Service Modeling for Opportunistic Edge Computing Systems with Feature Engineering

Teemu Leppänen[a,*], Claudio Savaglio[b], Giancarlo Fortino[b]

[a]*Center for Ubiquitous Computing, University of Oulu, Finland*
[b]*Department of Informatics, Modeling, Electronics and Systems, University of Calabria, Italy*

Abstract

The opportunistic context in which edge computing systems operate poses different challenges, particularly in the light of user mobility, such as resource management and system orchestration, real-time responsiveness and performance quality requirements. Starting from this consideration, we propose a novel development process for modeling opportunistic edge computing services, which relies on *(i)* the ETSI MEC reference architecture and Opportunistic IoT modeling for the early stage of system analysis and design, i.e., domain model and service metamodel; and on *(ii)* feature engineering for selecting those important, opportunistic aspects, leading to distributed, effective and efficient MEC services. We exemplify the proposed development process by presenting a microservice-based mobility management service for online analysis in MEC system. To analyze the properties of user mobility, i.e. resulting handovers between access points, we automatically construct Opportunistic Feature Vectors for Edge, with a subset of the opportunistic features preliminary identified, in a real-world data set with domain expertise, at the analysis phase. Further applications of data analysis and machine learning techniques on these vectors are straightforward and allow tackling the opportunism (and, hence, the unpredictability and complexity) featuring the development of such a use case service and, in general, of edge computing systems.

Keywords: Multi-access Edge Computing, Opportunistic Computing, Service Modeling, User Mobility, Feature Engineering

1. Introduction

5G networks, as an enabler for future wireless communication infrastructures, are being commercially deployed across the world. For the Internet of Things (IoT) systems, 5G technologies are able to provide network features

*Corresponding author: teemu.leppanen@oulu.fi
P.O.Box 4500, FI-90014 University of Oulu, Finland

such as low latency, high bandwidth and reliability. However, due to the physical distances between the data collection at the network edges and computation at the cloud, relying on the massive-scale upstream IoT data transfer, cloud-based centralized IoT systems are not able to fulfill the requirements of real-time computation- or data-intensive applications. Edge computing [1] has emerged as a next generation distributed computing solution to address these issues, by leveraging cloud resources to the network edges, i.e. the infrastructure components, in close proximity of the user equipment (UE, such as smartphones, notebooks or other networking-enabled end user device). Here, European Telecommunications Standards Institute (ETSI) is leading a standardization efforts towards a reference edge computing system architecture, called Multi-access Edge Computing (MEC) [2, 3]. As exemplified by MEC, cellular network base stations are reused as content processing, sharing and storage platforms for virtualized applications, while real-time radio network information is provided for the optimization of the platform resource usage.

Besides the potential, 5G and edge computing technologies pose also significant challenges related to the complexities of massive-scale distributed computing atop 5G connectivity. Opportunistic elements resulting from the user mobility and operational environment, dynamic data traffic and limited resources for applications, all in real-time in close proximity of UEs in massive-scale, call for artificial intelligence and machine learning (ML) solutions for service provisioning, and orchestration and management of the edge resources [3, 4, 5]. Considered the overall complexity of the scenario, the modeling activity is key to specify and to visualize both the structure and behavior of edge computing systems (as they are or as we want they to be). Indeed, models can represent guidelines for supporting different stakeholders in the whole development process with a tunable high level of abstraction and detail.

On these premises, in this paper we provide a development process, based on both graphical representations (i.e., metamodels) and statistics method (i.e., feature vectors), for modeling edge computing services by simultaneously addressing the opportunistic nature of edge computing systems. In particular, we apply the Opportunistic IoT Service modeling [6] approach to the ETSI MEC reference architecture for providing a novel service profile and model for MEC. The exploitation of feature engineering helps in tackling the opportunism (and, hence, the unpredictability and complexity) introduced by edge computing-related elements by identifying the more relevant factors impacting on the system (e.g., system properties or configuration parameters). To the best of our knowledge, such approach for edge computing system development has not been proposed so far. To validate it, we model an example MEC service, i.e., Mobility Analysis service, in which a set of opportunistic service features about user movements (identified during the modeling phase with the help of a domain expert and organized into an Opportunistic Feature Vectors for Edge a.k.a. OFVE) is further automatically refined by means of Pearson correlation. Finally, we show how the gained knowledge is beneficial to ease the design and implementation of such an opportunistic mobility service and, in general, of edge computing systems.

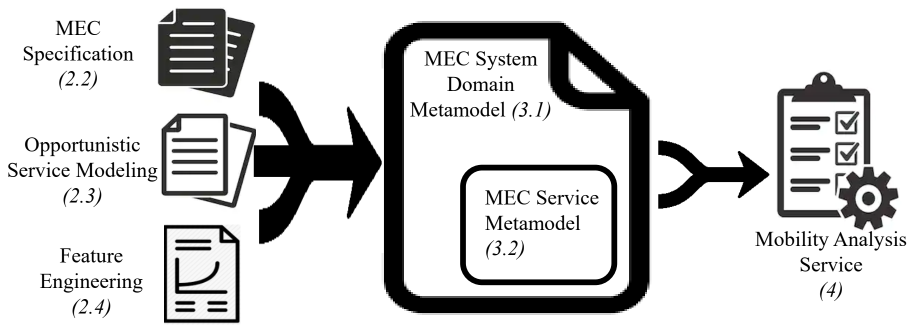
MEC Specification (2.2)
Opportunistic Service Modeling (2.3)
Feature Engineering (2.4)
MEC System Domain Metamodel (3.1)
MEC Service Metamodel (3.2)
Mobility Analysis Service (4)

Figure 1: Graphical outline of main contributions provided in the paper (related sections indicated in brackets).

The rest of the paper is organized as follows. In Section 2, we discuss and identify the several elements leading to opportunistic complexities in edge computing systems and how to address them at analysis (Opportunistic IoT metamodeling) and design (MEC specification and Feature Engineering) phases. In Section 3, we present our approach for MEC system modeling, from the full-fledged domain model to the specific service model, and provide the Mobility Analysis service as use case in Section 4. Finally, we draw the conclusions and outline our future work in Section 5. A graphical outline of main provided contents of this paper is reported in Figure 1.

2. Background

In this Section, we provide the main background elements for our proposal, namely an overview about the general challenges in edge computing systems and their opportunistic elements (2.1), the ETSI MEC reference architecture (2.2), the IoT system and service modeling (2.3), and finally the feature engineering process (2.4).

2.1. Edge Computing challenges and opportunistic elements

Edge computing [1] shifts the cloud-centric IoT computing resources towards a large-scale geographically distributed architecture. At the edge, servers and data centers are deployed in the proximity of mobile users, exploiting the network infrastructure such as cellular base stations and access points (AP). Primary edge computing architecture consists of powerful servers, co-located with the mobile network infrastructure, but additional edge resources can be deployed on existing location-based hardware, providing a further intermediate layers as in Fog computing [7]. Commonly, clustering algorithms, as in [8], are then utilized to optimized the edge server deployment with regard to placement of limited numbers of servers among the APs. Typically edge servers are considered powerful, thus capable of serving a number of APS, but as in Fog computing, each AP can be equipped with one or several small-scale computing platforms in location.

On these bases, edge computing paradigm promises significant improvements over the cloud-centric IoT systems, for users, 3rd party application developers, service providers and network operators. First, users are provided with dedicated resources to offload their data-intensive applications to a nearby edge platform. Second, users benefit from higher bandwidth and low latency in interacting with edge applications and services. Third, on the core network the data transmission load and latency are reduced, as data is already processed at one-hop distance at the edge platform. Fourth, system orchestration and resource management are distributed across the deployment architecture, resulting in a higher adaptivity, reactivity, context-awareness and responsiveness against the inherent dynamism IoT systems and networks. Therefore, the IoT applications that benefit most from the edge paradigm are *(i)* those with high computational and storage demands, *(ii)* those that generate a lot of network traffic, as well as *(iii)* applications requiring low latency and real-time interactions.

As for the benefits of edge computing, its opportunistic properties and their root causes are well acknowledged. These challenges at the edge are, broadly speaking, due to artefacts and elements related to the physical underlying networks, resource availability for large-scale distributed computing, application requirements and the side-products of system design and implementation. Hereafter, we report the key findings regarding the root causes for opportunistic properties in edge computing systems, as discussed in [3, 6, 8, 9, 10, 11, 12, 13, 14, 15, 16].

