A Development Approach for Collective Opportunistic Edge-of-Things Services

Roberto Casadei[a], Giancarlo Fortino[b], Danilo Pianini[a], Wilma Russo[b], Claudio Savaglio[b], Mirko Viroli[a]

[a] *Alma Mater Studiorum—Università di Bologna, Italy*
{roby.casadei,danilo.pianini,mirko.viroli}@unibo.it
[b] *Università della Calabria, Italy*
{g.fortino,w.russo}@unical.it, csavaglio@dimes.unical.it

Abstract

Technological advances have recently fostered the Internet of Things vision, in which systems of situated entities perceive and act upon the world, and interact with one another to provide novel kinds of services, which are inherently cyber-physical, increasingly contextual and opportunistic in nature, and possibly span different scales and domains. The requirements of such IoT applications, however, pose significant non/functional challenges to engineering efforts, mitigated by emerging computing paradigms. On the infrastructure side, Cloud, Fog, and Edge Computing provide virtualised, on-demand, elastic resource provisioning – at the distant data centres, Network core and Edge – supporting the abstraction and scalability needs of IoT settings while also altogether giving options for QoS-driven trade-offs. However, despite intense research in these fields, there is still a gap of approaches supporting the engineering of dynamic, heterogeneous smart environments, such as those involving "collectives" of devices coordinating in a complex fashion to provide "global" services. In this paper, we integrate the Aggregate Computing and Opportunistic IoT Service models and propose a full-fledged approach for the engineering – from analysis to simulation – of complex "Edge of Things" applications. We compare by simulation two deployment targets for the same collective application: one centralised/Cloud-based, and the other decentralised/Edge-based. We discuss the trade-offs each one introduces, and we draw recommendations on application-driven choices of the appropriate deployment.

Keywords: Internet of Things, Edge Computing, Smart City, Opportunistic Services, Aggregate Computing

1. Introduction

Recent advances in Information and Communication Technology have fostered the *Internet of Things (IoT)* vision, where heterogeneous entities situated in our environments (i.e., cities, places, homes, and bodies) and capable of sensing, actuation, and processing, are interconnected with one another and with other infrastructural components in order to enable novel kinds of applications and services. Such new services tend to exploit what the environment and infrastructure have to offer, and are increasingly *cyber-physical*, *context-aware*, *opportunistic* and *collective*, with the staunch goal of unleashing the full Internet of Things ecosystem potential [17, 18, 16]. In other words, the environment itself is made *smart* through Smart Objects (SOs) and innovative services based on those which also span multiple scales and domains [30]—with examples including energy-optimised management of smart buildings [24], augmented logistics capabilities in smart ports [39], infrastructural maintenance in smart cities [13], energy-efficient monitoring in ecological scenarios [32], cognitive robot-supported smart manufacturing [22], etc.

However, the "layout" of smart environments is often based on *heterogeneous infrastructure* (e.g., access points, base stations) and *heterogeneous devices* with different physical forms, communication capabilities, energetic needs, storage capacities, and processing power. Concerning storage and processing, the *Cloud Computing* paradigm extends the capabilities of SOs by allowing them to *offload* data and computations to distant data centres. Moreover, the recent *Fog* [7] and *Edge Computing* [33] paradigms promise to provide the benefits of Cloud Computing without incurring in its problems (e.g., high latency, data rate consumption etc.), by bringing virtualised infrastructure and on-demand, elastic resource provisioning closer to end-devices, at the Edge or core of the network. Actually, Edge Computing is not an alternative to Cloud Computing but extends and complements it, by inclusively supporting resource-constrained devices and near real-time scenarios, and paving the path to Quality of Service (QoS) improvements. However, all this heterogeneity pushes additional complexity to engineering endeavours and stresses traditional approaches, methods and programming paradigms [15].

In this paper, we present an approach for seamlessly supporting the development (from the analysis to simulation) of collective, opportunistic Edge-of-Things (EoT) applications [14]. In particular, we model Smart Environments as ecosystems hosting opportunistic IoT services that may exploit Edge and cloud resources and may be collective in nature. The approach integrates the Opportunistic IoT Service model and the Aggregate Computing paradigm, whose conceptual alignment has been shown in [9]. As proof of concept, in a Smart City scenario, we show a crowd detection application which is first modelled as an Opportunistic IoT Service, then designed by following either a centralised/Cloud-based and a distributed/Edge-based approach, and finally implemented and simulated according to the Aggregate Computing paradigm. Simulation results show how a distributed/Edge-based approach can provide improved reactivity of collective applications at the expense of precision due to data from different SOs being processed in different Edge Servers.

The novel contributions with respect to our previous work [9] are threefold:

1. a unified domain model for describing Collective and Opportunistic IoT Services to be provided both at the core and at the Edge of the network;
2. a computation and communication performance model for enabling a quantitative analysis of the performed simulations;
3. a detailed comparison and result analysis of the centralised/Cloud-based and distributed/Edge-based approaches, in order to analyse their pros and cons in supporting EoT applications.

The remainder of the paper is organised as follows. Section 2 provides a brief overview about the SO-based IoT vision – highlighting the importance of SOs in service provision and eliciting the features characterising an Opportunistic IoT Service – as well as about main concepts of emerging paradigms such as Cloud, Fog and Edge Computing. Such background is functional to introduce in Section 3 what we intend for EoT applications and our approach for their full-fledged development. This approach integrates Opportunistic IoT Service models and Aggregate Computing paradigm thus providing theoretical concepts and practical tools for engineering EoT applications from the modelling to the simulation phase. A crowd detection EoT application based on a real world urban mass event data set is developed according to the proposed approach in Section 4. We first describe it through the Opportunistic IoT Service model and then evaluate its performance in terms of responsiveness (measured as mean packet delay) and precision (measured as the number of SOs which are alerted of not entering crowded areas) in different scenarios of centralised/Cloud-based and decentralised/Edge-based computing. Finally, conclusions are drawn and future work delineated.

2. Background

2.1. SO-based IoT and Opportunistic Services

A Smart Environment can be defined as a cyber-physical environment in which a collection of heterogeneous devices elaborate data and interact with each other and people in a continuous interplay. In the IoT vision, such devices or SOs are everyday artefacts augmented with computing, communication, sensing, actuation and storing functionalities but, at the same time, constrained by limited hardware resources (CPU, RAM and storage), physical dimensions, and contextual factors (sharing of limited data rate, difficulty in energy harvesting, market-imposed pricing, etc.) [15]. On the basis of both physical inputs and data from on-off-site sources, conscious of the surrounding environment and able to act upon it, SOs eventually provide novel and advanced applications across different scales and domains. Indeed, SOs are increasingly embedded into the global information network as active participants in business, logistics, information and social processes, wherever and whenever needed and proper. Moreover, SOs are not only useful by their individual capabilities, but also through the intelligence that emerges from cooperation and negotiation among connected SOs [8].

On these premises, it is not surprising that the promising impact of such novel class of applications or *IoT services* on every aspect of our daily routine is revolutionary and with not entirely predictable consequences. In details, differently from conventional computing services (e.g., web-services) IoT services tend to exploit SOs functionalities and what the environment and infrastructure have to offer, so as to result inherently *(i) cyber-physical*, involving both disembodied and embodied agents interacting in non-trivial ways, *(ii) context-aware*, leveraging any implicit or explicit information about the current location, identity, activity, and physical condition of the involved stakeholders, *(iii) co-located*, being simultaneously available for different stakeholders sharing local cyber-physical resources, *(iv) dynamic*, being not a-priori created, *(v) transient*, lasting for a temporary time or existing until certain conditions are met, and *(vi) collective*, providing high-level features that result from coordinated activity of (possibly large-scale) "aggregates" (also known as "ensembles") of devices.

The aforementioned features are crucial to actually typify IoT services and their opportunistic characteristic of tactically exploiting transient conditions and resources. Therefore, such considerations lead to the definition of "Opportunistic IoT Service" [16, 18] as an *interface that allows an IoT entity to be engaged, under specific constraints and pre/post-conditions, in a temporary, contextualised and localised usage relationship. The service provision impacts the service provider(s) and service consumer(s) (and, in some cases, third parties indirectly involved to the service provisioning) by modifying their properties and/or their status.* Alongside an outstanding potential, however, the provision of inherently cyber-physical, increasingly context-aware, and opportunistic in nature IoT services from heterogeneous and resource constrained SOs pose significant functional and non-functional challenges. Indeed, an IoT service deployment phase is typically complex, time-consuming, and error-prone, involving not only software distribution but also the configuration of (even thousands of) heterogeneous devices according to their specific resources and surrounding environment [17]. Therefore, IoT services demand for full-fledged development methodologies and proper computing paradigms so to be thoughtfully modelled, designed, implemented and simulated before their actual deployment.

