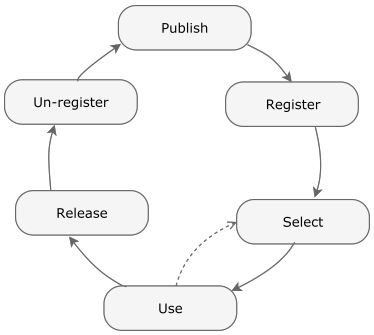

Figure 4: Resource Availability Phases.

Edge Cloud infrastructure in different time periods. This information is the keystone in which to develop the resources availability prediction model that will be presented in Section 6.4.

6.1.3. Select

Selection occurs at the time a user makes a service execution request to the Ad-hoc Edge infrastructure. It consists in the Admission control process of identifying from the available host resources those that are candidate to execute the requested service. In more detail, this phase refers to the process of the resource appearing in the list of candidates to execute a service (or a part of it), given provided service requirements. This process is presented in detail in upcoming Section 6.3.

6.1.4. Use

Once the resource is selected as a potential executor of a service (or a part of it) the corresponding service components have to be instantiated into the target device. Due to nature of devices considered, diverse factors, as previously exposed, affects its availability to be part of the Edge infrastructure. Node churn and resource failures have to be considered in order to address resource volatility. Therefore, use phase has to implement a continuous monitoring of the operation of the execution and put in place mechanisms for effective workload migration

in case of node failure. At the same time, continuous monitoring will also allow the continuous adaptation of the size of the set-up to the defined application needs and user resource contribution constraints by triggering elasticity events. All operational adaptations of the service execution in a given resource will require of repetition of phase Selection with the objective to replace or acquire new resources.

6.1.5. Release

Release of service resources will be associated to the finalization of a service a resource executes. This phase will consider the clean-up of all occupied resources due to the service deployment so to ensure its used capacity is released and remains available to other services.

6.1.6. Un-register

Resources registered in the Ad-hoc Edge infrastructure will be continuously monitored. A resource that cannot be contacted for a certain period of time will be considered no longer available to participate in the Ad-hoc Edge infrastructure and therefore un-registered after a certain period of time.

6.2. Service Lifecycle in Ad-hoc Edge infrastructure

Figure 5 Ad-hoc Edge Service Life-cycle describes the proposed Service Life-cycle in the Ad-hoc Edge infrastructure. This life-cycle provides an adaptation of the Life-cycle of a job inside a IaaS cloud defined by Ghosh in [30] extending it to the concept of Edge/Cloud service as appliance considering multiple inter-related components. This lifecycle describes the flow of actions since a specific IoT Edge device makes a request for service deployment until this service is up-and-running in the Ad-hoc Edge infrastructure. In this context, it is observable that Admission control placement decisions take place in two specific phases of this flow: at initial placement decision for the complete service and as consequence of an adaptation decision, to allocate one or more service parts.

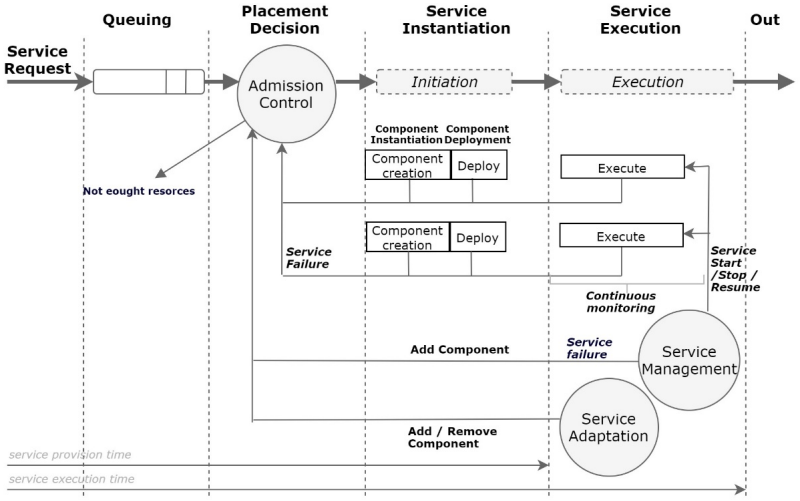

Figure 5: Ad-hoc Edge Service Life-cycle

6.3. Admission Control mechanism formulation

The Admission Control mechanism sort outs the problem of responding if a service s can be executed in a Ad-hoc Edge Computing infrastructure E, and if so which is the set of resources that can host it, R.

- If in the certain point of time in which the service execution request is done there arent hosts which available capacity is sufficient to fulfil the execution requirements of the service, $R = \emptyset$.
- Otherwise, R is a ordered list of hosts h.

The Admission Control mechanisms considers that an Ad-hoc Edge Computing infrastructure E is constituted by a set of computing nodes, hosts $H = \{1, ..., N_h\}$. As introduced in section Section 6.1, each host h is characterised by two kind of attributes: static and dynamic attributes. Among others these attributes include: Computing Capacity by quantifying the number of available processors, their architecture and characteristics C_h; Available Memory M_h; Available Disk D_h; Available Battery level B_h and Quality of a node, Q_h. Q_h is based on the assessed node quality of a node based on its historical behaviour

with regards to connections and disconnections to the Ad-hoc Edge infrastructure.

As introduced in Section 4 the Admission control considers that a service instance s to be executed in the Ad-hoc Edge Infrastructure belongs to a Service template which pose a set of requirements to the host to execute the service. In its minimal form, these will include: Minimum computing capacity C_t; Minimum available memory M_t; Minimum available disk D_t; Processor Architecture Requirement $Arch_t$ and Operating System Requirement OS_t.

In order to admit a service to be executed in the Ad-hoc Edge Cloud, the Admission Control process executes two main steps: Filter and Ranking.

Filter initial step is quantitative. It filters from all available resources those offering the sufficient device capacity in terms of dynamic characteristics (namely CPU, memory and storage) for the different parts that compose a service.

The second step, Ranking, focuses on Qualitative characteristics.It prioritizes requirements according to critical aspects in resource management in Ad-hoc Edge Cloud infrastructure: node stability and nodes current battery level. Nodes stability is estimated by means of the Node quality value taking into account the time the device has been part of the infrastructure. Table 1 presents the parameters considered in each IoT Device which takes part in the Ad-hoc Edge infrastructure and its role in Admission control mechanism while upcoming sections formulate in more detail the above mentioned two-step process.