Users and devices. The justification for edge computing comes from the limitations of UEs, constantly demanding more processing power and data for meeting the Quality of Experience (QoE), for a wide variety of application types, such as real-time video streaming. Moreover, unpredictable user movements and interactions complicate maintaining network connectivity for UEs, which rely on different communication technologies.

Applications. Edge applications are either offloaded from a UE or deployed by pull request into the edge platform either as virtual machines or as containers. Virtual machines encapsulate the application as self-contained package, which may be large sized. This may result in poor flexibility for large-scale applications due to internal complexities in the software. Also, instantiation, deployment and relocation of such components is resource-consuming, limiting their scalability. On the other hand, a container encapsulates a microservice, which in turn implements a single function. A collective of microservices then implements the required application/service logic. A benefit is that scalability is addressed with an increased variety of deployment options, as for each microservice an extensive set of edge components is available. However, online composition and orchestration, inherent communication latencies and handling of possible faults of such distributed application is challenging.

Systems and platforms. To fulfill the edge computing promises notwithstanding the opportunistic elements, application-specific edge resources need to be constantly provisioned ”at the right place at the right time”, to guarantee sufficient QoE across the edge deployment, also during the peak times with consequent high data traffic. In such direction, the edge platforms play a key role: they are fundamental large-scale open distributed systems which act as a *(i)* middleware, for directing data traffic and interconnecting application-specific components, and *(ii)* application execution platforms, for providing system services in real-time to both the system components and the applications. Edge platform deployments are limited by operator budgets and deployed on top of existing core networks, which topology dictates to where the edge system components are placed. In particular, edge platforms handle both horizontal and vertical application-specific network traffic. The former is generated by the collaboration of the system components, with the network topology and link capacities dictating the maximum data transfer limits. The latter is generated by the applications that rely on all the system layers for application execution, requiring information exchanges between UEs, edge components and the cloud.

Orchestration and Management. The Orchestration and management of an edge platform in a centralized fashion introduces inevitable latencies in its operation, where the information about system performance is scattered across the system and rapidly updating. Therefore, aiming at optimized performance and QoE, edge platform components need to be reactive and adaptive in handling of the opportunistic properties. In response to user mobility, application components and services need to relocate in the system in runtime, which requires collaboration and decision-making of multiple system components. Here, automated decision-making, service composition, enforcing system policies, load balancing and conflict resolution needs to incorporate context-aware information, while placed at the cloud due to high demand on computational power for large-scale operational data, providing a view over the whole system in real-time. The address adaptivty and reactivity, MEC platform components are designed partially autonomous in their decision-making, but are expected to provide feedback on their operation to the system management. Here, automatic solutions offer limited help, such as machine learning (ML) algorithms that process large-scale data beyond human comprehension and provide for example predictions for the operators.

2.2. MEC System Reference Architecture

Hereafter, we briefly present the ETSI MEC reference architecture for edge computing systems, that is addressed in the next Sections in edge service modeling. Detailed information of the MEC reference architecture, system components, their functionality and interactions are presented in the ETSI documentation[1].

The aim of the MEC standardization is to provide an open multi-vendor edge computing platform with different stakeholders in mind, such as mobile network operators, vendors, service providers and 3rd party developers. The

[1]https://www.etsi.org/technologies/multi-access-edge-computing

standards give guidelines on how to realize MEC systems, atop the reference architecture, and how to implement the system services to ensure uniform operation across edge deployments and application domains. The standards specify a set of Application Programming Interfaces (API) that provide detailed interaction patterns and interoperability between the system components. The APIs address system management and authorization; application enablement, deployment and lifecycle; mobility management and monitoring the system performance. In addition, the standards define proof-of-concept applications and sets of functional and non-functional Key Performance Indicators (KPI) for evaluating the edge system performance.

ETSI MEC system architecture extends the role of network infrastructure components from forwarding network traffic to caching and sharing content and running services and applications. The MEC system reference architecture [17] is illustrated in Figure 2. MEC applications are initiated either by mobile user requests or by 3rd party developers and/or service providers that launch services on-demand, by creating applications context description(s) and specifying usage policies and billing information, etc., for the services. MEC system operations are divided into system level and host level. A set of system services support, at both levels, distributed management of the system resources and application enablement, deployment and lifecycle management in MEC platforms atop the edge Virtualization Infrastructure. The system services provide, and rely on, real-time information about the system state, resource usage and local network conditions. For MEC application development, ETSI models the MEC applications as a set of autonomous and loosely-coupled microservices [18].

At the system level, the centralized Orchestrator has visibility and authority over the system services and platform management. MEC application instantiation and execution is managed by the Orchestrator, based on the application contexts created from the user requests. The requests are validated and the contexts are adjusted, if needed, to comply with the resource availability in hosts, with regard to application load, available services and latencies. The virtualized application packages, and required services, are instantiated on the selected Hosts, based on the hardware requirements and network resources. Here, the Orchestrator has knowledge of the service attributes, configurations and dependencies, usage policies and billing. Based on a user request, or its own decision, the Orchestrator triggers relocation and termination of the applications. The application relocation process is described in detail in [19], executed by the Application Mobility Service.

At the Host management level, a MEC Platform manages a set of MEC Hosts. The Platform Manager controls the application lifecycle and resolves resource conflicts, with regard to data traffic and use of services, in the platform. The manager also collectes real-time information about its own performance. In parallel, the platform Virtualization Manager allocates the virtualization resources for the applications and services.

The MEC Hosts provide computational, networking and data storage resources for the hosted applications and services, atop its own virtualization infrastructure. Hosts support multi-access technologies to connect to the local

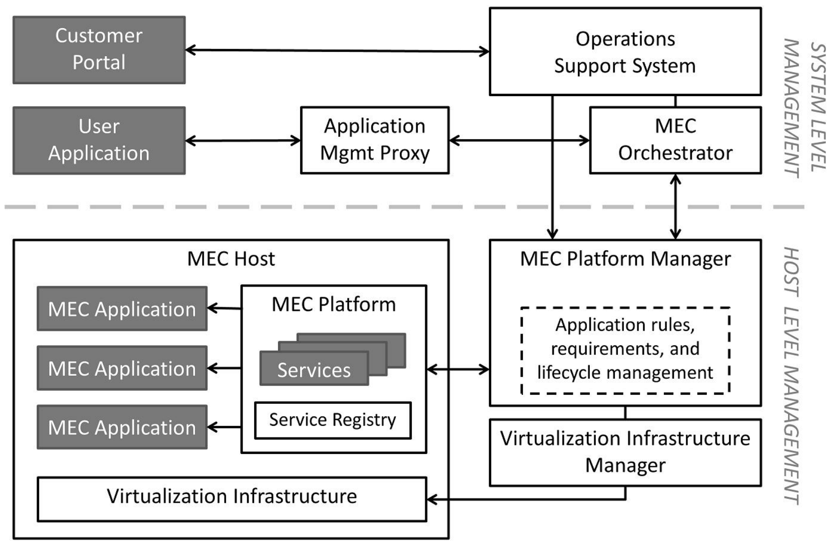

Figure 2: ETSI MEC Reference System Architecture

network and are capable of handling requests and data traffic of multiple UEs. Such a multi-tenancy requires sharing of platform resources, services and application instances. MEC host provides a service registry to discover, access, offer and advertise its services across the platform.

2.3. Modeling of IoT Systems and Services

As discussed in [14, 20, 21], the development of IoT applications imposes specific requirements, which are mostly orthogonal to traditional software engineering, including interoperability across heterogeneous system components, scalability of large-scale distributed deployments, security and privacy solutions, and software and hardware evolvability.

However, consolidated methodologies, spanning from analysis to design phase and including programming abstractions and software constructs for the implementation of such applications, are yet to emerge. In particular, the analysis activity helps in reducing the software development burden and time as well as increasing the software quality. By operational modeling the application requirements, system components, their functionality, intra- and interrelationships, organization and constraints, sets of software architecture models can be effectively represented. Simultaneously, non-functional requirements, such as quality attributes, system policies, maintenance guidelines and system evolution, can be expressed both informally and formally for further verification, validation and simulation before moving to the implementation phase.