2.2. From Cloud, Fog and Edge Computing towards Edge of Things

In past years, industry and academia have focussed on Cloud Computing paradigm to overcome SO's hardware limitations. By delegating data processing and storage to powerful yet remote data centres, a plethora of high-demanding but time-insensitive SO-enabled applications have been successfully developed, although disregarding their responsiveness, data rate, energy utilisation, and cost. Indeed, handling the massive and fast data generated by the SOs in a centralised manner creates congestion in the *Cloud Servers* and backhaul links, with the resulting latency, reliability, scalability, energy, and price issues.

Recently, the explosive growth in various access devices, the even rising IoT applications' demands as well as the ballooning end-users' expectations in terms of Quality of Service and Experience (QoS and QoE) [19] have jointly motivated huge industry investments and research interest on the network Edge. In particular, both the Fog [7] and Edge Computing [33] paradigms push some intelligence from the Cloud Servers closer to data sources and hence to end-users, following decentralised and often two-/three-tiers architectural models. Indeed, by shifting the function of centralised cloud computing into the SOs of networks there is a drastic reduction in the latency, data rate utilisation and energy consumption, resulting in a smooth IoT service provision. According to that, the IoT landscape is managed in terms of sub-networks (e.g., Local Area Networks) where SOs locally interact and perform lightweight computing before connecting to a powerful node a.k.a. *Edge Server*, serving both as base station, to collect data and perform tasks too much demanding for a single SO, and as gateway to mitigate SOs heterogeneity of different data formats and communication protocols [2]. Both SOs and Edge Servers can transmit tasks and data streams for batch processing to the Cloud Server, and such offloading [1] is typically regulated by an *Orchestrator or Control Node*, which monitors the resource availability of the whole system and dispatches tasks according to certain policies.

More recently, in order to establish a mutually beneficial, transparent and inter-dependent continuum of (latency sensitive, reliable, cyber-physical and opportunistic) IoT services involving both heterogeneous devices and different stakeholders across different domains, the Edge of Things (EoT) [14] has arisen as the seamless integration among the Cloud, Fog, and Edge Computing paradigms. Indeed, some functions are naturally better suited at the network core while others at the Edge: inspecting such a dynamic trade-off is an open research line.

3. Developing Edge of Things Applications: Aggregate Computing for Collective Opportunistic IoT Services

For a number of reasons, IoT services promise to be notably more complicated, heterogeneous and large-scale than conventional ones. As discussed in Section 2, the Edge Computing paradigm ideally enables time-sensitive, context-aware, and co-located applications by providing low latency, scalability, and distributed computation, even in presence of resource-constrained devices at the network's Edge. However, from a programming perspective, a computing paradigm is also required that provides useful features (global-to-local mapping, behaviour compositionality, abstraction, etc.) to realise, in practice, dynamic, collective and opportunistic applications. On this basis, we present an approach for seamlessly supporting the development (from the analysis to simulation) of this kind of Edge of Things applications by modelling them as Opportunistic IoT Services (see Section 3.1), by designing them through the Aggregate Computing paradigm (see Section 3.2) and, finally, by implementing and simulating them (see Section 3.3). The proposed approach follows the process shown in Figure 2.

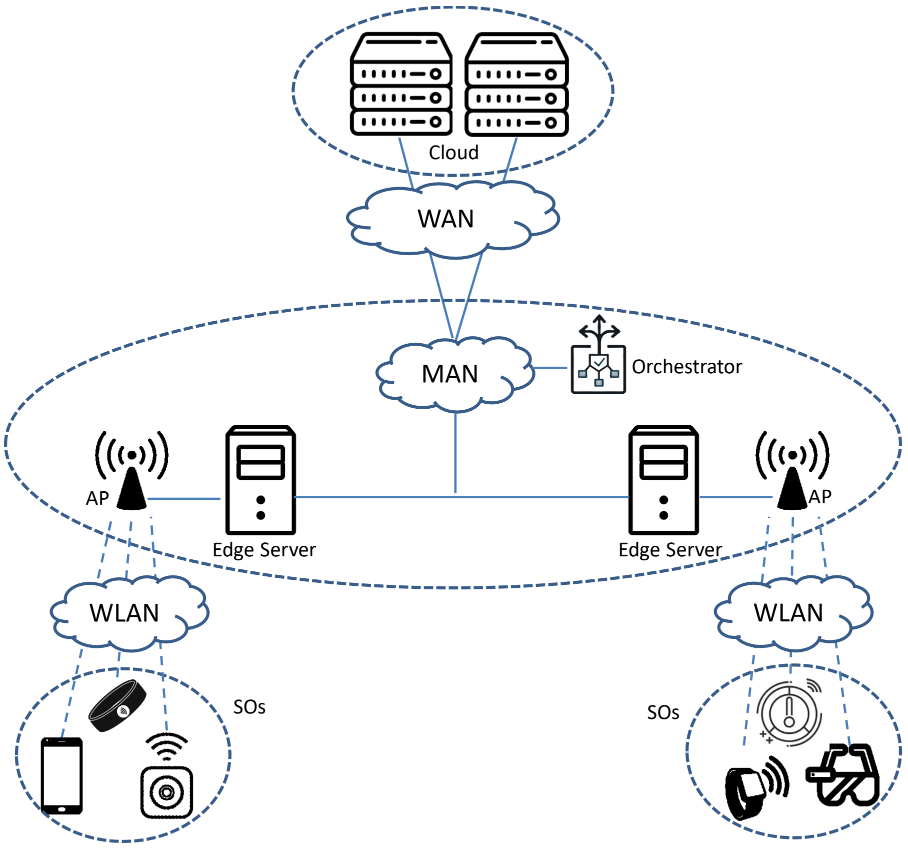

Figure 1: System architecture.

3.1. Modelling: Edge-oriented Collective Opportunistic IoT Services Model

The IoT domain model shown in Figure 3 elicits the main entities involved in the provision of an Edge-oriented, Collective Opportunistic IoT Service. This high-level representation outlines the key components (and relationships among them) of service-oriented IoT applications; hence, it provides a descriptive framework which is particularly useful for the initial analysis phase. In particular, the IoT domain model comprises the following classes of entities:

- *IoT Entity*: any subject that, according to its own attributes (indicated as “features”) and cyber-physical capabilities (indicated as “functionalities”), provides and/or consumes an IoT Service. For a more detailed

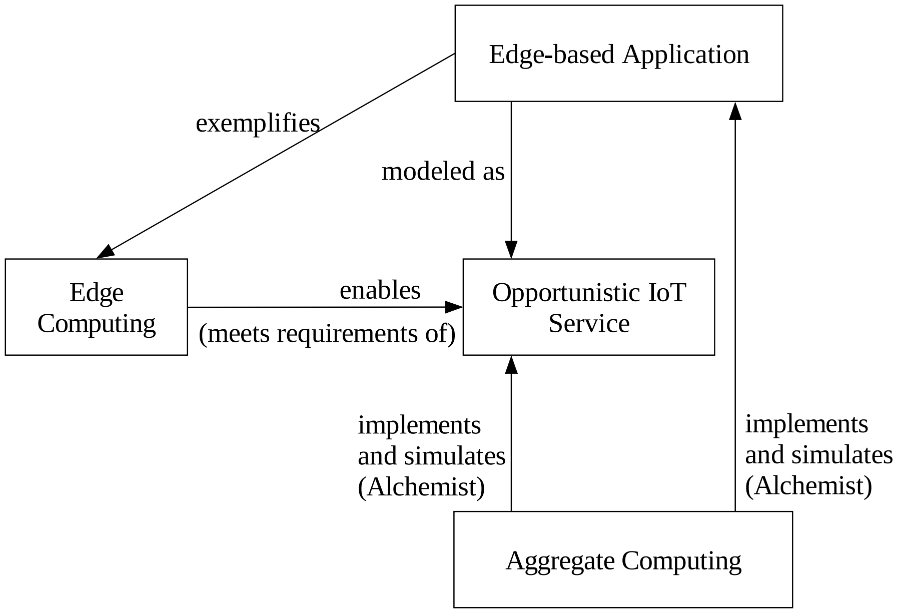

Figure 2: Process of the approach.

modelling, IoT Entities can be classified into three subcategories: Humans, Smart Objects, and Computer Systems (in their turn comprising Edge Servers, Cloud Systems, and Access Points, according to the Edge-oriented computing architecture). IoT Entities are inherently networked and can constitute collections defined as *IoT Ensemble*;

- *IoT Environment*: the physical and non-augmented environment (a lake, a wood, an agricultural field, etc.) in which IoT Entities and physical elements (e.g., trees, obstacles, and weather phenomena) are co-located during the service execution;

- *IoT Service*: a cyber-physical service provided by an IoT Entity. Opportunistic features of any IoT Service are reported by the Service Profile and Service Model which enable its accurate description, automatic discovery, composition, and fruition. The Service Profile represents a high-level description and contains main attributes characterising the IoT Service itself and its relationships with the IoT Entities and the IoT Environment involved in the service provision (e.g., functional requirements for a valid IoT Service execution, the resulted events with associated service quality parameters). The Service Model, instead, describes in detail how the service works, providing information (in terms of Preconditions, Effects, Inputs, and Outputs) about the operations that concretely contribute to realising the IoT Service. If collectively provided by an IoT Ensemble, an IoT Service becomes a *Collective IoT Service*;

- *Context*: dependencies among IoT services and both IoT Entities and IoT Environment. Indeed, service provision is expected to exploit any implicit or explicit information regarding IoT Entity, IoT Environment, or other IoT Services.