6.3.1. Filter

The filter process determines from all available hosts $H = \{1, ..., N_h\}$, the set of host H' which available resources and characteristics meet the services requirements. These are set in terms of computing capacity, memory and disk are sufficient to host the service, as well as, the host meet the execution environment characteristics determined processors architecture and operating system, according to the requirements exposed by the service template T. Therefore H' is the subset of H hosts which:

Parameter	Category	Type	Threshold	Rationale
% of connected time	Dynamic	Ranker, Quality	Maximize	Priories connection stability
# of disconnections	Dynamic	Ranker, Quality	Minimize	Penalize host with higher number of disconnections
Computing Capacity, Memory and Disk	Dynamic	Filter, Capacity	Available values	If the host does not have enough capacity to host the service, it is discarded
Processor Architecture	Static	Filter, Capacity	Defined host value	If the host does not posses the required value, it is discarded
Upload / Download speed	Dynamic	Filter, Capacity	Maximize	If the host does not have a certain value of upload/download network capacity, it is discarded
Battery level	Dynamic	Filter, Capacity	Maximize	If the host does not have a minimum level of battery, it is discarded
Battery level	Dynamic	Ranker, Quality	Maximize	Devices with higher battery level available are considered better.

Table 1: IoT Edge Device parameters classification

- Computing Capacity is greater than service s minimum computing capacity, $C_h \geq C_t$
- Available Memory is greater than service s minimum available memory, $M_h \geq M_t$
- Available Disk is greater than service s minimum available disk, $D_h \geq D_t$
- Processor Architecture corresponds to s requirement, $Arch_h = Arch_t$
- Operating System matches to s requirement, $OS_h = OS_t$

- And the available battery level is above a certain battery level threshold B_E established for the Ad-hoc Edge Computing infrastructure E, $B_h \geq B_E$

If there are not hosts fulfilling the above mentioned capacities and characteristics the $H' = \emptyset$. Therefore $R = \emptyset$ and the admission control process for service s resolves that the service can not be hosted in the Ad-hoc Edge Cloud infrastructure at current point of time.

6.3.2. Ranker

If $H' \neq \emptyset$ the Ranker node returns the list of hosts $R = \{1, ..., N_R\}$ in H' ordered according to the assessed quality of the node, Q_h. As it will be presented in detail in the upcoming section, Q_h value is calculated making use of historical values on resource availability to the infrastructure: percentage of connected time and number of disconnections since the initial registration of the host. In addition to this, in order to determine the overall node quality, the percentage of available battery of the host is used with the purpose of balancing from all available hosts that are above a certain battery level, those with closest values to 100 percent battery levels and adequate historical behaviour, so give current devices status a prominent role into devices selection decision.

6.4. Resource Availability prediction

As it has been previously mentioned, the Resource Availability prediction in this work extends the concept of Node Quality presented in [26]. This builds on top of two parameters:

Percentage of connected time: which represents the percentage of time the node has been available to the infrastructure from its initial registration to the Ad-hoc Edge Cluster. This value is expressed as a value between 0 and 1, where higher percentages of connected time result on better node stability and reliability to the overall infrastructure.

Number of disconnections: Reflects the number of disconnections this specific node has had, normalised to the higher number of disconnections from all nodes currently available in the Ad-hoc Edge infrastructure. This parameter provides the sense of balance of the quality of the specific node behaviour with respect to the rest of available nodes in the infrastructure.

In more detail, the Quality of a node represents the probability of a IoT Edge Device to remain connected to the Ad-hoc Edge Infrastructure for a certain period of time [26]. A significant difference among this work and [26] as well as existing literature on Node availability prediction reported in Section 8, is the area of application which has been typically in Volunteer Computing environment. These make use mechanisms to calculate availability prediction forecast the behaviour of available peers in weeks, based on the behaviour on the different week days over the last month. This is be motivated due to the roots of volunteer computing on using spare capacity of desktop computers in enterprise and home environments. This resulted in resource usage patterns in Volunteer Computing which make distinction among working days for determining the time these resources have more probability of remaining idle.

Instead in Ad-hoc Edge Cloud environment we aim to understand the behaviour of complex IoT devices located at the Edge of the network for determining periods in which these edge resources have more probability of having resources available for the Ad-hoc Edge infrastructure. For understanding this behaviour, there are significant difficulties at current state of affairs: lack of real deployments of these kind of scenarios for which we could gather data that permitted us to understand the behaviour on the use patterns of the devices, and not to forget, the fact that this behaviour can significantly differ based on the specific usage scenarios we aim to address.

In relation to IoT devices, usage patterns available literature in this field is very limited [27, 28, 31]. In these, no references to specific time usage patterns could be found. All analysed literature develops mainly on use of energy of devices and is very much oriented to a specific type of IoT device. As means of

example, the most closer example to the kind of IoT Edge device we aim to make use of, is in [31]. This work focuses on analysing the behaviour of SmartWatches users while observing usage patterns of 32 users in 70 days of continuous use. This analysis concludes with the general statement that SmartWatches devices are in idle state 89 percent of the time.

Smartphone usage patterns have been much more deeply studied however number of analysis on mobile phone usage patterns that include references to idle time and time use patterns is also limited [32, 33, 34] . Findings of the available studies conclude that overall a phone is in idle state 89 percent of the time[34] (note that percentage is the same range than the assessed for smartwatches in [31]) . [34] has analysed the activity of Users on smartphones logs of 25 users over 6 months. When focusing at week day behaviour, overall differences between week and weekend days do not appear as significantly different. [33] looks at patterns for mobile data traffic. It studies 3.3 terabytes of data gathered by university of Cambridge using Device Analyzer. This information contains over 100 billion records of 17,000 android devices worldwide and spreads between December 2010 and January 2014. These do not reveal any significant difference on Mobile data traffic among weekend and working days. Otherwise, [32] collects information from 230K mobile apps and 600M daily unique users in 221 countries. [32] focuses on application sessions, concluding that during the weekend, application sessions seem to start and finish later, considering one hour escalation.

More interestingly for our purpose, these works also look at mobile phone day hour patterns from different perspectives: [32] shows that mobile users are more active on the mobile device throughout the day, having the maximum activity during evenings locating the activity peak at 9pm. [33] arrives to similar pattern, stating that although the data traffic is quite stable during the day hours, it significantly increments and peaks around 22:00 hours. Finally [35]studies at temporal distribution of the traffic of apps focusing on more specific temporal patterns. It also shows diurnal patterns with regards to traffic volume and network access time. However, it locates the minimum use around 1AM and

2AM. Showing that they start growing around 4AM, having the peak about noon. This starts decreasing after 3PM and is reduced after 8PM.