Steps in this direction have already been taken, often adopting the principles of Model-Driven Engineering (MDE) for IoT system and application software development illustrated in [22, 23, 24, 25]. A first group of works focused on a conceptual, high-level modeling of IoT systems. For example, a unified modeling solution with different modeling approaches in different layers of IoT systems is presented in [26]. IoT-specific metamodels are presented in [27, 28, 29, 30], aiming to provide conceptual representations with standardized workflows and ontologies. Such works, although considering a holistic view over the IoT system elements, provide a modeling approaches focused on static environments with established interactions and layered system architectures, largely neglecting the opportunistic characteristics [14, 15]. A second group of works, still dealing with the modeling activity for IoT, focused on approaches specifically tailored for edge computing systems and services. For example, Cai et al. [31] focus on mobile application modeling in IoT dynamic environments, leading to service component definition to configure such services online. In [6] a full-fledged approach for the engineering from analysis to simulation of complex opportunistic and collective services at the IoT edge has been proposed, leveraging on the benefits of Aggregate Computing. Modeling and design of cognitive IoT systems has been addressed in [32, 33], with focus on reactive, proactive and cognitive behavior and online adaptation of the system, through refinement of the design models. Modeling of non-functional and QoS properties of IoT systems have been addressed in [34, 35, 36, 37].

The key findings of this short literature review on IoT systems and services modeling are the following. A model-driven engineering approach is particularly effective to support the whole development of complex systems such as the edge computing ones. The modeling activity, indeed, is widely acknowledged as a cornerstone for simplifying the further development steps. The technical knowledge and domain expertise, already at the analysis and design stages, increases awareness of development challenges and helps to produce better solutions in later stages of the implementation, deployment and maintenance stages. Exemplified in [8], domain knowledge about edge technologies, systems and mobile networks provides useful insights about expected performance, network topology and deployment options, reliability and privacy concerns, that are otherwise difficult to capture. Therefore, to systematically face the inherent opportunistic nature of the IoT environment, the most suitable way consists in a systematic and comprehensive approach relying on MDE and both technical and domain expertise.

2.4. Feature engineering for opportunistic modeling

Applied ML provides a set of powerful tools for analysing, learning and predicting about edge computing systems' properties, operations and performance. To realize the benefits of ML, the data analyst(s) shall bear the key and delicate process of tools setting for the target problem, by identifying the relevant variables and data to consider, the most fitting model with respect to the main system properties, and finally the correct learning algorithm to use. Then, model evaluation and tuning follows before the decision on how to present the results.

These preparatory steps, before the ML modeling, are commonly referred as feature engineering [38, 39], i.e. the "act of extracting features from raw data and transforming them into formats that are suitable for the ML model". Thus, the aim of feature engineering is to drop the difficulty in ML modeling, by preparing the input data to be meaningful for the identified problem and designing the right set of features that effectively represent the underlying problem. This way, the ML model capability of completing its task and/or its accuracy can be increased. In practise, feature engineering is difficult to generalize and is often described ambiguously as "bag of tricks" [38].

Nevertheless, in general, feature engineering process consist of the following steps. The first step is the problem definition, including describing the target(s) and finding the relevant data, which is typically collaboration between domain expert(s) and data analyst(s). The second step is the data preparation to preprocess and transform the data into format(s) that are easy for the mathematical ML tools to ingest, e.g. numeric and categorical variables presented in problem-specific feature vectors. The third step is the design of a feature set, i.e. feature extraction, aiming to properly represent the underlying problem and describe the inherent structures in the data. At this stage, automatic ML tools provide ways to extract large sets of simple statistical features from the data, but with the risk of obtaining high dimensional feature space. Another approach is to utilize domain expertise to extract and construct additional set of features that can for example incorporate contextual knowledge. Therefore, the fourth step is to utilize feature selection techniques, as presented in [39], for example, to reduce the dimensionality of the feature space. Generally, this step improves the data quality and results in a simpler model, easier dataset interpretation and maintenance, and increased algorithmic efficiency, traceability and effectiveness. However, feature selection is often experimental and iterative as well as highly context-aware, being the features meaningful just with respect to the data, the model and the task at hand [38].

In this paper, we utilize the outlined feature engineering process in Section 4, tailored for presenting and analyzing the opportunistic properties related to user mobility in MEC systems.

3. Opportunistic Service Modeling for MEC

Aiming to fill the gap between the high-level service modeling for opportunistic edge systems and the evaluation of those opportunistic aspects with data analysis, we propose a comprehensive IoT service modeling approach for MEC that follows the ETSI specifications. The service development process presented hereafter follows the MDE principles, by focusing on conceptual models to be refined in turn in the analysis (technology agnostic Opportunistic IoT Service Model), design (MEC specifications as the architecture for the edge system), and implementation (MEC specifications as a standard to realizing the

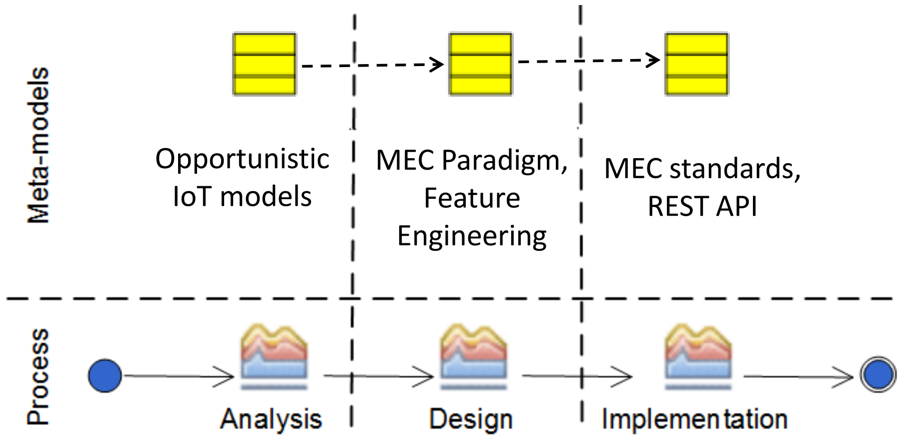

Figure 3: Phases of the proposed development process for edge computing services

service) phases. Such a process is depicted in Figure 3. In this section, we focus on the analysis and design parts of the development process.

3.1. Domain model for MEC

Following the Opportunistic IoT Service modeling [6], we identify the main elements of a generic IoT system to be modeled: *(i)* IoT Entities, that are classified as IoT devices, users and proxies, which interact with the environment by producing and consuming services, *(ii)* Services, that implement the functionality in the system by enabling atomic or composed configurations and applications, *(iii)* Context, that is needed for incorporating the domain- and system-specific information derived in the applications, *(iv)* Environment, namely the dynamic and complex physical IoT system environment which sets constraints on how the entities and services can operate, with available resources, in the resulting logical and physical application and service compositions, *(v)* Quality attributes, such as functional and non-functional QoS and QoE, including availability and flexibility, reliability, efficiency, maintainability, security and privacy, that compromise the system performance if omitted.

Such generic building elements have been mapped into the MEC reference architecture in order to define the high-level domain model for describing the main entities involved in MEC application or service creation, instantiation, provisioning, relocation and termination in the MEC systems (please note that, since the ETSI specifications do not differentiate between applications and 3rd party services, from the development and deployment perspectives they are both modeled through microservice as the basic building blocks for their functionalities). The resulting domain model for MEC systems according to the aforementioned Opportunistic IoT Service modeling categories (i.e., IoT Entity, IoT Environment and IoT Service) is illustrated in the Figure 4. Focusing on the MEC service perspective, we make the following observations:

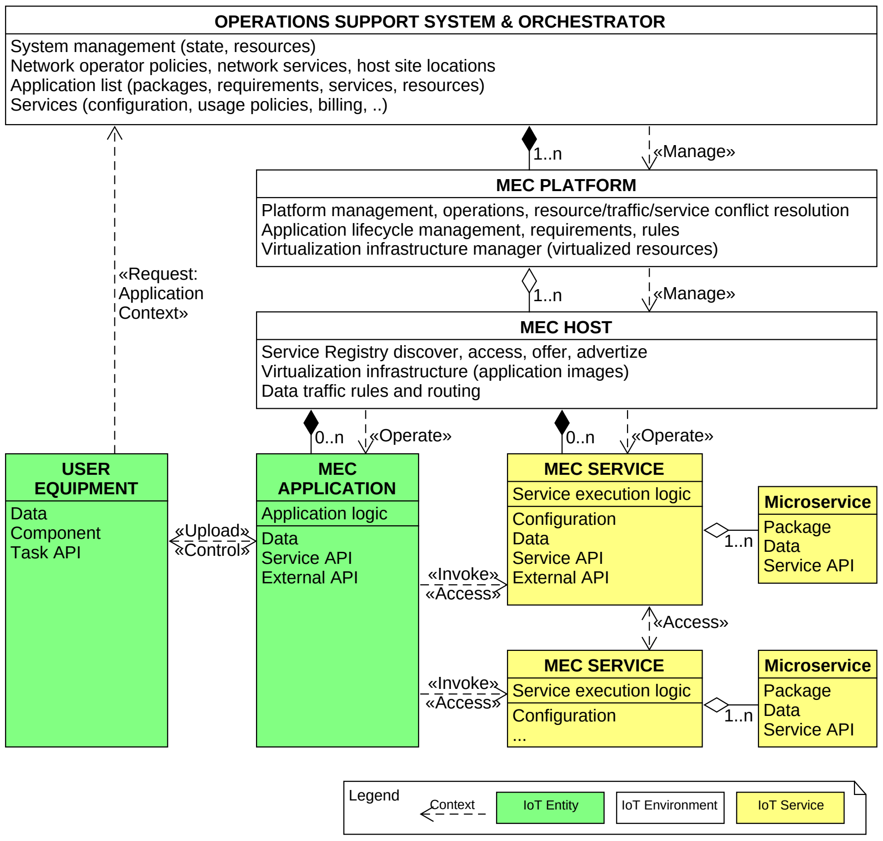
OPERATIONS SUPPORT SYSTEM & ORCHESTRATOR
System management (state, resources)
Network operator policies, network services, host site locations
Application list (packages, requirements, services, resources)
Services (configuration, usage policies, billing, ..)
1..n
«Manage»
MEC PLATFORM
Platform management, operations, resource/traffic/service conflict resolution
Application lifecycle management, requirements, rules
Virtualization infrastructure manager (virtualized resources)
«Request: Application Context»
1..n
«Manage»
MEC HOST
Service Registry discover, access, offer, advertize
Virtualization infrastructure (application images)
Data traffic rules and routing
0..n
«Operate»
0..n
«Operate»
USER EQUIPMENT
Data
Component
Task API
«Upload»
«Control»
MEC APPLICATION
Application logic
Data
Service API
External API
«Invoke»
«Access»
MEC SERVICE
Service execution logic
Configuration
Data
Service API
External API
1..n
Microservice
Package
Data
Service API
«Access»
«Invoke»
«Access»
MEC SERVICE
Service execution logic
Configuration
...
1..n
Microservice
Package
Data
Service API
Legend
Context
IoT Entity
IoT Environment
IoT Service

Figure 4: Domain model for MEC System

IoT Entities. We consider the MEC applications, other MEC system or 3rd party services and UEs to represent the IoT Entities that provide or consume the service in question. The UEs both produce data that the services consume, e.g. sensor data, but also the services may access or control the application components in the UEs. In turn, the service content is consumed by the applications or other services, to provide feedback for orchestration as an example.

IoT Service. The service provided, where the opportunistic elements are reported first in the Service Profile, i.e., its high-level description, and then in the Service Model, including its inputs, outputs and collective functionality. We address these aspects in the next Section in detail. Since the MEC specifications do not differentiate between applications and 3rd party services, from the development and deployment perspectives they are both modeled through microservice as the basic building blocks for their functionalities.

IoT Environment. Firstly, the IoT Environment consists of the physical

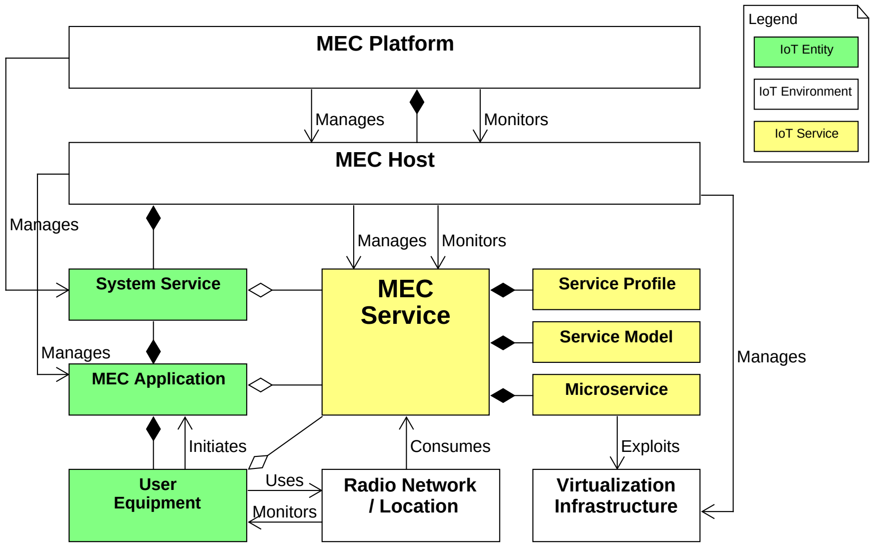

Figure 5: Opportunistic MEC Service Metamodel.

and virtual MEC system environment, including MEC Platforms and Host(s), where the Entities and MEC system elements are co-located. This includes MEC Platform and Hosts, where the system Orchestrator imposes system policies and environmental constrains into the service execution. But also the physical environment where the service operates, including local networks, is taken into account at the Platform level.

Context. Describes the dependencies between IoT Entities, Environment and Services in the model. MEC services are instantiated on request, either standalone or as a part of a MEC application, where the dependencies are described in its service configuration. This includes data from UEs and the utilized content from other services, which may system services or 3rd party services but also external, e.g. a weather service. Also the MEC system services utilize information provided by the service, or created from its operation on the system, for orchestration and management. However, additional dependencies are created through the utilization of services in forthcoming application and service contexts that are not yet known. Here the microservices paradigm is useful, as it facilitates straightforward reuse of existing service components. In such cases, these dynamic dependencies include service configurations and aggregates, their usage policies and billing information, which the Orchestrator resolves.

3.2. Metamodel for Opportunistic MEC Service

Based on Opportunistic IoT Service modeling [6, 14] procedure, the next step is to provide a metamodel for an opportunistic edge service, based on the MEC

Table 1: Mapping between IoT Service Profile and MEC Application and Service contextand configurations.

Opportunistic IoT Service Model	**MEC Specification**		
IoT Service Profile	*Application Context, Application List*	*Application Mobility*	*Service Info*
Service Name	Name, Address		Name
Service Description	Application Description, Provider		Description
Service Category	Continuity	Scope, State, Continuity	Category
Service Parameters	Latency, Bandwidth, Memory, Storage, Data Traffic Rules, Service Dependencies		
Service Outputs	Notifications, (Service) Interface, Transport Depencencies	Mobility Type	
Service & Context Preconditions	Application Packages, Host Features, Service Dependencies, Hardware Descriptor, Storage Descriptor, Transport Protocol	Mobility Type	State
Service & Context Effect	Interface, Notifications	Mobility Type	State
Service Provision Constraints	Continuity	Mobility Type	State

specifications. Such a metamodel captures the elements of both approaches (Opportunistic IoT Service Model and MEC specifications) for conceptual and functional mapping. The resulting mapping is depicted in Table 1. For clarity, the set of parameters is not exhaustive and identifiers have been omitted, to focus on the service instantiation, provisioning, interoperability and lifecycle management aspects.

- Service Name. Identifier for the service, available in Application Context or ServiceInfo, including the network Address parameter.
- Service Description. Human- and machine-readable description of the service, based on standardized ontologies, that is published as a part of the MEC system Descriptors, and on the users requests (Application Contexts) and service provisioning configurations. Includes also the descriptive Provider information, billing and usage policies.
- Service Category. Edge platform services are categorized into *(i)* system services, *(ii)* 3rd party services, and *(iii)* external services that are hosted in remote systems. System services provide the functionality needed in the core platform architecture, exemplified by Application Package Management and Radio Network Information (RNIS). The 3rd party services are location-, context- and application-specific services that are instantiated

on-demand. Additional complexity is added with external services, deployed outside the system control, but needed in the realization of service logic, e.g. weather data. Service Category is an entry of a service ontology/taxonomy, available in MEC in ServiceInfo. The service Scope defines the service as either dedicated (for particular UE) or shared service.