The presented IoT domain model extends the one reported in [9] with an explicit account of the *collective dimension* of opportunistic, service-based IoT applications, which is expressed in terms of a specialised kind of Opportunistic IoT Services, defined "collective" in the sense that it is provided and consumed by an IoT Ensemble (i.e., a collection of IoT Entities).

3.2. Design: Aggregate Computing—an Engineering Approach for Collective IoT Services

Aggregate Computing [5] is an engineering approach for collective adaptive systems and smart environments—suitable for Opportunistic IoT Services, as shown in [9]. Its distinguishing feature is a *global stance* on system design and programming: the developer directly targets a distributed *collective* of devices as a whole (also called an *aggregate*), like it was an individual computational machine, through an *aggregate program* expressed as a *functional composition* of building blocks of collective behaviour. Ultimately, the aggregate behaviour will emerge by a complex set of computation and interaction acts carried out by individual devices, which is driven by a *context-aware, local interpretation* of the aggregate specification.

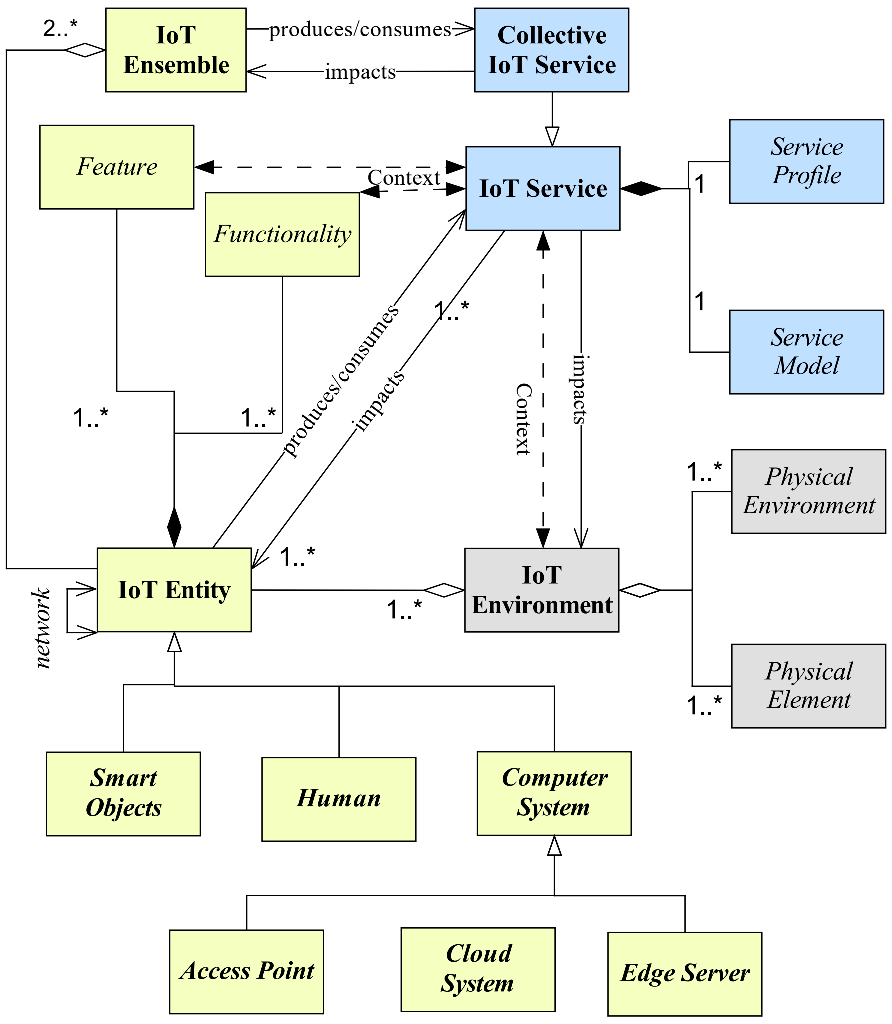

Figure 3: Edge-oriented Aggregate Computing-based IoT domain model.

By a structural point of view, an *aggregate system* is just a *logical network* of nodes, where communication links derive from an application-specific *neighbouring relationship*. Typically, there is a correspondence (i) between a logical node and a physical device, such that the former inherits the position in space of the latter, and (ii) between logical and physical neighbourhoods, such that the logical links map network connectivity. An aggregate system is also usually *situated* into some *environment*, which is abstracted in each node through a logical set of *sensors* and *actuators*. The behaviour of an aggregate system is given by having each node repeatedly: (i) sample the environment, to update its context; (ii) compute the aggregate specification according to the local operational semantics; and (iii) perform deliberate actuations—this is overall called a *(local) computational round*. Interaction is continuous, driven by the aggregate specification, and possible only among neighbours.

The fundamental characteristics of Aggregate Computing make it a suitable paradigm for modelling complex situated coordination and adaptive, self-organising behaviour in Edge-of-Things applications. First, it directly supports decentralised computation and asynchronous, neighbourhood-driven communication, allowing systems to scale to large numbers of devices. Second, when targeting a capillary network of devices immersed into some environment, it provides abstractions for "programming the space" – due to its roots in spatial/field-based computing [36] –, as in the pervasive/ubiquitous computing vision, and supports a set of situated activity algorithms that leverage the spatio-temporal and locality features of systems. Note that the ability to relate and manipulate elements (both people and devices) according to their spatio-temporal relationships [26] is particularly important in forthcoming scenarios like EoT and opportunistic computing. Third, thanks to its abstract model, Aggregate Computing supports different deployment strategies [37], ranging from fully decentralised, peer-to-peer to multi-layer architectural styles involving Edge/Fog/Cloud layers; moreover, it can in principle support adaptivity of the execution strategy of aggregate systems, for fault-tolerance (e.g., switching to physical neighbouring interactions when remote connectivity is absent) or improved QoS (e.g., by moving computations and interactions to cloud or fog nodes in order to apply various sorts of optimisations) [37], possibly driven by opportunistic infrastructure. To see how this is possible, it is sufficient to note that the logical nodes can be deployed as "avatars" to machines different from the corresponding physical devices, which ultimately must be responsible only for providing sensor data and controlling actuators. More details of why this is desirable and how it might work in practice can be found in [37].

In summary, Aggregate Computing provides declarative abstractions for self-adaptive, collective opportunistic computing that are paramount for novel, dynamic scenarios such as those emerging in the context of smart cities, IoT and Edge Computing [11].

3.3. Implementation and Simulation Strategies

In our approach, a Collective Opportunistic IoT Service is first implemented – using aggregate computing techniques – and then simulated to verify both its functional correctness and its non-functional characteristics. Simulation is a key step to assess the behaviour of a service in a range of virtual environments – capturing the salient, expected real-world circumstances – before the deployment phase, which is time/cost expensive, especially for the large-scale scenarios we consider.

The implementation of a service consists of two main parts: (i) the aggregate program expressing the logic of system behaviour, and (ii) the software platform that deals with sensor/actuator management, communication, and scheduling of computational rounds. Currently, support for aggregate system development is provided by two projects: Protelis [28] and Scafi [10]. An aggregate program is written in an aggregate programming Domain-Specific Language (DSL) and compiled/interpreted into a set of local, device-wise instructions executed by a proper "virtual machine" in a functional, deterministic way against the local context, which is kept up-to-date by the platform. Protelis [28] provides an *external DSL* [38] (implemented in Xtext [6]), targeting the Java Virtual Machine (JVM); it has a C-/Java-like syntax, is duck-typed and provides bidirectional integration with JVM-compliant code. Scafi [10] has the goal of providing a seamless integration of the Aggregate Programming paradigm in the mainstream JVM platform; accordingly, it provides a strongly-typed *internal DSL*, embedded into the Scala programming language [25]. Concerning platform support, Protelis includes a set of interfaces and base abstract classes to be implemented, capturing required capabilities of the host device. By contrast, Scafi provides an actor-based middleware [37, 12] supporting two architectural styles: a (i) fully decentralised, peer-to-peer model and (ii) a centralised model where a server mediates communications (allowing, e.g., to model logical neighbouring relationships). Both projects provide simulation support, e.g., through incarnations of the Alchemist meta-simulator [27]. In its essence, a simulator for aggregate systems needs to (i) control the scheduling of computational rounds, and (ii) control the update of local contexts against which rounds are executed; this second step involves handling communications (including, e.g., neighbourhood interaction, network properties), device evolution (e.g., state, local failures), and environment evolution (e.g., physical configuration, dynamics).