In order to overcome the above mentioned difficulties related the availability of data to establish resource IoT Edge resource usage data, we have decided to centre our prediction model on the usage patterns of mobile phones, which patterns on usage data are at this stage much more mature. In fact, mobile phones are the most spread and widely used devices available at the Edge the network. Figures on penetration of Edge devices present also analogies in term of scale with expected distribution of IoT Edge devices.

Based on patterns identified in the three main data sources identified [32, 33, 35] in mobile phone usage patterns, we determine that the appropriate time period to study in this context is based on IoT Edge device behaviour during the last week and considering four different time period over the day to analyse connection and disconnection patterns.

Specifically, these time periods over the day we consider comprehend hours among:

- Day period 1 (DayP1): From 0:00:00 to 5:59:59
- Day period 2 (DayP2): From 6:00:00 to 13:59:59
- Day period 3 (DayP3): From 14:00:00 to 19:59:59
- Day period 4 (DayP4) From 20:00:00 to 23:59:59

This way, in order to predict the probability of an IoT Edge Device to continue to be part of the Ad-hoc Edge Infrastructure we will make use of the data on its behaviour during the last week, capturing this data according to the four periods in which we have divided each of the days, and obtaining a probability of a node to remain connected for a given time period.

For each host h we keep a data structure of disconnection probabilities as an array of 4 positions which stores hosts probability of disconnection of each day period (host $h \in \{1, 2, ...N\}$, $Pr_{hDayPeriodp}$ day period $p \in \{1, ..4\}$) cal-

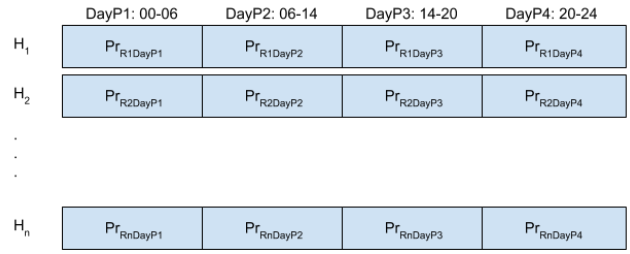

Figure 6: Representation of Data Structure for Disconnection Probabilities

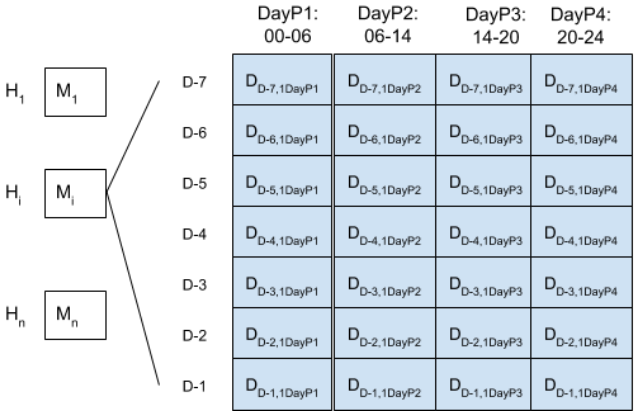

Figure 7: Representation of Data Structure for Storing the disconnection patterns per day period of the last 7 days

culated by the prediction model using the information of the last 7 days. See representation in Figure 6.

Data Structure of disconnection probabilities:

In addition to this, we keep data structure to collect disconnection patterns during the last week of execution for every node. This maintains the disconnection patterns of all nodes in the last 7 days. The data used structure is represented in Figure 7. It is a matrix of 7x4, that It will store for a time period of one week, the number of times a node has changed state (disconnections and connections) per each of the four day periods.

In detail per each host there is a matrix M_h which is the seven position

matrix corresponding to the node h. D is the Day number $d \in \{1, ..7\}$) and day period is $p \in \{1, ..4\}$). $D_{d,p}$ is the number of disconnections experienced by host h in day d period p. This matix is kept updated by a daily process that each day analyses the performance of all nodes, updating the number of disconnections that each host experienced in the previous day aggregated by each of the four time periods of the day. This process makes use of the monitoring information kept by each node.

Therefore, we define the probability of disconnection of a host h in a given day period as analogy to [26] as: $Pr_{h,p} = normalize(\frac{\sum_{d=D-7}^{D} D_{d,p}}{7})$

By using this probability we are able to calculate by means of Algoritm 1 the connection probability of a certain host in given time period as the inverse probability of a node from its initial registration, represented by $t_{registration}$ and the next t_{units} as:

$$Pr_{h,t_{units}} = 1 - (DiscProb(h, t_{units}) * DiscProb(h, -t_{registration}))$$

Algorithm 1 Disconnection Probability

```
1:  procedure DiscProb(h, x)
2:      nPeriods ← numTimePeriods(x)   ▷ Number of time periods within x
3:      sum ← 0
4:      if x > 0 then
5:          for i ← 1 to i = nPeriods do
6:              p ← period(now() + i)
7:              sum ← sum + Pr_{h,p}
8:      else
9:          for i ← 1 to i = nPeriods do
10:             p ← period(now() − i)
11:             sum ← sum + Pr_{h,p}
        return sum/nPeriods
```

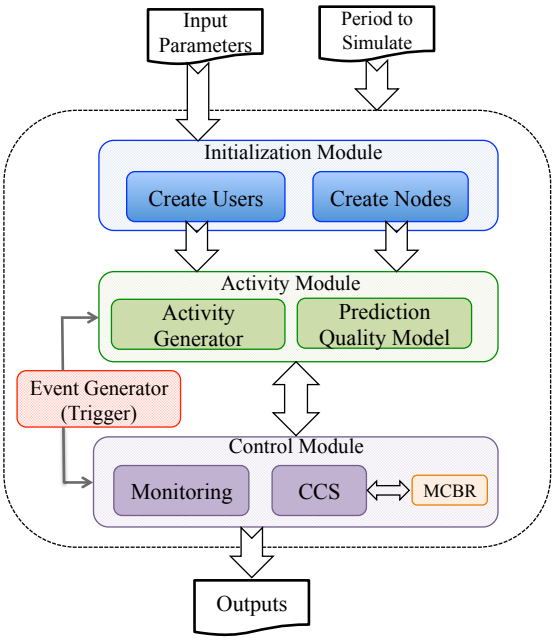

Figure 8: Simulator architecture

7. Evaluation

Evaluation in this work has focused on the assessment of the resource availability prediction model previously presented in section 6.4. With this purpose we have developed a set of experiments making use of a distributed large-scale resource allocation simulator [26].

The simulator allows us to represent large-scale and high dynamic and heterogeneous environments with diverse rates of node churn and diverse nodes configurations. It was implemented in Java Standard Edition 7.0, and its architecture it is shown in Figure 8. As can be seen, it is composed of three main modules: *Initialization*, *Activity* and *Control*.