- Service Parameters. Functional and non-functional QoS and QoE parameters of the service, such as maximum allowed access latency and reliability, and their related KPIs. Includes common core execution capabilities derived from the MEC specifications, such as Data Traffic Rules and Operator Policies.

- Service Inputs. Information required for the service execution through the system and 3rd party service APIs, and external data sources, including remote systems and UEs. MEC standards provide Service configurations with Dependencies that link to the inputs as a part of collective or aggregate.

- Service Outputs. The content and output(s) produced by the service that is published through its API. In MEC system, includes Application Context, Exported Interface and Notification(s) to other system components. In addition, in system management, the ServiceInfo and Mobility parameters are needed in service relocation and performance monitoring, for example.

- Service & Context Preconditions. The functional conditions required for the service instantiation, execution and relocation, such as Application Packages and Service Dependencies. The MEC Orchestrator assigns the application execution to a MEC Platform that can fulfill the Service Parameters in its Hosts. In MEC specifications these parameters include the logical service collections and aggregations, as described in the configuration, and Required Feature(s), Hardware Requirements, Storage Resources, Transport Protocol and Data Traffic Rule(s).

- Service & Context Effect. The events resulting from service execution, which are published as the MEC Notifications. The collective or aggregate service logic can be built on these effects, advancing the execution state between microservices. Feedback of the service execution, based on the effects, is provided by the Host to the Platform and the system Orchestrator.

- Service Provision Constraints. Constraints arising from the relevant system components. From MEC specifications, includes Continuity, Colocation and Mobility aspects. These constraints are described in the Service configuration, that is validated by the Orchestrator and then imposed through the Platform Manager to the Host(s) and the Virtualization Infrastructure.

To follow the Opportunistic Service Modeling procedure, we create an operational service model for presenting the main elements of a MEC System, as described in the previous Section, but also for identifying its opportunistic elements. Such parameters are difficult to handle through a static metamodel as in [14], therefore an additional and complementary model is needed. The resulting metamodel, based on the domain model and Service Profile, is illustrated in Figure 5 and it defines the Context for the modeled Service. Starting from the IoT Entities, we identify the service content consumers, as System Services, Applications and UEs, where only the system services are static. As discussed, service relocation may be launched based on UE movement or system orchestration. Similarly, the IoT Environment consists of the MEC system components, where the Host can be considered static as during instantiation the requirements are guaranteed. However, the opportunistic physical environment is represented by the system services RNIS and MEC Location. The microservices as IoT Service components are instantiated on the service request a a part of the configuration, but their location varies according to the Orchestrator decision-making and is dynamic in response to system orchestration.

Moreover, the metamodel describes the interactions between these opportunistic elements, e.g. "initiates", "exploits", "uses", "manages" and "monitors", that depend on the Service configuration as Provision Constraints, Inputs and Outputs, Context Preconditions and Effects. Given such Service Profile and metamodel, the inherent opportunistic elements of the MEC system and of the modeled service can be identified, and addressed further in the further steps of the software development process.

4. Use Case: Mobility Analysis Service

In this Section, we instantiate the proposed development process for opportunistic MEC services on a simple yet effective Mobility Analysis Service (MECMMS), that was initially proposed in our earlier work in [40, 41, 42]. Indeed, a crucial component in mobile edge computing system is the mobility management system service(s) that collect data of user movement while connected to the system [3]. The aim is to optimize application relocation, for example, by detecting movement traces and patterns of users and making predictions of such events. Particularly challenging, with respect to edge computing, is a handover triggered by user movement between APs that are associated with different edge hosts, as such a handover requires application relocation, instantiation, etc., across the core network, administrated by the system orchestrator. In general such system service(s) in large-scale requires holistic view over the system state and cloud-scale computing capacity, where real-time responsiveness is challenging. Steps towards cognitive approaches to handle mobility in online, tailored for edge and depending on ML methods, and that also incorporate contextual information, have been suggested in [9, 43, 44].

We address the complexity and the opportunism of such a MECMMS by following the proposed development process. First, a service is modeled with the MEC Service metamodel and the Service Profile, resulting a Service Model.

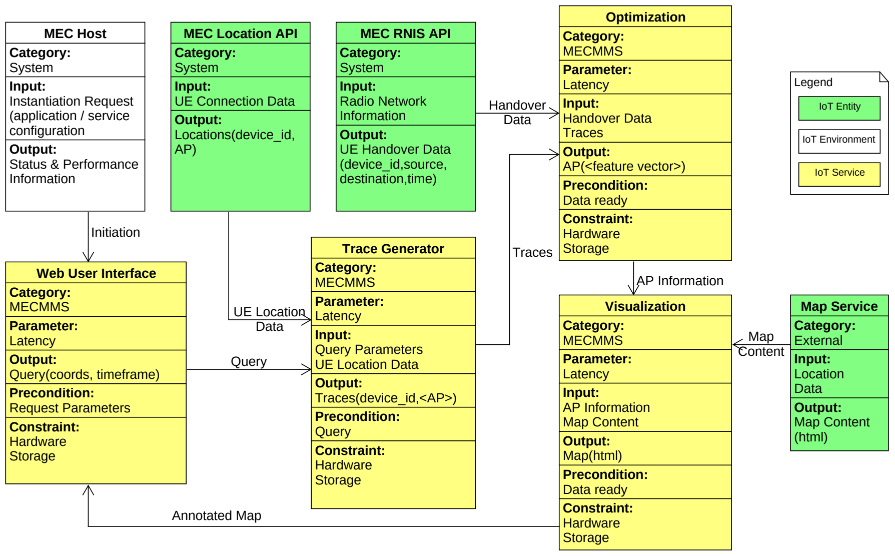
MEC Host
Category: System
Input: Instantiation Request (application / service configuration
Output: Status & Performance Information
MEC Location API
Category: System
Input: UE Connection Data
Output: Locations(device_id, AP)
MEC RNIS API
Category: System
Input: Radio Network Information
Output: UE Handover Data (device_id,source, destination,time)
Optimization
Category: MECMMS
Parameter: Latency
Input: Handover Data Traces
Output: AP(<feature vector>)
Precondition: Data ready
Constraint: Hardware Storage
Legend
IoT Entity
IoT Environment
IoT Service
Handover Data
Initiation
Traces
AP Information
Web User Interface
Category: MECMMS
Parameter: Latency
Output: Query(coords, timeframe)
Precondition: Request Parameters
Constraint: Hardware Storage
UE Location Data
Query
Trace Generator
Category: MECMMS
Parameter: Latency
Input: Query Parameters UE Location Data
Output: Traces(device_id,<AP>)
Precondition: Query
Constraint: Hardware Storage
Visualization
Category: MECMMS
Parameter: Latency
Input: AP Information Map Content
Output: Map(html)
Precondition: Data ready
Constraint: Hardware Storage
Map Content
Map Service
Category: External
Input: Location Data
Output: Map Content (html)
Annotated Map

Figure 6: Service Model for MEC Mobility Analysis Service

Then, an example service implementation is described that addresses a optimization of MEC system operation. Lastly, a QoS attribute in MEC is analyzed with a real-world data set, exemplifying how the service can address the opportunistic properties of MEC systems.

4.1. MEC Mobility Analysis Service

The Service Model of MECMMS presented hereafter is based on the MEC Service metamodel. In particular, we focus on the MEC Host level and omit the modeling of the system level orchestration and management components, as an extensive set of components and their interactions need be taken into account without adding to the contribution of this paper.

MECMMS is considered a system service on the MEC system, a part of the MEC Mobility Management set of services. When an relocation request is received by the Application Mobility Service [19], it collaborates with MECMMS to analyze and plan the relocation across the platform. An instance is launched in each MEC Platform (*IoT Environment*) upon instantiation. MECMMS is designed as a set of microservices, that rely on the outputs of other system services and external services, to implement the service functionality. The preconditions and constrains for MECMMS are that an instance is placed on each platform and that the microservice collective is deployed into the same platform for real-time content delivery in the close proximity of users. As described in [41], a set of MECMMS instances can be deployed as a hierarchy in the System, according to the capacity of Hosts, where additional preconditions are the microservice container size and adequate processing power. The service parameter, i.e. quality measure, is latency with regard to the content delivery, which is crucial for the system level orchestration of application relocation, in response to user mobility while maintaining QoE as in continuity. The inputs for MECMMS are UE location information and a geographical map. In the Context of MECMMS, the opportunistic property are the users (*IoT Entities*), that move across the system in runtime, represented by the two standardized MEC services (Location and RNIS API). The Output of MECMMS is a dataset containing user mobility analysis results and an annotated map, as described later in this Section. The dataset provides information on user mobility across the system for orchestration and management of application relocations.