Simulation is of paramount importance in at least two phases in the design of aggregate-based Edge-of-Things applications. Simulation is usually the best tool (often, due to the complexity of the target deployments, the only tool) available for inspecting how the system works. Since the results depend in general on the perception and communication of situated devices, it is hard to get a picture of how the system works, and under which conditions, without a simulation of a number of actual interacting devices. Usually, however, further details are included in the simulation once designers are confident that the system complies to its requirements. These details are often related to the final deployment, and include network aspects. For instance, in the case of a distributed sensing application, a first step would be verifying that the aggregation and processing of the information works correctly. Once this first step is completed, further requirements could be injected; for instance, an accurate simulation of packages lost due to network jamming (e.g. with varying antenna power), several displacements of the sensing devices, or a comparison between different network deployments (e.g. with a different count of SOs). In a sense, simulation is used first as a tool to gather guarantees about *functional* requirements, as well as a rapid prototyping support tool; and then used as a means to acquire insights on performance and other *non-functional* requirements, by executing on a more detailed model.

4. Case Study and Simulations: An Edge-of-Things Application for Mass Urban Events

In this section, we discuss an example evaluation of an existing aggregate application for large scale urban events on different deployment targets on the Edge and on the cloud. Our goal is to evaluate the expected performance of an application which showed promising results, evaluating under which conditions we would achieve good performance. The core application has been presented in a previous work [9]: it is a crowd detection and steering application for large scale urban events. The application has been conceived and developed in the framework of aggregate computing. For the sake of simplicity, in this work we reuse the crowd detection aggregate algorithm, but we turn off the steering system. In fact, we are interested in evaluating the reactivity of the application and its cost on the infrastructure, rather than the efficacy of the steering system.

4.1. Scenario Modelling

Our case study leverages data from a real-world mass event, during which, by means of the official event app, 1497 high-quality, anonymised GPS traces of attendees were recorded [3]. The IoT domain model of Section 3.1 is instanced on the case study in Figure 4. The *CrowdSafety service* exploits *Smart Lamps*, *Smart Cameras* and *Smart Traffic Lights* deployed around *Vienna*'s *Bridges*, *Squares* and *Streets* and aimed to monitor overcrowded zones (*Crow Tracking* process). All the data are hence elaborated by a *Data Processing Centre* and, when a zone crowding is reaching the concerning levels, a warning to *Citizens* close to such zone is spread (*User Alert* process). The *Context* information required are the Citizen position, the estimated density of the monitored zone and its topology, since bridges, squares and streets have different concerning levels. In this case study, we do not consider coordination *among* Edge Servers or the adoption of Orchestrators to regulate activity of the Edge Servers (e.g., by dynamically tuning density thresholds for crowd danger), though this could be useful to enact global-level, application-specific policies. As per any well-written aggregate program, the application can work under the assumption

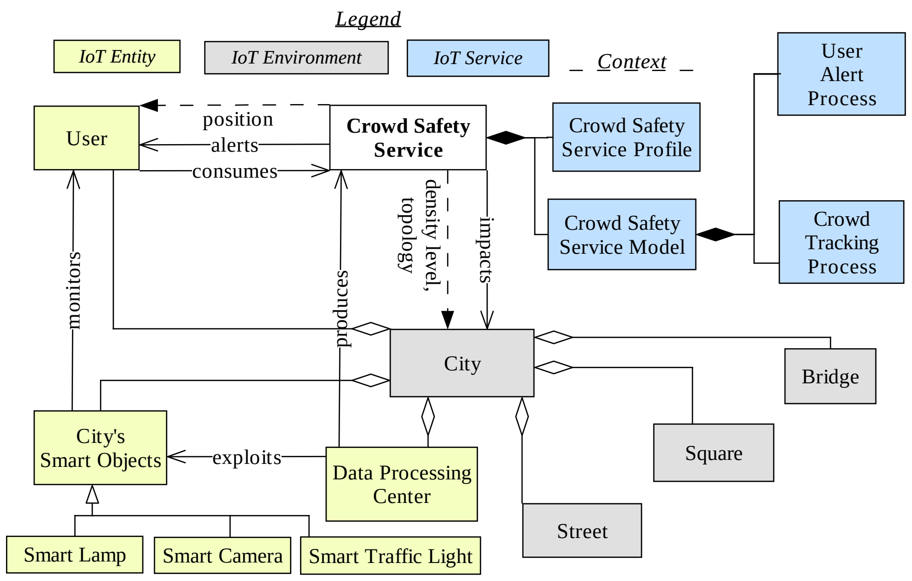

Figure 4: CrowdSafety Service model.

that nearby devices have means to communicate. Due to the inherent model of aggregate programming, the specific communication means of devices, whether purely P2P, mediated by the cloud, or happening at some intermediate level (such as on edge servers), it is entirely transparent to the software—besides for what concerns its actual responsiveness. In order not to introduce unwanted (and unrealistic) synchronisation, we let SOs compute at a slightly different frequency: $1Hz \pm 5\%$.

In this work, we want to compare the same software as executed in two different deployments:

1. a *Cloud* deployment, where single devices communicate by sending data to the cloud via cellular network technology;
2. an *Edge* deployment, where devices communication is mediated by local edge servers, each one coupled with a Wi-Fi access point, with which nearby SOs can communicate with using a Wi-Fi connection. For the remainder of this work, the term access points refers to the networking wireless interface of an Edge Server, and not to stand-alone access points

components.

Relying on the cloud opens the possibility of letting very far away devices communicate. By sending everything to a centralised service, and retrieving any subset of such data, it is in fact possible to emulate a network of very long range P2P devices. The larger the cloud-emulated P2P network range, the greater is the *computational effort* required for the infrastructure to sustain the computation. The cost for this operation is paid primarily in terms of reactivity and network resource occupancy: the round trip time for a message to reach one of its designated neighbours must in fact take into account the time to reach the cloud and get processed. Moreover, the cellular network may face issues in case of a very-high device density, which results in the best case in performance drops (further increasing the round trip time and decreasing the reactivity of the program as a consequence), and in a loss of connectivity in the worst.

On the other hand, the Edge deployment has much shorter round trip times, but features two relevant downsides: *(i)* there must be access point-equipped edge servers scattered around the city for nearby users to connect to; and *(ii)* communication is limited to the areas surrounding such access points, the application will not be able to work appropriately in case none is available in the surroundings of some device.

Approximate localisation of smart objects, which is required for the crowd sensing aggregate program to work, is provided in the Edge case by proximity with the closest Edge Server, and in the Cloud case by GPS positioning. In this work, we assume the GPS to be precise, and we do not simulate drifts between the SOs' position and their reported GPS location.

For this work, we decided not to take into account the case of opportunistic, purely peer-to-peer network. This choice is rooted in the lack of purely peer-to-peer communication means of commercial devices by the time of writing, hence the difficulty in estimating their network properties. Should we consider a real implementation of the software for an upcoming event, we would be realistically limited to the two options in analysis: whether to send everything towards the Cloud, or dislocate access points around the event areas and use them as communication promoters.

4.1.1. Dislocating Edge Servers for an Edge Computing Platform

For the Edge case, it is relevant to discuss the dislocation of access points. Considered the extension of the area where the simulation takes place (about $300km^2$), it is rather unrealistic to propose a coverage made by randomly-displaced access points. Considering a Wi-Fi range of 100 meters, this would result in about 30000 access points to cover the area with optimal displacement. At the same time, the “hot spots” of an urban event are usually known to the organisers in advance: if a concert, or the start or finish line of a race is located at some point, it is very likely that a larger crowd will form around those areas. Moreover, organisers of the event may have access to historical data regarding past editions of the same event.

Despite this knowledge, however, access points can not get installed just everywhere: some places may have physical obstacles, or may be not equipped with the necessary technology (usually at least an electricity source and possibly a network port for Internet access). Namely, a certain amount of randomness is part of the displacement process. In order to emulate a “best effort” displacement of access points, we adopted the following strategy:

1. for each location sampled in our GPS trace, we computed a bivariate normal distribution [20] centred in the sample position;
2. we use such distributions as base for a mixture bivariate normal distribution [29], weighting each sample equally;
3. we then randomly sample from the latter distribution the position of our access points-equipped edge servers.