The first module *(Initialization)* is responsible for initializing the environment. It creates both the users and nodes with their properties. The second module *(Activity)* is made up of two submodules: the *Activity Generator* submodule and the *Prediction Quality Model* submodule. The *Activity Generator*

submodule is in charge of modifying the state of the nodes. They are turned on and off following a probability. We have defined three kinds of nodes in function of their quality: *low*, *middle* and *high*. *Low* nodes have a high probability of disconnection and a low probability of reconnection, while *High* nodes have a low probability of disconnection and high probability of connection. Each of these kinds of nodes behaves differently, and allows us to simulate different scenarios. On the other, the *Prediction Quality Model* submodule is responsible of predicting the quality of each node. The last module *(Control)* is composed of two submodules: the *CCS* submodule and the *Monitoring* submodule. The *CCS* submodule allows to introduce in a easy way own resources algorithms to test in the distributed system. We have implemented the Resource availability prediction algorithm previously described in section 6.4 in this submodule.

All the experiments have been executed in a laptop equipped with Intel i7 core x86-64 bits processor, 8 GB memory and 128GB SSD Disk over 10 days experimentation. In this environment for all the experiments we have represented three different connection patters for nodes, by defining three typologies of Node behaviour, namely High, Medium and Low quality nodes, to which we assign a certain percentage of nodes over the total experiment number of nodes. These percentages determine the number of nodes of each disconnection pattern that will be present during the execution of the experiment. These are represented in Table 2.

Pattern	**High**	**Medium**	**Low**
Nodes_High	60%	20%	20%
Nodes_Medium	20%	60%	20%
Nodes_Low	20%	20%	60%

Table 2: Nodes Connection pattern amount distribution

In order to assess the impact of scale both in terms of nodes and services we have performed the experimentation the execution of up to 200, 400, 600, 800 and 1000 services over 50, 100, 150, 200, 250, 300, 350, 400, 450 and 500 nodes

of the three different node quality typologies distribution described before. All executed services have been constructed with the same configuration, with two replicated components. Overall, the execution of these different service and node combinations have resulted in the execution of 150 experiments findings of which we summarize in the following subsections.

7.1. Node Quality

Figure 9 presents per each of the three types of nodes configurations the total number of nodes created by the experiment for each node category, the number of disconnections of each node quality and the time each type of node has been connected during for the execution of 600 concurrent services. In addition, Figure 10 presents the values of two of nodes characteristics defined in our IoT Edge Device parameters classification used in this experiments: a dynamic parameter, battery level, for which we represent number of nodes with average battery below and above 50 percent; and an static characteristic, processor architectures, for which we represent the number of nodes in the experiment that correspond to each of the four defined processor architectures (x86, arm and the combination of these with pluggable GPUs). These configurations have the purpose of representing the heterogeneity on the characteristics of nodes defined to take part in the Ad-hoc Edge infrastructure and their detailed characteristics.

Based on these nodes configurations, tests performed with execution of 600 services over 50 to 500 nodes obtain nodes qualities aligned to the defined nodes behaviours. Results obtained in these tests are depicted in Figure 11 . High quality nodes obtain node qualities values between 0.86 and 0.90 while these qualities descend to values that range among 0.80 and 0.88 in configurations with 60 percent presence of medium quality nodes, and among 0.77 and 0.82 for high presence of low-quality nodes. It is important to note that the performed experimentation show for 600 services execution increments of quality of percentages among 3 and 5 for High, Medium and Low-quality nodes, however keeping coherent values for probabilities of disconnection and its associated asses node quality values independently of the number of nodes being employed