The resulting MECMMS Service Model is illustrated in Figure 6. MECMMS operation is based on four microservices (*IoT Service*), namely Web User Interface (UI), Trace Generator (TG), Optimization function and Visualization. The service uses two standardized MEC System services, i.e. the Location API and the RNIS API, and an external Map Service. The front-end for MECMMS is UI, that provides parameters of the Region-of-Interest (ROI), i.e. geographical area and a timeframe, for the movement data analysis. The ROI is sent to the TG microservice, that retrieves UE location data, i.e. connections to APs within the ROI during the timeframe, through the MEC Location API. TG calculates the Traces of the UE movements across the MEC system, that is the UE Location as connections to an AP. The Traces, with UE handover data from RNIS, are then analysed further in the Optimization services that publishes AP Information data, that includes a set of mathematical features generated from the data. Lastly, an annotated map of the set of APs is created by the Visualization in response to the initial request by the Web UI.

With respect to the Service Profile, the MEC specifications address mobility management services and related application relocation in [45]. User movement, resulting a handover, or an on-demand system orchestration request, e.g. for performance optimization, load balancing and enforcing policies, initiates an application relocation request. As a Precondition, the optimal time window for the relocation and the target Host(s) needs to identified. Here, to address QoE concerns, a set of nearby Hosts can be assigned as a relocation group, on the premise that the application can be run in any of them. Another solution, is to introduce an optimization function as a part of the MEC Platform Manager functionality [45].

4.2. Analysis of the Edge Opportunistic Properties

In this subsection, we first present the output dataset of the MECMMS service. Then, with a real-world data set, an data analysis is performed in order to identity a feature set that describes the opportunistic properties of user movement in our data set, targeting optimization for MEC application relocation. In the analysis, we follow the principles of feature engineering introduced in Section 2.4 aiming to identify and exploit those opportunistic features that impact on the MECMMS service provision.

4.2.1. Setting up the problem

As a starting point for the problem definition, a domain expert has identified the application relocation, as the opportunistic property of MEC systems, to be addressed in data analysis by feature engineering. In the MEC specifications [45], Continuity is a performance metric for a MEC system, describing the requirement for the application to continue its operation during a UE session transition, e.g. a handover. Particularly, application relocation between MEC Hosts results in an interruption. Continuity is defined with different categories: No Continuity, Low Continuity, Soft Continuity and High Continuity. Each category has different time limits for tolerable interruptions.

The MECMMS service, as illustrated in Figure 6, provides a data set per AP that describes the above-mentioned set of features and data analysis results. However, since the MEC system nor the MECMSS Service real-world implementations are available, to replace the dsata sources, i.e. the MEC system APIs, we utilize an existing real-world data set in [46]. This data set contains UE connection data in one second accuracy, collected between the years 2007-15, in 1300 WiFi APs deployed in an open WiFi network across the City of Oulu, Finland. The size of the whole data set is about 275M data points. However, for demonstrative purposes, we utilize a subset of the data containing about half a million data points collected in 410 WiFi APs during February 2015.

As the result of data analysis, a set of APs are identified being considered the most opportunistic ones across the deployment, regarding the different attributes of Continuity.

4.2.2. Data preparation

Based on our previous work [8, 40], the data set has already been prepared: *(i)* removal of the handovers between wireless network APs, which last less than 5 seconds (the last handover was removed as an outlier) to mitigate the effect of rapid user movements, *(ii)* detection of a set of APs with very strong transmitters, to which almost all UEs in an area were connected at some point of time, even from a long distance (these handovers where removed from the data set as outlier), *(iii)* removal of those APs with less than 100 connections per day, aiming to focus on high workload APs and to provide sufficient samples for further analysis. As a result of this data preparation process, the initial set of 410 APs has been reduced to 56 APs with 502600 data points in total.

Aiming to study Continuity, we preliminary prepare the data, which originally only provides the following data attributes for a handover event: Timestamp, Device ID, Origin AP, Destination AP and Distance between APs. The Device ID is dropped, since we are interested about the handovers without considering individual UE movement patterns. The variables in the data set are defined as numeric variable (Distance), categorical variables (Origin and Destination APs) and date and time (Timestamp).

Since the timestamp only reports when a user connected to an particular AP, the time for the session and user movement can't be directly estimated. Typically, wireless network connections are intermittent, where the actual user disconnection time from the origin AP is not given in the data. Likewise, the data does not show how fast the user moved between the APs, i.e. by rapidly in a vehicle or slowly by walking. Therefore, we utilize the geospatial distance between APs, i.e. GPS coordinates that are available in a separate file, as an approximation for the maximum allowed relocation delay. In the literature, different distance measures, such as geospatial coordinates or hop counts, have been commonly utilized to estimate communication latencies. However, in our data set, the underlying network topology is not available. Moreover, typically connections from longer distances are considered to suffer from low QoE, proportional to the distance.

As a result of the above data preparation step, we decided to focus on *geospatial distance* as the attribute representing different aspects of user mobility to study Continuity. Assuming user mobility patterns can be predicted between APs, we can consider distance of an expected handover to be proportional to the time the MEC system has for selecting the relocation group and completing the application relocation process, while maintaining the QoE. Moreover, when the distance matrix of APs in a deployment can be constructed, variety of mathematical methods are available to further analyze the data in ML applications.

Finally, in this step, since we are interested about the opportunistic properties of handovers, the data divided into a table per each AP, containing all the handover events, where the AP in question was the Destination AP. Such a data transformation enables an easy way to calculate common statisctical measures of handovers for each AP.

4.2.3. Design of Opportunistic Feature Set

Next, we design the feature set to be used in mathematical modeling of Continuity.

Since the distance data in our dataset is numeric, we can straightforwardly construct feature vectors that represent the different measures and attributes of the distance data for each AP. We call those vectors as Opportunistic Feature Vectors for Edge (OFVE), to highlight that the vectors describe the salient and opportunistic - dynamic, diverse, sporadic and scattered - aspects of the edge computing. Having described such a initial feature vector, the relevant set of features to the problem at hand can be selected.

With respect to the considered use case, based on common statistics, the following feature set is initially considered representing the opportunistic properties of the handover distance(s):

MAX The longest distance of a handover to an AP; the longer the distance, the more users can connect to this AP and the worse the QoE becomes, leading to opportunistic connectivity.

MEAN The arithmetic mean of the handover distances to an AP, reflecting the tendency towards longer handover distances.

MODE The distance that appears most of them for an AP, reflecting the tendency towards longer handover distances.

Table 2: Correlation matrix of the selected features.

	Max	**Mean**	**Skew**	**Unique**
Max	1.00			
Mean	0.33	1.00		
Skew	0.09	-0.52	1.00	
Unique	0.40	0.45	-0.34	1.00

RANGE The range of handover distances to an AP, reflecting the difference in handover distances.

SD The standard deviation of the handover distances to an AP; large SD means large variation in distances, leading to opportunistic connectivity.

SKEW The measure of asymmetry (extremes) in the handover distance distribution of an AP; significantly skewed distribution reflects towards abnormal handover distances.

Whereas the above features are the standard statistical measures, in addition, a domain expert can propose a set of "handcrafted" features, that can be computationally more complex, but are assumed to be more expressive. In the use case, we consider the following additional features:

CV The coefficient of variation for comparison between APs, useful since the extend of variability between MEAN and SD is observed large in the data set.

UNIQ The number of unique source APs from where handovers to an AP were observed; the more individual UEs connect to the AP, the more diverse and dynamic its workload becomes.

ENT The entropy as a measure of uncertainty (and diversity) of the handover distances in an AP.

4.2.4. Feature selection

To reduce the dimensionality of an OFVE feature space (potentially very wide, with many features based on the MEC standards and domain expertise, for example), a plethora of feature selection techniques are available, as surveyed for example in [39].