As a result, there will be higher probability of finding an access point where it is actually useful, but, nevertheless, it is also likely that some of them will be positioned in areas with few or no users. This way, we attempt at emulating both knowledge of the scenario by the organisers, and potential mistakes due to the imperfection of such knowledge.

4.1.2. Network Delay Model

In order for us to compare the two potential deployments, it is necessary to define network measures, and, in particular, how messages are delayed depending on deployment and characteristics of the involved hardware. Inspired by [34], we compute our delay d as a function of: a propagation delay p, which is constant and depends on the network deployment; the message size s; and the data rate b:

$$d = p + \frac{s}{b}$$

For Edge communication, we set p to zero, as there is no additional delivery to perform after the data packet has arrived to its destination. At the same time, however, we compute b differently between the Cloud case and the Edge case: we impose the communication data rate with the Cloud to be fixed, while we modulate the data rate of the access points depending on the count of connected SOs: if n SOs are attached to an access point, its actual data rate will be $b = \frac{b_0}{n}$, with b_0 being the nominal data rate of the device. Packet size is evaluated by measuring the size of the outgoing message objects produced by the aggregate language.

4.1.3. Measures and Free Variables

In this experiment, we are interested in four measures:

1. the average network delay time, as a measure of the reactivity of our application;
2. the number of people informed that they are getting close to an overcrowding area, as a metric of precision;
3. the average packet size generated by the application;
4. and, finally, the overall amount of data processed, which we use as indirect metric of energy consumption.

In our model, in fact, we did not include an accurate simulation of SO resource consumption. We can, however, link the overall power consumption of the system to the overall amount of data processed in a time unit by respectively,

	Cloud	Edge
Edge Server count	0	geometric in [10, 2000]m
Wi-Fi range	n.a.	geometric in [5, 500]m
Computation effort	geometric in [10, 1000]m	n.a.
Propagation delay	geometric in $[10^{-4}, 10]$s	0s
Device bit rate	geometric in $[100, 10^{7}]$B/s	

Table 1: Free variables for the two scenarios in exam. Computation effort represents the communication range used by Cloud-emulated P2P network. Geometrically distributed values have been chosen to correctly sample over multiple orders of magnitude. Values sampled from such distribution are not, in general "round" numbers. We used nine different samples per geometric variable.

Edge Servers and the Cloud. In fact, we argue that higher data volumes imply both higher energy consumption by the network apparatuses (included the radio of things participating the system), and by Edge Servers and Cloud CPUs (which ultimately must deal with a higher amount of data). On the other hand, this metric has obvious limitations, such as processing efficiency: it is likely that a Cloud data centre have highly optimised energy consumption and resource sharing via virtualisation, and, as such, it may be more efficient in terms of energy consumed per byte processed than a setup with multiple Edge Servers. This effect is however compensated by the energy use by network apparatuses. Previous studies show that network energy costs for moving data towards data centres may well compare to the energy required to store and process such data [4]. Our goal in this work is not to provide accurate energy estimates for diverse deployments, but rather to get a picture of how energetic requirements may compare from different setups, using the overall processed data as metric.

Free variables are summarised in Table 1, the Cartesian product of every combination is executed ten times with a different random seed. Reported values are the mean of those runs.

4.1.4. Implementation Details of the Case Study

Once the experiment design phase was completed, we implemented the case study according to the strategies described in Section 3.3. Protelis [28] was chosen as language for writing the aggregate program, while Alchemist [27] was selected as simulation platform. Data generated by the simulator was processed with NumPy [35] and xarray [21], finally, matplotlib [23] was leveraged for charting. For the sake of reproducibility, the software project for the experiment is publicly available and released as open source[1]: including: the Protelis code, the simulator configuration, software patches, data processing and chart generation scripts, and execution automation scripts.

4.2. Simulation Results

We first separately evaluate the Cloud and the Edge setups in order to understand how the network and deployment target characteristics influence the reactivity and the performance of the aggregate software.

4.2.1. Using the Cloud

For what concerns the Cloud setup, we are interested in understanding how data rate, propagation delay, and amount of computational effort invested by the Cloud service provider influence the computation performance, reactivity, packet size and power consumption. We tune the computational effort by changing the maximum distance at which the Cloud allows SOs to communicate—namely, the Cloud systems uses, for computing the crowding at a device location, data generated by all devices situated within the maximum allowed range.

Figure 5 depicts how data rate influences the quality of the results and the reactivity of the application. For what concerns the ability of the application to detect overcrowded areas, a larger data rate does not influence the results, which are instead very sensible to the computational effort. In particular, the system requires considering at least the information of SOs within 56 meters to achieve

[1]`https://bitbucket.org/danysk/experiment-2018-information-sciences`

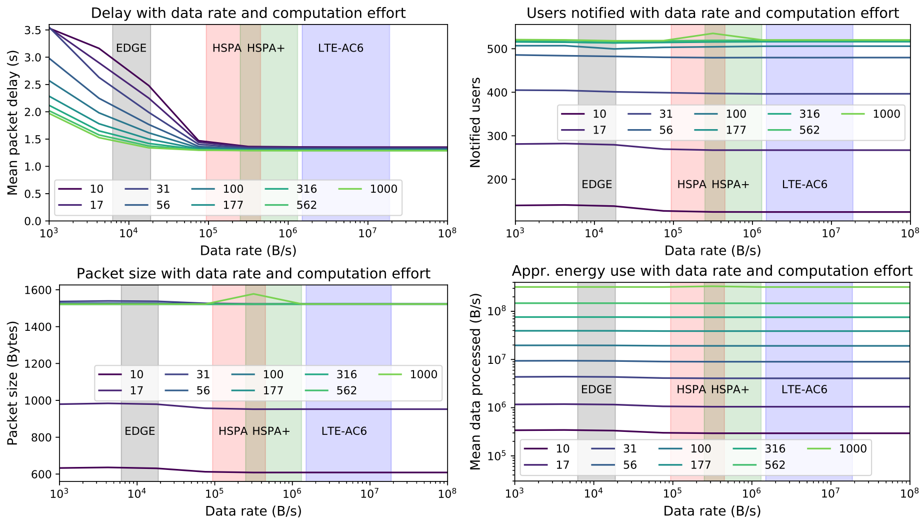

Figure 5: Impact of data rate on packet delay (top left), notified users (top right), packet size (bottom left) and total data processed (bottom right). Different computational efforts (see Table 1) on the Cloud side are depicted as lines of different colours. The associated number represents the virtual range within which the Cloud allows SOs to share data. Numbers are not "round" as they geometrically space the range of interest (see Table 1). The system obtains reasonable performance from 56m+ simulated meters of connectivity. Larger data rates have no direct impact on the number of warned people, however they do have an important effect on the application reactivity: the mean packet delay steadily decreases until the data rate typical of a 3G-HSPA+ connection. Data rates are not correlated with package size nor with increased amount of data generated and processed (and, hence, energy consumed). They are instead both influenced by the effort invested in the computation: packet size reaches a plateau when at 31m of simulated connection range, while power consumption linearly grows with the increase in computation effort.

acceptable results. The importance of larger ranges decreases progressively, reaching a plateau at about 177 meters. Data rate impacts instead the reactivity of the application. Interestingly, the larger amount of data gathered by raising the communication range (and, hence, raising the computational effort required by the Cloud) partially compensates for lower data rates, with more densely-connected scenarios achieving sensibly lower mean packet delays than sparsely-connected ones. In order to achieve low delays, a data rate similar to that offered by current 3G/HSPA+ or better is required. This is seemingly a requirement which is not unrealistic to achieve, however, when many SOs are located within the same area, the performance of the network may degrade consistently (also dependently on the user behaviour) [31]. As such, achieving the typical data rate of some kind of connection cannot be taken for granted during crowded public events. Packet size is not influenced by data rate. Once the system is connected enough (namely, the computational effort is high enough) to allow for the crowd detection system to actually work with acceptable performance, the measured packet size reaches a plateau. Power consumption is not influenced by data rate as well, but grows approximately linearly with the computation effort.

Propagation delay depends mostly on the backbone performance and Cloud data process time. Our experiments, whose outcome is framed in Figure 6, show that on average the effect of propagation delay is higher as higher is the data rate, and on average the propagation delay becomes the bottleneck to application reactivity when it reaches about 500ms. Results on the number of detected overcrowded areas (and notified people) are consistent with the data provided by the experiments performed varying data rate in Figure 5: a decrease in the reactivity of the application does not prevent detection of overcrowded areas, instead, so does lowering the computational effort of the Cloud below a certain threshold. Results concerning packet size and energy consumption return similar results as well: those two metrics are not measurably influenced by propagation delay as they are not by data rate.