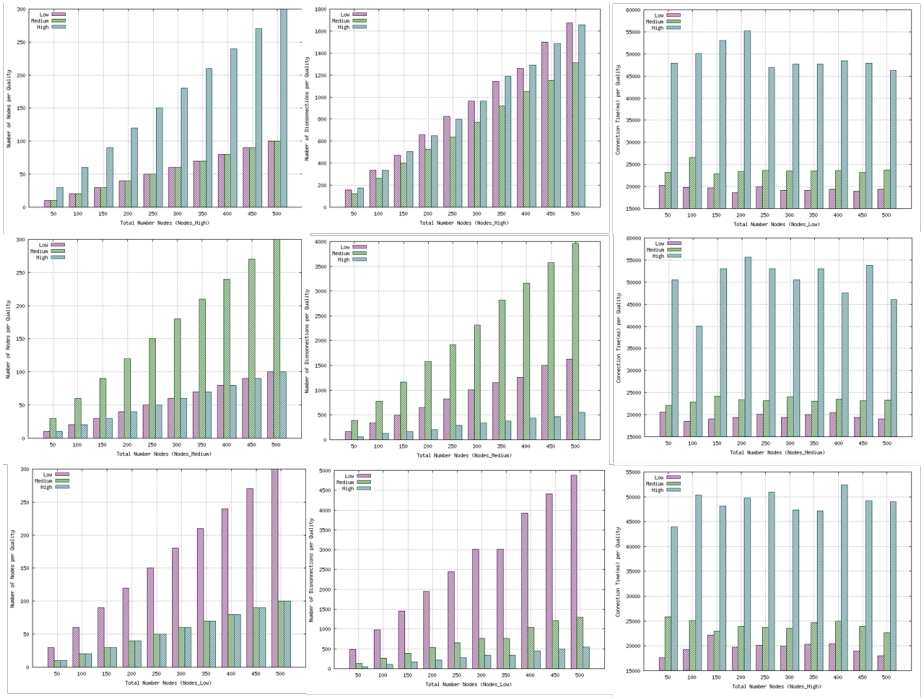

Figure 9: Nodes Number, Number of Disconnections and Time Connected per pattern

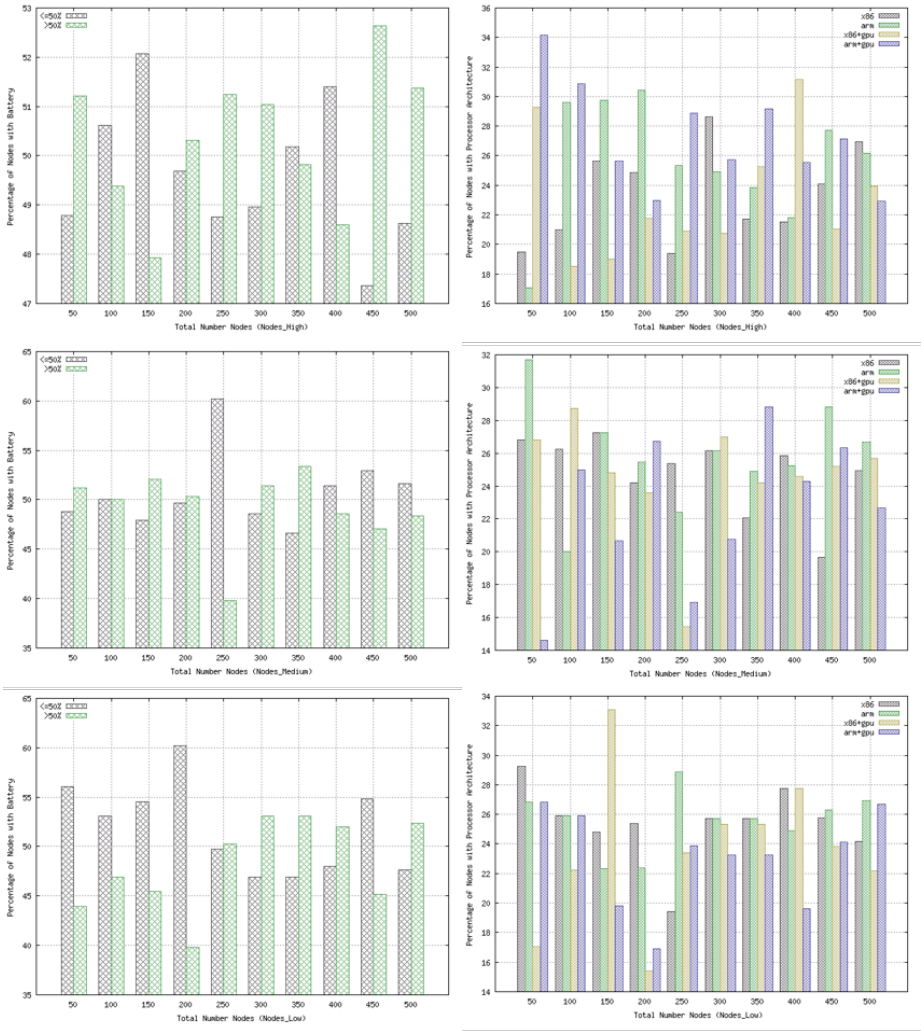

Figure 10: Nodes Characteristics per pattern

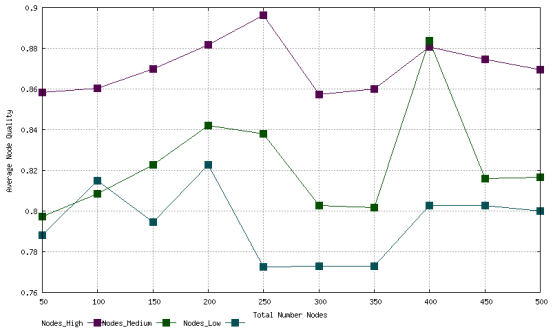

Figure 11: Nodes Quality per pattern

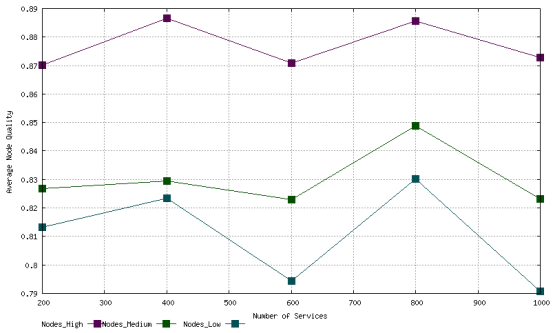

Figure 12: Average Nodes Quality in 50-500 Nodes executing 200-1000 services

in the experiment.

This assessment is further demonstrated when we analyse the performance of obtained average node quality values by generalising services execution figures from 600 services to the execution of 200 to 1000 services over 50 to 500 nodes (Figure 12). Again, we can spot that both node scale and services scale do not influence obtained nodes qualities. Consequently, we conclude that Node Quality is solely determined by the specific node characteristics and its historical behaviour in relation to connection and disconnection to the system.

7.2. Service Quality

The quality of a Service is assessed by Garlanet Simulation Environment[26] as the sum up of the Qualities of the different nodes that execute a service together with normalised values for nodes characteristics. The configuration of Ad-hoc Edge experimentation has considered the execution of two replicas of each service representing two identical service components following the Service Model described in Section 4. In addition to this, the assessed Service Quality take into account the characteristics of the normalised values per each node for computing capacity, memory and available disk, available upload and download connectivity speed and battery level. The values gathered during this experimentation for Service Quality obtained on the execution of 200 to 1000 services over 50 to 500 nodes are illustrated in Figure 13. In this, we observe that maximum Service Quality value is collected on the execution of 1000 services over 500 nodes accounting for 4.5 and minimum values of 2.2 are found in the execution of services with more presence of Low quality nodes. This behaviour is again aligned with previous conclusions on Node Quality and allow us to infer that based on the results of our experimentation the presented Ad-hoc Edge Resource management and its associated mechanism for Node Quality prediction is adequate for managing the expected scale and heterogeneity expected in such dynamic and distributed environment.

8. Previous Works

Resource Availability prediction has been an area extensively studied in Contributory communities and Volunteer Computing. To the best of our knowledge, our work is the first aiming to bring these strategies to the area of Edge computing. Hence, in this section analyses related literature in these two areas: articles which provide mechanisms for resource availability prediction, as well as, works that study placement and scheduling in Edge and Fog computing. An extensive analysis of the state of the art focusing on architectures, challenges