For the sake of simplicity, we utilize Pearson correlation to reduce the feature space resulting from feature extraction described in the previous Section 4.2.3. We set the correlation coefficient to 0.5 for the OFVEs and remove the features that have higher correlation, while keeping the features with negative correlation to gain diversity for mathematical modeling.

Therefore, the dimension of distance OFVE, with initial nine features, is reduced to a set of four features that are MAX, MEAN, SKEW and UNIQUE, as illustrated in Table 2. The handcrafted feature UNIQUE is thus automatically determined to be more expressive of distance variation than some other standard statistical measures, demonstrating the need of domain expertise in modeling.

Table 3: Top opportunistic APs ranked with each feature.

Max		**Mean**		**Skew**		**Unique**	
ID	**Value**	**ID**	**Value**	**ID**	**Value**	**ID**	**Value**
823	111.75	**834**	12.40	**581**	26.74	**848**	202
197	108.02	**930**	11.88	**762**	9.44	**920**	185
936	104.89	**931**	9.61	**767**	8.28	**823**	175

4.2.5. Analysis

For visual analysis and interpretation by a domain expert, the APs identified in the Table 3, and in more detail in Table 4, are reported on a map in Figure 7. From the Figure, it clearly emerges that the opportunistic APs are placed along the major routes from north, west and south to the centre of the city of Oulu. As an example, the AP 930, placed next to highway crossings, has high mean distance of handovers. APs 823, 848 and 920, on the southern and northern end of the city centre, have the unique source APs. Interestingly, the AP 834, next to a long range bus station, has significantly high mean handover distances with low skewness, possibly explained by the patterns of connections of arriving passengers. Three exceptions to the initial assumption are shown: the two APs right at the city commercial centre (762,767) and one AP next (581) to a sports arena. The low maximum of handover distances in AP 762 can be explained by its very central location. The AP 581 has the lowest maximum and mean of distances and lowest number of unique source APs, with the highest skewness, possibly confirming that users arrive to the sports arena only from certain directions and slowly, e.g. by walking.

With respect to Continuity in the presented network topology, the applications in the categories No Continuity and Low Continuity (tolerable to interruptions of several minutes [45]), there is no observable downsides, except provisioning enough resources during sporadic events in populated areas, such as the sports arena (AP 581). Deploying Soft (tolerable to short interruptions) and High Continuity (with strict latency limits) applications, low handover distances escalated by the movement speed are challenging, due to small time window to orchestrate and deploy application components across the Host(s), as in AP 581 and central AP 762. Therefore, defining relocation groups, in the proximity of those APs, is an important architectural deployment decision. When the number of unique source APs is high (e.g. APs 823,848 and 920), and possibly distances low, a large relocation group or with careful design of server deployment [8], costly inter-host handovers could be avoided. With high mean (and standard deviation) and skewness (APs 581, 834 and 930), orchestration becomes problematic, clearly requiring system services for automatic planning, scheduling and load balancing.

Table 4: Top opportunistic APs.

ID	Max	Mean	Skew	Unique
197	108.02	4.04	7.09	72
581	65.53	0.67	26.73	36
762	73.26	3.74	9.43	144
767	99.89	4.74	8.28	127
823	111.75	6.58	4.90	175
834	88.13	12.40	0.89	153
848	92.63	6.83	4.91	202
920	90.84	6.97	3.44	185
930	102.56	11.88	2.93	156
931	102.56	9.61	3.32	124
936	104.88	6.77	4.02	157

5. Conclusion

Multiple heterogeneous factors from different sources introduce opportunistic behaviors in edge computing systems (large scale deployments, user mobility, ephemeral interactions, etc). Whereas the knowledge of the dynamics of the operational environment are often based on domain expertise, more systematic methods can actually help in addressing these issues throughout the whole edge application and/or service lifecycle. Such methods result particularly effective, if applied from the beginning with service modeling, including initial analysis and design steps aiming to identify and later address the opportunistic elements and their properties. Service modeling is an activity already investigated in the IoT context, but often overlooking the distinct requirements, properties and features of edge computing systems.

In this paper, we presented a novel service modeling approach, applied over the ETSI MEC reference architecture, that particularly addresses the opportunistic properties in edge computing. Moreover, we have formalized the identification and management of the opportunistic properties, based on domain expertise, through feature engineering methods for capturing and selecting sets of meaningful aspects for further analysis and ML modeling. To exemplify our proposal, we modeled a microservice-based User Mobility Analysis service for MEC and built opportunistic vecture fectors for mathematical analysis of user mobility activities in a real-world data set. Such refined information enables obtaining insights about user behavior in large scale, mobile, edge computing scenario to fulfill the edge promise of application resources ”at the right place at the right time”. Lastly, as illustrated by the modeling and analysis results based on the proposed approach, a set of edge infrastructure components were identified, where opportunistic properties can be observed and addressed by the network operator.

Our future work will focus on further developments for edge modeling methodologies and studying feature engineering methods in real-world use cases, atop

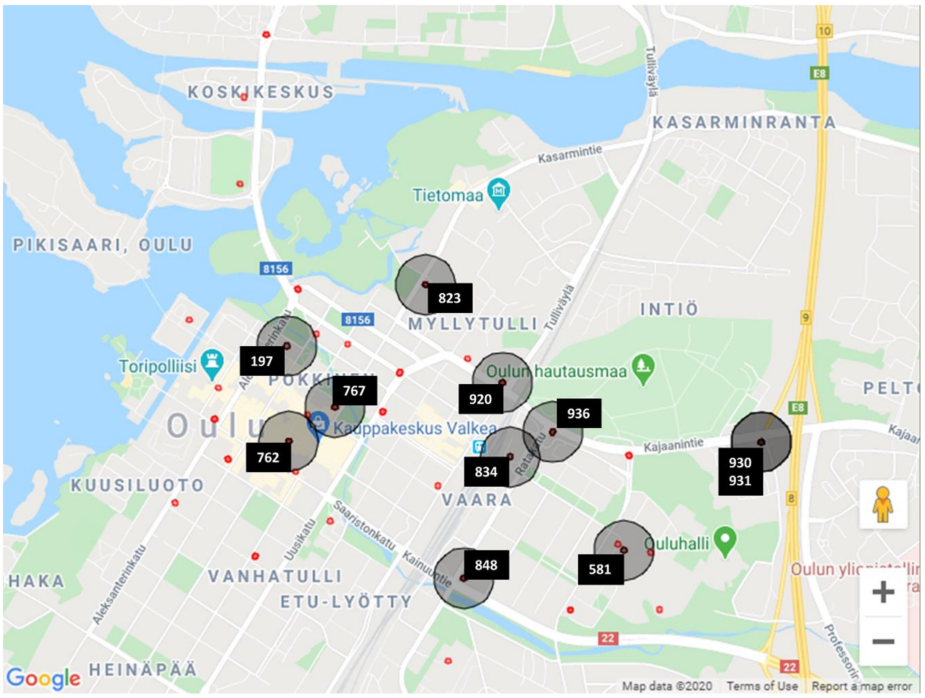

Figure 7: The identified set of opportunistic APs (red dots depict the other APs).

a 5G testbed [47] as a part of edge computing infrastructure.

Acknowledgment

The authors would like to thank Mr. Tommi Järvenpää and Mr. Lauri Lovén for providing the initial user mobility data set used to built MECMMS service prototype and the illustrations. This work is supported by the Academy of Finland 6Genesis Flagship (grant 318927) research program and the Future Makers program of the Jane and Aatos Erkko Foundation and the Technology Industries of Finland Centennial Foundation.

References

[1] W. Shi, J. Cao, Q. Zhang, Y. Li, L. Xu, Edge computing: Vision and challenges, IEEE internet of things journal 3 (5) (2016) 637–646.

[2] ETSI, Mobile edge computing a key technology towards 5g, White Paper 11.

[3] Y. Mao, C. You, J. Zhang, K. Huang, K. B. Letaief, Mobile edge computing: Survey and research outlook, arXiv preprint arXiv:1701.01090.

[4] R. Li, Z. Zhao, X. Zhou, G. Ding, Y. Chen, Z. Wang, H. Zhang, Intelligent 5g: When cellular networks meet artificial intelligence, IEEE Wireless communications 24 (5) (2017) 175–183.

[5] C. Jiang, H. Zhang, Y. Ren, Z. Han, K.-C. Chen, L. Hanzo, Machine learning paradigms for next-generation wireless networks, IEEE Wireless Communications 24 (2) (2016) 98–105.