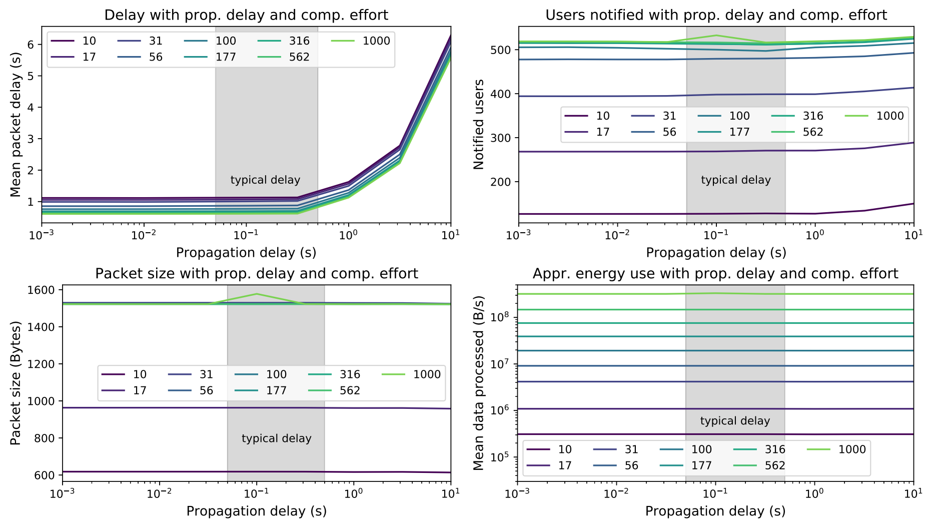

Figure 6: Impact of propagation delay on packet delay (top left), notified users (top right), packet size (bottom left) and total data processed (bottom right). Different computational efforts (see Table 1) on the Cloud side are depicted as lines of different colours. The associated number represents the virtual range within which the Cloud allows SOs to share data. Numbers are not "round" as they geometrically space the range of interest (see Table 1). Results are consistent with those depicted in Figure 5: longer delays do not decrease significantly the count of people warned, however, they can compromise the reactivity of the application, especially if such times get past one second. Similarly to data rate, propagation delay does neither significantly affect packet size nor power requirements, which are both instead influenced by the simulated P2P range: the former reaches a plateau when the computational effort crosses 31 meters, while the latter scales linearly with such value.

4.2.2. Using the Edge

For the Edge setup, we are interested in understanding how the number of available SOs, the wireless connection data rate, and the wireless connection range impact computation performance, application reactivity, packet size, and energy consumption (measured indirectly by evaluating the overall amount of data processed by Edge Servers).

Results for data rate are depicted in Figure 7. Larger data rates are beneficial for both the application reactivity and the crowd detection precision, however, benefits reach a plateau well before the typical performance of a modern Wi-Fi connection is reached: the data rate provided by a modern Wi-Fi connection does not represent a bottleneck for the performance of an aggregate crowd detection application. The behaviour of packet size is interesting, as it reaches a plateau when there are just enough Edge Servers to cover the event area, then decreases due to the presence of multiple servers covering the same area, thus splitting the load. The overall energetic expenditure is not directly influenced by higher data rates, at least for realistic conditions with modern Wi-Fi connections; predictably, however, it grows linearly with the count of installed Edge Servers.

Much more interesting are the results of connection ranges depicted in Figure 8. Data shows a relationship between the number of SOs, their connection range, and the overall system precision and performance. The more SOs are available, the lower is the actual wireless range required to obtain sufficient precision in crowd detection. Such precision reaches a peak, then it slowly decreases progressively with larger ranges. This is likely due to competition for time on air by multiple devices and saturation of the network capabilities of the access points. Packet size behaves similarly to system performance: it reaches a peak approximately corresponding to the best system performance, then slowly decreases. Such small reduction in the amount of data generated per smart object, however, does not compensate for the impact on energy consumption of larger wireless ranges and consequent higher count of SOs connected to each

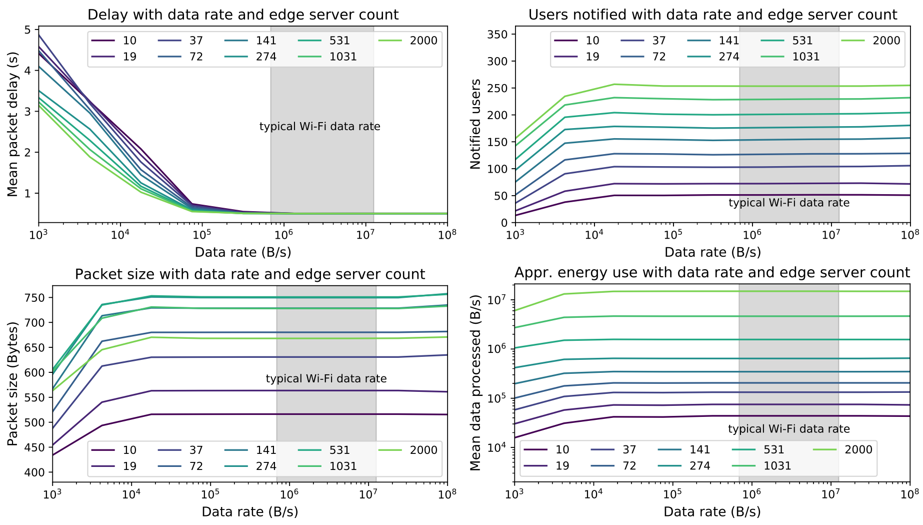

Figure 7: Impact of wireless channel data rate on packet delay (top left), notified users (top right), packet size (bottom left) and total data processed (bottom right). Different Edge Server counts are depicted as lines of different colours, associated to such count. Numbers are not "round" as they geometrically space the range of interest (see Table 1). Data shows that (predictably) the system benefits from a higher number of Edge Servers: the denser the Edge Servers, the better the software performs (more users are warned). Data rate seems to be less important for reactivity: in fact, in the range of data rates to be expected by a modern Wi-Fi connection, there is no appreciable difference. Very low data rate connections may cause longer delays and ultimately negatively affect the system performance, however, this happens in our case only for connections with a data rate lower than 100KB/s. Interestingly, packet size grows until it reaches a peak around 300/500 total Edge Servers. It then decreases. This is likely due to the effect of better coverage of denser areas: for undercovered scenarios, there are simply not enough things connected to each Edge Server to let the packet size grow (packet size is related to the count of other things the aggregate systems is exchanging data with). With 300/500 Edge Servers, the scenario begins to be reasonably covered, with further Edge Server density the load can be split among close ones, decreasing the packet size as well. Energetic expenditure is not directly influenced by data rates, unless they get lower than a dozen KB/s: the system performance is so degraded that little data actually gets to the Edge Servers to be processed, lowering the power consumption. It is, however, an extreme situation, since the system in this condition does not perform adequately, as demonstrated by the top two charts. Finally, energetic expenditure grows approximately linearly with the count of Edge Servers deployed.

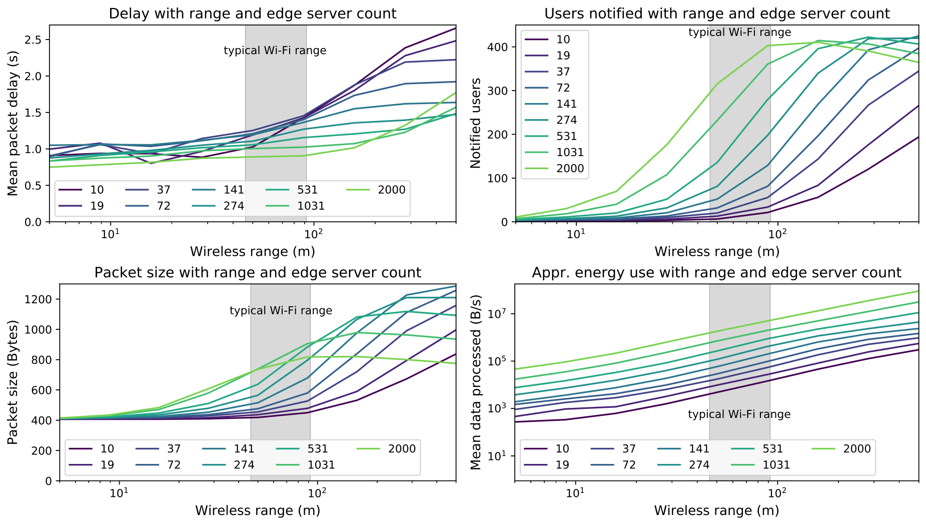

Figure 8: Impact of wireless communication range on packet delay (top left), notified users (top right), packet size (bottom left) and total data processed (bottom right). Different Edge Server counts are depicted as lines of different colours, associated to such count. Numbers are not "round" as they geometrically space the range of interest (see Table 1). Data shows that the system benefits from a higher number of access points, and the lower the Wi-Fi range, the stronger the benefit of a denser coverage. Higher ranges, however, are also correlated with higher delays, as there will be more SOs connected, and the available bandwidth capacity will need to be shared, decreasing reactivity. The behaviour of the packet size is interesting: it grows to a plateau which is correlated to the Edge Server count: denser deployments will feature smaller packet size, assuming a communication range which allows for the same final quality of service. Finally, power consumption grows linearly with increased wireless range: even though packet size (and, hence, the amount of data produced by single smart objects participating the system) does not grow over a threshold, and actually in fact slowly decreases, such effect does not compensate for the much greater count of reachable SOs, that must be taken into account by Edge Servers when processing information.