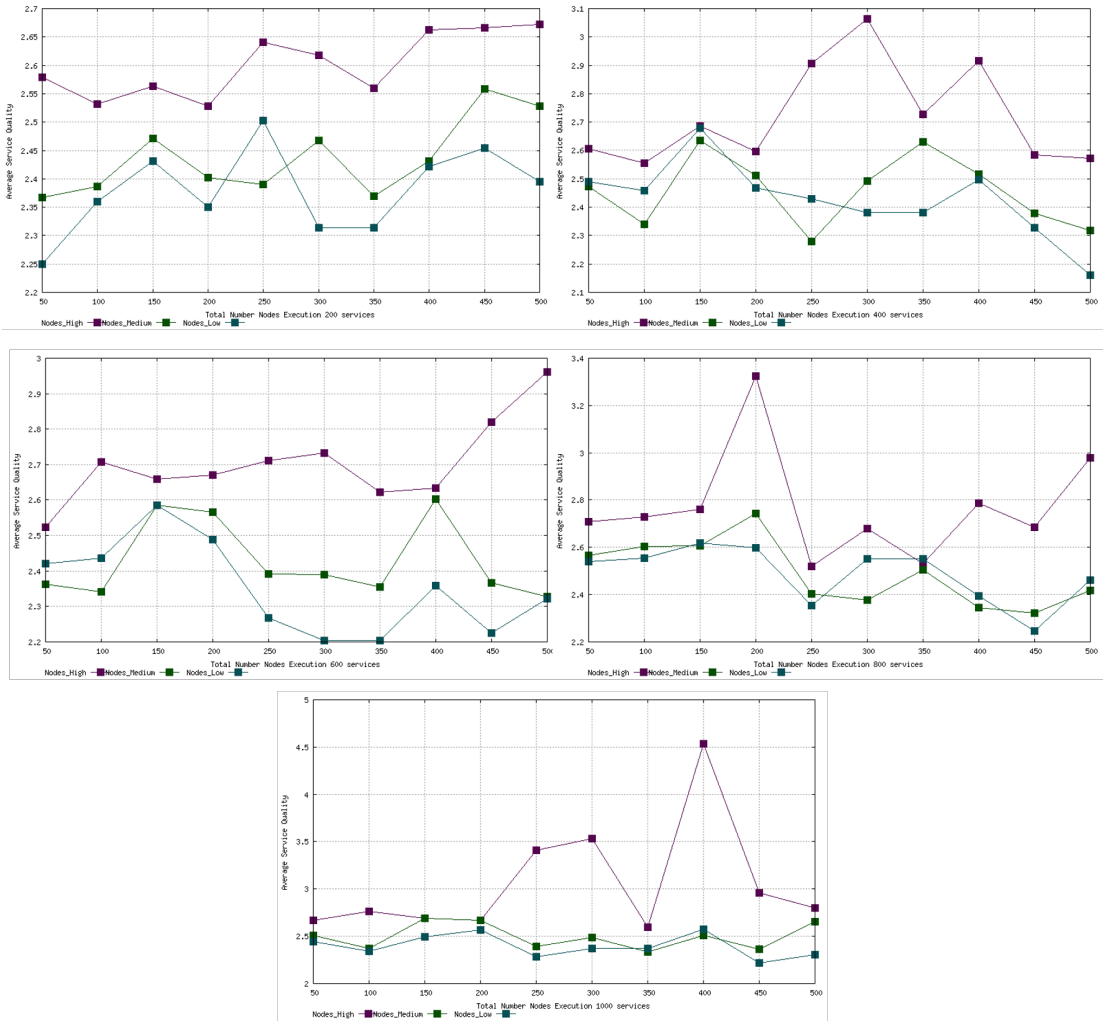

Figure 13: Services Quality per pattern

and approaches to Ad-hoc Edge computing was provided by some of the authors in [2].

As already mentioned in this article [26] provides a multi-criteria optimisation strategy for selection of reliable nodes for hosting services of a distributed micro-blogging social network, Garlanet. In [26] prediction of node availability makes use of the behaviour of nodes in relation to connections and disconnections over the past four weeks to calculate the probability of disconnection of a node in a given week day. Along these lines nonetheless using a different granularity level for representing the availability of a host [36, 37] makes use of the hosts availability per hour for the last week in order to represent resource availability. This two works had available existing traces of 11000 hosts in SETI@HOME collected by means of the instrumentation of BOINC server during seven months period [37]. This allowed to understand day of week and hour of week hosts connection patterns for the constituent hosts in these Contributed Desktop Grids environments. Beyond individual hosts availability analysis, some works have also analysed the collective availability of the infrastructure: [38] focused on providing mechanisms to ensure availability of a collection of a determined number of resources over a given time period. Differently [39] calculated the weekly availability of specific groups of resources. Another interesting research in this area is presented by [40] studying the types of applications that can better take advantage of non-dedicated infrastructures beyond bag of tasks types of applications. Even before all this references and specifically focusing on P2P systems [41] studied the behaviour of Internet hosts in the internet using Gnutella and Napster application software.

In the case of the Ad-hoc Edge infrastructure presented in this paper, the fact that complex IoT devices are not yet that widely spread hinders at this stage the availability of data to be analysed to determine connection and disconnection patterns for these kinds of devices. In order to overcome these difficulties, we have extrapolated the behaviour of available specific edge devices, mobile phones, and made use of its common usage patterns in order to describe our resource availability model, as presented in detain in Section 6.4. This has

driven us to define a mechanism which instead of focussing in connection and disconnection patterns over week days we make use of the resource historical behaviour over four different day periods in a week. We understand that in the long term, this model could be further refined in order to consider the specific behaviour per type of complex IoT Device and for particular IoT usage scenarios.

Research in Fog and Edge computing scheduling and placement has only recently stated to embrace the aspects of edge node availability and churn. Broggi [42] recognises that IoT and Fog nodes mobility and dynamicity has not yet been addressed with sufficient extend and recognises the emerging need of addressing application placement adapted to the phenomenon of node instabilities at the Edge. [43] introduces the concept of mobility in the Edge infrastructure formulating it as User mobility. It studies how user mobility affects the demand that has to be served from Edge infrastructures, so to optimise user application provision bringing it closer to the user. In this context, [43] still considers Edge infrastructures as stationary environments. Differently, [44] is one of the few examples that at this stage it contemplates uncertainty of Edge node availability in fog infrastructure management. It elaborates a centralised environment that the users contact in order to execute services in the fog infrastructure comprised by several fog nodes for which a predefined availability rate has been calculated. It formulates an Integer Program to indicate is a user has to send a service to one of the Edge nodes and the probability of failure of the service execution is used as QoS measurement. Two main aspects differentiate the Ad-hoc Edge Cloud approach and this work: the consideration of a central decision system as well as, our view on the need of formulating resource availability based on the Edge devices current status, given all dynamic aspects that influence it. [45] develops on the placement of Virtual Network Functions in Edge infrastructure nodes for scenarios in which fog nodes are mobile. It models the location of a fog node as Random Waypoint Model formulating different among others movement velocity and probabilities for nodes to be static and paused. While the problem approach in this work is different and complementary to Ad-hoc Edge Cloud, it assesses that higher mobility probabilities drive to lower qualities of

service similarly to our conclusions. Lera [46] elaborates in a Service placement policy that takes into account availability for complex services composed by multiple service components as addressed by Ad-hoc Edge Cloud. This work tackles the node availability system in a completely different approach to Ad-hoc Edge computing. It creates the concept of community of nodes, as a set of devices which are mutually interconnected. By deploying services to devices in communities it aims to avoid service failure due to failure of an specific node, however adding the need of an additional management layer at community level. The concept of community is also present in [47] which analyses user mobility and need for service migration.