[6] R. Casadei, G. Fortino, D. Pianini, W. Russo, C. Savaglio, M. Viroli, A development approach for collective opportunistic edge-of-things services, Information Sciences 498 (2019) 154–169.

[7] 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, ACM, 2012, pp. 13–16.

[8] T. Lähderanta, T. Leppänen, L. Ruha, L. Lovén, E. Harjula, M. Ylianttila, J. Riekki, M. J. Sillanpää, Edge server placement with capacitated location allocation, arXiv preprint arXiv:1907.07349.

[9] M. Chen, W. Li, G. Fortino, Y. Hao, L. Hu, I. Humar, A dynamic service migration mechanism in edge cognitive computing, ACM Transactions on Internet Technology (TOIT) 19 (2) (2019) 1–15.

[10] M. Chen, Y. Hao, C.-F. Lai, D. Wu, Y. Li, K. Hwang, Opportunistic task scheduling over co-located clouds in mobile environment, IEEE Transactions on Services Computing 11 (3) (2016) 549–561.

[11] W. Li, X. You, Y. Jiang, J. Yang, L. Hu, Opportunistic computing offloading in edge clouds, Journal of Parallel and Distributed Computing 123 (2019) 69–76.

[12] T. Leppänen, J. Riekki, Energy efficient opportunistic edge computing for the internet of things, in: Web Intelligence, Vol. 17, IOS Press, 2019, pp. 209–227.

[13] R. Olaniyan, O. Fadahunsi, M. Maheswaran, M. F. Zhani, Opportunistic edge computing: concepts, opportunities and research challenges, Future Generation Computer Systems 89 (2018) 633–645.

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

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

[16] P. Mach, Z. Becvar, Mobile edge computing: A survey on architecture and computation offloading, IEEE Communications Surveys & Tutorials 19 (3) (2017) 1628–1656.

[17] ETSI, Multi-access Edge Computing (MEC); Framework and Reference Architecture, ETSI GS MEC 003.

[18] A. Reznik, R. Arora, M. Cannon, L. Cominardi, W. Featherstone, R. Frazao, F. Giust, S. Kekki, A. Li, D. Sabella, et al., Developing software for multi-access edge computing, ETSI, Sophia Antipolis, France, White Paper 20.

[19] ETSI, Multi-access Edge Computing (MEC); Application Mobility Service API, ETSI GS MEC 021.

[20] P. Asghari, A. M. Rahmani, H. H. S. Javadi, Service composition approaches in iot: A systematic review, Journal of Network and Computer Applications 120 (2018) 61–77.

[21] T. Usländer, T. Batz, Agile service engineering in the industrial internet of things, Future Internet 10 (10) (2018) 100.

[22] B. Morin, N. Harrand, F. Fleurey, Model-based software engineering to tame the iot jungle, IEEE Software 34 (1) (2017) 30–36.

[23] K. Jahed, J. Dingel, Enabling model-driven software development tools for the internet of things, in: IEEE/ACM 11th International Workshop on Modelling in Software Engineering, 2019, pp. 93–99.

[24] C. M. Sosa-Reyna, E. Tello-Leal, D. Lara-Alabazares, An approach based on model-driven development for iot applications, in: IEEE International Congress on Internet of Things, 2018, pp. 134–139.

[25] F. Ciccozzi, I. Crnkovic, D. Di Ruscio, I. Malavolta, P. Pelliccione, R. Spalazzese, Model-driven engineering for mission-critical iot systems, IEEE software 34 (1) (2017) 46–53.

[26] K. M. Abbasi, T. A. Khan, I. U. Haq, Hierarchical modeling of complex internet of things systems using conceptual modeling approaches, IEEE Access 7 (2019) 102772–102791.

[27] S. Meissner, D. Dobre, M. Thoma, G. Martin, Internet of things architecture iot-a project deliverable d2. 1–resource description specification, URL http://www. meet-iot. eu/deliverables-IOTA/D2_1. pdf [last accessed: Jan 2017].

[28] M. Thoma, S. Meyer, K. Sperner, S. Meissner, T. Braun, On iot-services: Survey, classification and enterprise integration, in: 2012 IEEE International Conference on Green Computing and Communications, IEEE, 2012, pp. 257–260.

[29] S. De, P. Barnaghi, M. Bauer, S. Meissner, Service modelling for the internet of things, in: 2011 Federated Conference on Computer Science and Information Systems (FedCSIS), IEEE, 2011, pp. 949–955.

[30] W. Tan, Y. Fan, M. Zhou, Z. Tian, Data-driven service composition in enterprise soa solutions: A petri net approach, IEEE Transactions on Automation Science and Engineering 7 (3) (2009) 686–694.

[31] H. Cai, Y. Gu, A. V. Vasilakos, B. Xu, J. Zhou, Model-driven development patterns for mobile services in cloud of things, IEEE Transactions on Cloud Computing 6 (3) (2016) 771–784.

[32] F. Cicirelli, A. Guerrieri, A. Mercuri, G. Spezzano, A. Vinci, Itema: A methodological approach for cognitive edge computing iot ecosystems, Future Generation Computer Systems 92 (2019) 189–197.

[33] E. Mezghani, E. Exposito, K. Drira, A model-driven methodology for the design of autonomic and cognitive iot-based systems: Application to healthcare, IEEE Transactions on Emerging Topics in Computational Intelligence 1 (3) (2017) 224–234.

[34] L. Santos, J. Pereira, E. Silva, T. Batista, E. Cavalcante, L. J, Identifying requirements for architectural modeling in internet of things applications, IEEE, 2019.

[35] M. P. Alves, F. C. Delicato, P. F. Pires, Iota-md: a model-driven approach for applying qos attributes in the development of the iot systems, in: Proceedings of the Symposium on Applied Computing, 2017, pp. 1773–1780.

[36] X. T. Nguyen, H. T. Tran, H. Baraki, K. Geihs, Optimization of non-functional properties in internet of things applications, Journal of Network and Computer Applications 89 (2017) 120–129.

[37] B. Costa, P. F. Pires, F. C. Delicato, W. Li, A. Y. Zomaya, Design and analysis of iot applications: a model-driven approach, in: 14th Intl Conf on Dependable, Autonomic and Secure Computing, 14th Intl Conf on Pervasive Intelligence and Computing, 2nd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress, IEEE, 2016, pp. 392–399.

[38] A. Zheng, A. Casari, Feature engineering for machine learning: principles and techniques for data scientists, O'Reilly, 2018.

[39] S. Khalid, T. Khalil, S. Nasreen, A survey of feature selection and feature extraction techniques in machine learning, in: Science and Information Conference, IEEE, 2014, pp. 372–378.

[40] T. Leppänen, C. Savaglio, L. Loven, T. Jrvenp, R. Ehsani, E. Peltonen, G. Fortino, J. Riekki, Edge-based microservices architecture for internet of things: Mobility analysis case study, in: IEEE Global Communications Conference (GLOBECOM), 2019, [Accepted].

[41] T. Leppänen, Distributed artificial intelligence with multi-agent systems for mec, in: 2019 28th International Conference on Computer Communication and Networks (ICCCN), 2019, pp. 1–8.

[42] T. Leppänen, C. Savaglio, L. Lovén, W. Russo, G. Di Fatta, J. Riekki, G. Fortino, Developing agent-based smart objects for iot edge computing: Mobile crowdsensing use case, in: International Conference on Internet and Distributed Computing Systems, Springer, 2018, pp. 235–247.

[43] M. Chen, Y. Hao, Label-less learning for emotion cognition, IEEE transactions on neural networks and learning systems.

[44] M. Chen, Y. Hao, K. Lin, Z. Yuan, L. Hu, Label-less learning for traffic control in an edge network, IEEE Network 32 (6) (2018) 8–14.

[45] ETSI, Mobile Edge Computing (MEC); End to End Mobility Aspects, ETSI GS MEC 018.

[46] V. Kostakos, T. Ojala, T. Juntunen, Traffic in the smart city: Exploring city-wide sensing for traffic control center augmentation, IEEE Internet Computing 17 (6) (2013) 22–29.

[47] J. Haavisto, M. Arif, L. Lovén, T. Leppänen, J. Riekki, Open-source rans in practice: an over-the-air deployment for 5g mec, in: 2019 European Conference on Networks and Communications, IEEE, 2019, pp. 495–500.