Edge Server. Energy consumption, here indirectly measured by considering the gross amount of data processed by Edge Servers, linearly grows with higher wireless ranges. For large urban areas, with typical Wi-Fi ranges, a number of Edge Servers in the order of a few thousands seems the most reasonable choice.

In general, however, a good Edge deployment must take multiple factors into account:

- SO count and wireless range must be considered together;
- a shortage of SOs may confer the global system an erratic behaviour;
- different wireless technologies require possibly very different amount of SOs;
- just throwing as many SOs as possible and using as a large connection range as possible is not going to offer the best performance, as, once an efficiency peak is reached, performance then degrades;
- even worse, both the latter factors contribute linearly to the energy expenditure, so increasing any of them should be considered with care.

4.2.3. Cloud/Edge Comparison

Finally, we pick six configurations that we consider to be representative of possible real world deployments, and compare their performance in time. The rationale behind each configuration is summarised in Table 2, while Table 3 shows the parameters of each experiment.

Results are depicted in Figure 9. Cloud deployments can achieve higher precision than Edge deployments, due to the fact that all devices participate the system, while those out of the coverage of the local access point cannot participate in computation in the Edge deployment. On the other hand, however, Edge-based deployments have very high reactivity. In order for a Cloud deployment to achieve similar reactivity, a good, last generation cellular connection with good signal coverage is required. Moreover, Edge setups generate and elaborate much less data per unit of time due to locality, which directly impacts

Name	**Rationale**
Edge-sparse	An Edge deployment with not enough SOs, communication on a classic 802.11 Wi-Fi network
Edge-dense	An Edge deployment with barely enough SOs, communication on a classic 802.11 Wi-Fi network
Edge-denser	A dense, well-designed Edge deployment, communication on a classic 802.11 Wi-Fi network
Cloud-2G	A low performance Cloud deployment with SOs communicating over a typical 2G (Edge) cellular connection (or where the current performance is comparable due to high density)
Cloud-3G	A reasonably performant Cloud deployment with SOs communicating over a typical 3G (HSPA+) cellular connection (or where the current performance is comparable due to high density)
Cloud-LTE	A very performant Cloud deployment with SOs communicating over a high performance 4G-LTE cellular connection

Table 2: Rationale behind the configurations chosen for the comparison.

Name	Edge-sparse	Edge-dense	Edge-denser	Cloud-2G	Cloud-3G	Cloud-LTE
Data rate (b/s)	43M	43M	43M	32K	2.4M	180M
SOs	274	1031	2000	0	0	0
Propagation delay (ms)	0	0	0	1000	316	100
Range/Effort (m)	89	89	89	32	56	100

Table 3: Free variables for the two scenarios in exam.

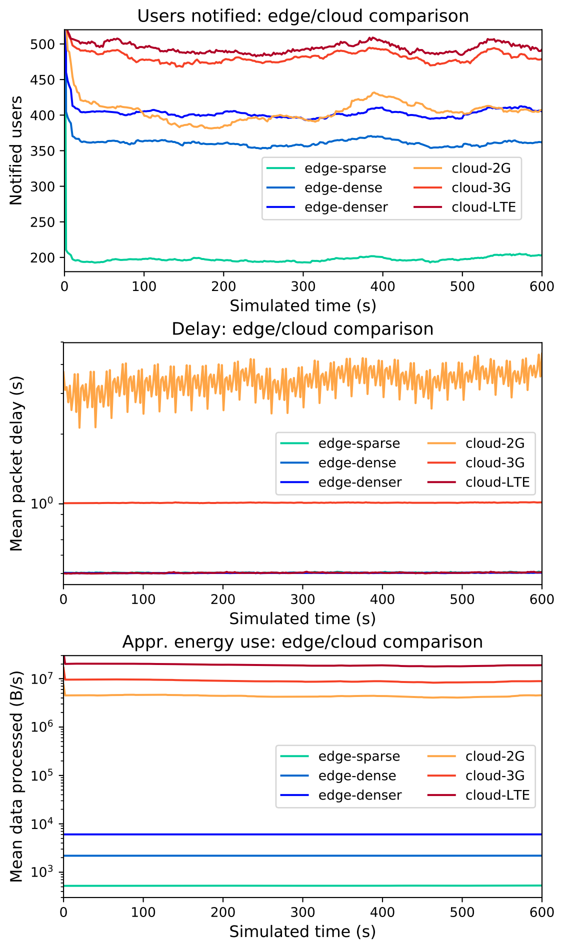

Figure 9: Edge (cold colours) and Cloud (warm colours) compared with different configurations, whose parameters are described in Table 3, and the rationale of the choices is explained in Table 2. Cloud deployments equipped with enough processing power to sustain complex computations can achieve higher precision than Edge deployments, due to the fact that all devices participate to the system, while those out of the coverage of the local access points cannot participate in computation in the Edge deployment. On the other hand, however, Edge-based deployments have very high reactivity. In order for a Cloud deployment to achieve similar reactivity, a good, last generation cellular connection with good signal coverage is required. Data locality typical of Edge deployments, which hinders the quality of the results with respect, allows for much lower (three orders of magnitude) data processed per unit of time. This aspect has direct consequences on the overall power required to make the system work.

energy requirements. According to our data, in choosing between a Cloud or an Edge (or a hybrid) deployment type three very relevant factors must be taken into account: how important is for the application to quickly respond to new situations with respect to how important it is to collect information from every possible source, and how relevant is the power usage and environmental sustainability of the application. The higher reactivity is required and the more power consumption is a concern, the more an Edge-based deployment is preferable, since it generally provides lower packet delays and higher resilience to high SO density, as well as lower power consumption.

5. Conclusion

In this paper, we illustrated a conceptual and operative framework for the development of IoT applications and ecosystems. Specifically, we extended the model in [9] with an explicit account of Collective Opportunistic IoT Services, which are globally offered by dynamic ensembles of entities comprising things, humans, and infrastructural elements including Access Points, Cloud, and Edge Servers. We believe that extending the traditional viewpoint by (also) thinking and working by a "systems" perspective – i.e., considering dynamic aggregate-level behaviour and properties – is key to unlock the potential of the IoT. Also the ability of exploiting the Edge/Fog/Cloud continuum for both enabling and improving the QoS of applications is crucial. Accordingly, in this paper we quantitatively analysed, by means of simulations, the impact of executing Collective Opportunistic IoT Services for crowd management in large-scale urban events (implemented with the Aggregate Computing framework) across different infrastructural configurations. Data from our experiments showed that Edge deployments provide increased reactivity for aggregate applications, whereas Cloud deployments may provide additional precision at the expense of reactivity and centralisation issues.

In future work, we intend to investigate two main directions. One direction focusses on how aggregate computations may be automatically and opportunistically deployed and executed on hybrid Edge/Cloud infrastructures. That is, the goal is to understand how to exploit the declarativity of the Aggregate Programming paradigm through context-aware execution platforms, and accordingly design proper middleware support—hence substantiating the first proposals advanced in [37]. The other direction concerns actual design and programming of Collective Opportunistic IoT Services and Edge-of-Things applications. Aggregate Programming already provides useful abstractions that leverage spatial distribution and context propagation, but additional mechanisms are needed to better capture – also linguistically – opportunistic computation and formation of dynamic teams of devices.

References

[1] Alam, M. G. R., Hassan, M. M., Uddin, M. Z., Almogren, A., Fortino, G., 2019. Autonomic computation offloading in mobile edge for iot applications. Future Generation Computer Systems 90, 149–157.

[2] Aloi, G., Caliciuri, G., Fortino, G., Gravina, R., Pace, P., Russo, W., Savaglio, C., 2017. Enabling IoT interoperability through opportunistic smartphone-based mobile gateways. Journal of Network and Computer Applications 81, 74–84.