The consideration of Edge node heterogeneity in combination to user mobility is addressed in [48] that presents a scheduling mechanism for privacy constrained real-time jobs in Edge micro data centres. Heterogeneity in this work is considered at the level of the different compute capacities available on each node, however no concrete formulation on how to defined these characteristics is provided.

9. Conclusion and Next Steps

It is undoubtable that penetration of connected devices in society presents staggering figures in the last decade. As presented in this work, today connected devices are not only spreading everywhere, but they are quickly gaining complexity in terms of their ability to carry significant compute and storage capacities. This combined with the fact that AI processes are multiplying the need of timely processing at the Edge, reveals the urgent need of exploiting all compute capacity available at the Edge of the network. This presents specific challenges in relation to nodes scale, heterogeneity and availability, specifically in relation to node churn, to resource management practices in this context. Analyzing these issues together with the definition of admission control processes for the defined Ad-hoc Edge infrastructure, together with the evaluation of defined mechanisms for prediction of resource availability have been the core

contribution of this work. In this our evaluation reveal that node behavior in relation to connection and disconnection are key for the overall performance of services provisioned by the infrastructure, as we intuitively understood. Also, at this stage, the data we could use to develop our availability prediction model relates to the behavior of mobile devices. This is due to the fact that still reports on usage patterns of a diversity of complex IoT devices, such as drones, robots and connected cars, are not available. We understand that the diversity of IoT devices and different usage contexts of this devices will facilitate in the future to define more specific prediction models adjusted to the diversities of the common behavior of additional IoT devices in concrete usage scenarios. In addition, it is important to notice, that in this paper we have focused in the analysis of the computational part of decentralized cloud management however we acknowledge that a multitude of concerns remain to be explored both at network, security and data storage levels. Equally, overall service QoS and analysis of applications which could most benefit of exploiting capacity available in distributed and spread in complex IoT devices are areas subject to future study.

References

[1] F. Bonomi, R. Milito, J. Zhu, S. Addepalli, Fog Computing and Its Role in the Internet of Things, in: Proceedings of the first edition of the MCC workshop on Mobile cloud computing, 2012, pp. 13–16. doi:10.1145/2342509.2342513.
URL http://doi.acm.org/10.1145/2342509.2342513{%}5Cnpapers2://publication/doi/10.1145/2342509.2342513

[2] A. J. Ferrer, J. M. Marqués, J. Jorba, Towards the Decentralised Cloud: Survey on Approaches and Challenges for Mobile, Ad Hoc, and Edge Computing, ACM Comput. Surv. 51 (6) (2019) 111:1—-111:36. doi:10.1145/3243929.
URL http://doi.acm.org/10.1145/3243929

[3] AWSGreengrass, Amazon Web Services Greengrass.
URL http://aws.amazon.com/solutions/case-studies/

[4] M. Azure, Azure IoT Edge.
URL https://azure.microsoft.com/en-us/campaigns/iot-edge/

[5] D. Chen, J. Cong, S. Gurumani, W.-m. Hwu, K. Rupnow, Z. Zhang, Platform choices and design demands for IoT platforms: cost, power, and performance tradeoffs, IET Cyber-Physical Systems: Theory & Applications 1 (1) (2016) 70–77. doi:10.1049/iet-cps.2016.0020.

[6] A. J. Ferrer, J. M. Marqués, J. Jorba, Ad-hoc Edge Cloud : A framework for dynamic creation of Edge computing infrastructures.

[7] A. Taivalsaari, T. Mikkonen, A Taxonomy of IoT Client Architectures, IEEE Software 35 (3) (2018) 83–88. doi:10.1109/MS.2018.2141019.

[8] Raspberry Pi.
URL https://www.raspberrypi.org/

[9] S. Tang, A List of Chip/IP for Deep Learning.
URL ttps://medium.com/@shan.tang.g/a-list-of-chip-ip-for-deep-learning-48d05f1759ae

[10] Intel Movidious.
URL https://www.movidius.com/

[11] NVIDIA Jetson.
URL https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/

[12] Google Coral (beta).
URL https://coral.withgoogle.com/

[13] HEADS Project, D3.3. Final Framework of resource-constrained devices and networks, Tech. rep. (2016).

URL http://heads-project.eu/sites/default/files/HEADSD3.3V1.0.pdf

[14] Docker, Enterprise Container Platform for High-Velocity Innovation.
URL https://www.docker.com

[15] Unikernel, Unikernels, Rethinking Cloud Infrastructure.
URL http://unikernel.org/

[16] Katacontainers, Kata Containers.
URL https://katacontainers.io/

[17] GVisor, gVisor, Container Runtime Sandbox.
URL https://github.com/google/gvisor

[18] D. Stutzbach, R. Rejaie, Understanding Churn in Peer-to-peer Networks, in: Proceedings of the 6th ACM SIGCOMM Conference on Internet Measurement, IMC '06, ACM, New York, NY, USA, 2006, pp. 189–202. doi:10.1145/1177080.1177105.
URL http://doi.acm.org/10.1145/1177080.1177105

[19] Kubernetes Pod Overview.
URL https://kubernetes.io/docs/concepts/workloads/pods/pod-overview/

[20] AWS CloudFormation.
URL https://aws.amazon.com/cloudformation/

[21] Etcd Discovery service protocol.
URL https://etcd.io/docs/v3.3.12/dev-internal/discovery{_}protocol/

[22] K. Konstanteli, T. Cucinotta, K. Psychas, T. Varvarigou, Admission Control for Elastic Cloud Services, in: 2012 IEEE Fifth International Conference on Cloud Computing, 2012, pp. 41–48. doi:10.1109/CLOUD.2012.63.

[23] S. Bagchi, Admission control and scheduling of remote processes in loosely-coupled distributed systems, Computers & Electrical Engineeringdoi:10.1016/j.compeleceng.2013.08.013.
URL http://www.sciencedirect.com/science/article/pii/S0045790613002243

[24] D. Chang, G. Xu, L. Hu, K. Yang, A network-aware virtual machine placement algorithm in mobile cloud computing environment, in: Wireless Communications and Networking Conference Workshops (WCNCW), 2013 IEEE, 2013, pp. 117–122. doi:10.1109/WCNCW.2013.6533325.

[25] J. Park, H. Yu, K. Chung, E. Lee, Markov Chain Based Monitoring Service for Fault Tolerance in Mobile Cloud Computing, 2011 IEEE Workshops of International Conference on Advanced Information Networking and Applications (2011) 520–525doi:10.1109/WAINA.2011.10.
URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5763554

[26] Multi criteria biased randomized method for resource allocation in distributed systems: Application in a volunteer computing system, Future Generation Computer Systems 82 (2018) 29–40. doi:10.1016/j.future.2017.11.039.
URL https://doi.org/10.1016/j.future.2017.11.039

[27] L. Reinfurt, U. Breitenbücher, M. Falkenthal, F. Leymann, A. Riegg, Internet of Things Patterns for Devices, in: Ninth international Conferences on Pervasive Patterns and Applications (PATTERNS) 2017, Xpert Publishing Services (XPS), 2017, pp. 117–126.

[28] N. Mousavi, B. Aksanli, A. S. Akyurek, T. Š. Rosing, Accuracy-resource tradeoff for edge devices in Internet of Things, 2017 IEEE International Conference on Pervasive Computing and Communications Workshops, PerCom Workshops 2017 (2017) 581–586doi:10.1109/PERCOMW.2017.7917627.

[29] H. M. N. D. Bandara, A. P. Jayasumana, Collaborative applications over peer-to-peer systems-challenges and solutions, Peer-to-Peer Networking and Applications 6 (3) (2013) 257–276. doi:10.1007/s12083-012-0157-3.

[30] R. Ghosh, K. S. Trivedi, V. K. Naik, D. S. Kim, End-to-End Performability Analysis for Infrastructure-as-a-Service Cloud: An Interacting Stochastic Models Approach, in: 2010 IEEE 16th Pacific Rim International Symposium on Dependable Computing, IEEE, 2010, pp. 125–132. doi:10.1109/PRDC.2010.30.
URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5703236

[31] E. Poyraz, G. Memik, Analyzing power consumption and characterizing user activities on smartwatches: Summary, Proceedings of the 2016 IEEE International Symposium on Workload Characterization, IISWC 2016 (2016) 219–220doi:10.1109/IISWC.2016.7581282.

[32] Describing Patterns and Disruptions in Large Scale Mobile App Usage Data, Proceedings of the 26th International Conference on World Wide Web Companion (2017) 1579–1584.

[33] A. A. Bhih, P. Johnson, M. Randles, Diversity in Smartphone Usage, Proceedings of the 17th International Conference on Computer Systems and Technologies 2016 - CompSysTech '16 (June) (2016) 81–88. doi:10.1145/2983468.2983496.
URL http://dl.acm.org/citation.cfm?doid=2983468.2983496

[34] A. Shye, B. Scholbrock, G. Memik, P. A. Dinda, Characterizing and Modeling User Activity on Smartphones: Summary, Proceedings of the ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems (2010) 375–376doi:10.1145/1811099.1811094.

[35] Q. Xu, J. Erman, A. Gerber, Z. Mao, J. Pang, S. Venkataraman, Identifying diverse usage behaviors of smartphone apps, Proceedings of the 2011

ACM SIGCOMM conference on Internet measurement conference - IMC '11 (2011) 329doi:10.1145/2068816.2068847.
URL http://dl.acm.org/citation.cfm?doid=2068816.2068847

[36] D. Lázaro, D. Kondo, J. M. Marqus, Long-term availability prediction for groups of volunteer resources, Journal of Parallel and Distributed Computing 72 (2) (2012) 281–296. doi:10.1016/j.jpdc.2011.10.007.
URL http://dx.doi.org/10.1016/j.jpdc.2011.10.007

[37] D. Kondo, A. Andrzejak, D. P. Anderson, On correlated availability in internet-distributed systems, Proceedings - IEEE/ACM International Workshop on Grid Computing (2008) 276–283doi:10.1109/GRID.2008.4662809.

[38] A. Andrzejak, D. Kondo, D. P. Anderson, Ensuring collective availability in volatile resource pools via forecasting, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 5273 LNCS (contract 34084) (2008) 149–161. doi:10.1007/978-3-540-87353-2-12.

[39] K. Ramachandran, H. Lutfiyya, M. Perry, Decentralized resource availability prediction for a desktop grid, CCGrid 2010 - 10th IEEE/ACM International Conference on Cluster, Cloud, and Grid Computing (i) (2010) 643–648. doi:10.1109/CCGRID.2010.54.

[40] Discovering Statistical Models of Availability in Large Distributed Systems: An Empirical Study of SETI@home, Parallel and Distributed Systems, IEEE Transactions on 22 (11) (2011) 1896–1903. doi:10.1109/TPDS.2011.50.

[41] J. R. Douceur, Is remote host availability governed by a universal law?, ACM SIGMETRICS Performance Evaluation Review 31 (3) (2003) 25–29. doi:10.1145/974036.974039.
URL http://portal.acm.org/citation.cfm?doid=974036.974039

[42] A. Brogi, S. Forti, C. Guerrero, I. Lera, How to Place Your Apps in the Fog - State of the Art and Open ChallengesarXiv:1901.05717.
URL http://arxiv.org/abs/1901.05717

[43] L. F. Bittencourt, J. Diaz-Montes, R. Buyya, O. F. Rana, M. Parashar, Mobility-Aware Application Scheduling in Fog Computing, IEEE Cloud Computing 4 (2) (2017) 26–35. doi:10.1109/MCC.2017.27.

[44] N. Daneshfar, N. Pappas, V. Polishchuk, V. Angelakis, Service Allocation in a Mobile Fog Infrastructure under Availability and QoS Constraints, 2018 IEEE Global Communications Conference, GLOBECOM 2018 - Proceedings (2019) 1–6doi:10.1109/GLOCOM.2018.8647488.

[45] C. Mouradian, S. Kianpisheh, M. Abu-Lebdeh, F. Ebrahimnezhad, N. T. Jahromi, R. H. Glitho, Application Component Placement in NFV-Based Hybrid Cloud/Fog Systems with Mobile Fog Nodes, IEEE Journal on Selected Areas in Communications 37 (5) (2019) 1130–1143. doi:10.1109/JSAC.2019.2906790.

[46] I. Lera, C. Guerrero, C. Juiz, Availability-aware service placement policy in fog computing based on graph partitions, IEEE Internet of Things Journal 6 (2) (2019) 3641–3651. doi:10.1109/JIOT.2018.2889511.

[47] S. Filiposka, A. Mishev, K. Gilly, Community-based allocation and migration strategies for fog computing, IEEE Wireless Communications and Networking Conference, WCNC 2018-April (2018) 1–6. doi:10.1109/WCNC.2018.8377095.

[48] K. Fizza, N. Auluck, O. Rana, L. Bittencourt, PASHE: Privacy Aware Scheduling in a Heterogeneous Fog Environment, Proceedings - 2018 IEEE 6th International Conference on Future Internet of Things and Cloud, FiCloud 2018 (2018) 333–340doi:10.1109/FiCloud.2018.00055.