[3] Anzengruber, B., Pianini, D., Nieminen, J., Ferscha, A., 2013. Predicting social density in mass events to prevent crowd disasters. In: Social Informatics - 5th International Conference, SocInfo 2013, Kyoto, Japan, November 25-27, 2013, Proceedings. pp. 206–215.

[4] Baliga, J., Ayre, R. W. A., Hinton, K., Tucker, R. S., jan 2011. Green cloud computing: Balancing energy in processing, storage, and transport. Proceedings of the IEEE 99 (1), 149–167.
URL `https://doi.org/10.1109/jproc.2010.2060451`

[5] Beal, J., Pianini, D., Viroli, M., 2015. Aggregate programming for the Internet of Things. IEEE Computer 48 (9).

[6] Bettini, L., 2016. Implementing Domain-Specific Languages with Xtext and Xtend. Birmingham, ISBN: 9781786464965, Packt Publishing Ltd., UK.
URL `/files/https://www.packtpub.com/application-development/implementing-domain-specific-languages-xtext-and-xtend-second-edition`

[7] Bonomi, F., Milito, R., Zhu, J., Addepalli, S., 2012. Fog computing and its role in the internet of things. In: Proceedings of the first edition of the MCC workshop on Mobile cloud computing. ACM, pp. 13–16.

[8] Bui, K.-H. N., Jung, J. J., 2019. Computational negotiation-based edge analytics for smart objects. Information Sciences 480, 222–236.

[9] Casadei, R., Fortino, G., Pianini, D., Russo, W., Savaglio, C., Viroli, M., feb 2019. Modelling and simulation of opportunistic IoT services with aggregate computing. Future Generation Computer Systems 91, 252–262.
URL `https://doi.org/10.1016/j.future.2018.09.005`

[10] Casadei, R., Pianini, D., Viroli, M., 2016. Simulating large-scale aggregate MASs with Alchemist and Scala. In: Computer Science and Information Systems (FedCSIS), 2016 Federated Conference on. IEEE, pp. 1495–1504.

[11] Casadei, R., Viroli, M., 2018. Collective abstractions and platforms for large-scale self-adaptive iot. In: 2018 IEEE 3rd International Workshops on Foundations and Applications of Self* Systems (FAS* W). IEEE, pp. 106–111.

[12] Casadei, R., Viroli, M., 2018. Programming actor-based collective adaptive systems. In: Programming with Actors. Springer, pp. 94–122.

[13] Cicirelli, F., Guerrieri, A., Spezzano, G., Vinci, A., 2017. An edge-based platform for dynamic smart city applications. Future Generation Computer Systems 76, 106–118.

[14] El-Sayed, H., Sankar, S., Prasad, M., Puthal, D., Gupta, A., Mohanty, M., Lin, C.-T., 2018. Edge of things: the big picture on the integration of edge,

iot and the cloud in a distributed computing environment. IEEE Access 6, 1706–1717.

[15] Fortino, G., Russo, W., Savaglio, C., Shen, W., Zhou, M., 2017. Agent-oriented cooperative smart objects: From IoT system design to implementation. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 1–18.

[16] Fortino, G., Russo, W., Savaglio, C., Viroli, M., Zhou, M., Jun. 2017. Modeling opportunistic IoT services in open IoT ecosystems. In: De Meo, P., Postorino, M. N., Rosaci, D., Sarné, G. M. (Eds.), WOA 2017 – 18th Workshop "From Objects to Agents". Vol. 1867 of CEUR Workshop Proceedings. Sun SITE Central Europe, RWTH Aachen University, pp. 90–95.

[17] Fortino, G., Russo, W., Savaglio, C., Viroli, M., Zhou, M., Feb 2018. Opportunistic cyberphysical services: A novel paradigm for the future internet of things. In: 2018 IEEE 4th World Forum on Internet of Things (WF-IoT). pp. 488–492.

[18] Fortino, G., Savaglio, C., Zhou, M., Aug 2017. Toward opportunistic services for the industrial internet of things. In: 2017 13th IEEE Conference on Automation Science and Engineering (CASE). pp. 825–830.

[19] Ghahramani, M. H., Zhou, M., Hon, C. T., 2017. Toward cloud computing qos architecture: Analysis of cloud systems and cloud services. IEEE/CAA Journal of Automatica Sinica 4 (1), 6–18.

[20] Hamedani, G. G., Tata, M. N., nov 1975. On the determination of the bivariate normal distribution from distributions of linear combinations of the variables. The American Mathematical Monthly 82 (9), 913.
URL `https://doi.org/10.2307/2318494`

[21] Hoyer, S., Hamman, J. J., apr 2017. xarray: N-d labeled arrays and datasets in python. Journal of Open Research Software 5.

[22] Hu, L., Miao, Y., Wu, G., Hassan, M. M., Humar, I., 2019. irobot-factory: An intelligent robot factory based on cognitive manufacturing and edge computing. Future Generation Computer Systems 90, 569–577.

[23] Hunter, J. D., 2007. Matplotlib: A 2d graphics environment. Computing in Science & Engineering 9 (3), 90–95.

[24] Minoli, D., Sohraby, K., Occhiogrosso, B., 2017. Iot considerations, requirements, and architectures for smart buildingsenergy optimization and next-generation building management systems. IEEE Internet of Things Journal 4 (1), 269–283.

[25] Odersky, M., Rompf, T., 2014. Unifying functional and object-oriented programming with scala. Communications of the ACM 57 (4), 76–86.

[26] Pedrycz, W., 2018. Granular computing for data analytics: a manifesto of human-centric computing. IEEE/CAA Journal of Automatica Sinica 5 (6), 1025–1034.

[27] Pianini, D., Montagna, S., Viroli, M., 2013. Chemical-oriented simulation of computational systems with Alchemist. Journal of Simulation.

[28] Pianini, D., Viroli, M., Beal, J., 2015. Protelis: Practical aggregate programming. In: Proceedings of ACM SAC 2015. ACM, Salamanca, Spain, pp. 1846–1853.

[29] Ray, S., Lindsay, B. G., oct 2005. The topography of multivariate normal mixtures. The Annals of Statistics 33 (5), 2042–2065.
URL `https://doi.org/10.1214/009053605000000417`

[30] Savaglio, C., Fortino, G., Zhou, M., 2016. Towards interoperable, cognitive and autonomic IoT systems: an agent-based approach. In: Internet of Things (WF-IoT), 2016 IEEE 3rd World Forum on. IEEE, pp. 58–63.

[31] Shafiq, M. Z., Ji, L., Liu, A. X., Pang, J., Venkataraman, S., Wang, J., jun 2016. Characterizing and optimizing cellular network performance during

crowded events. IEEE/ACM Transactions on Networking 24 (3), 1308–1321.
URL https://doi.org/10.1109/tnet.2016.2533612

[32] Sheng, Z., Pfersich, S., Eldridge, A., Zhou, J., Tian, D., Leung, V. C., 2019. Wireless acoustic sensor networks and edge computing for rapid acoustic monitoring. IEEE/CAA Journal of Automatica Sinica 6 (1), 64–74.

[33] Shi, W., Cao, J., Zhang, Q., Li, Y., Xu, L., 2016. Edge computing: Vision and challenges. IEEE Internet of Things Journal 3 (5), 637–646.

[34] Sonmez, C., Ozgovde, A., Ersoy, C., may 2017. EdgeCloudSim: An environment for performance evaluation of edge computing systems. In: 2017 Second International Conference on Fog and Mobile Edge Computing (FMEC). IEEE.
URL https://doi.org/10.1109/fmec.2017.7946405

[35] van der Walt, S., Colbert, S. C., Varoquaux, G., mar 2011. The NumPy array: A structure for efficient numerical computation. Computing in Science & Engineering 13 (2), 22–30.

[36] Viroli, M., Beal, J., Damiani, F., Audrito, G., Casadei, R., Pianini, D., 2018. From field-based coordination to aggregate computing. In: Lecture Notes in Computer Science. Springer International Publishing, pp. 252–279.
URL https://doi.org/10.1007/978-3-319-92408-3_12

[37] Viroli, M., Casadei, R., Pianini, D., 2016. On execution platforms for large-scale aggregate computing. In: Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct. ACM, pp. 1321–1326.

[38] Voelter, M., 2013. DSL Engineering: Designing, Implementing and Using Domain-specific Languages. CreateSpace Independent Publishing Plat-

form.
URL https://books.google.it/books?id=J2i0lwEACAAJ

[39] Yang, Y., Zhong, M., Yao, H., Yu, F., Fu, X., Postolache, O., 2018. Internet of things for smart ports: Technologies and challenges. IEEE Instrumentation & Measurement Magazine 21 (1), 34–43.