Distributed Artificial Intelligence with Multi-Agent Systems for MEC

Teemu Leppänen
Center for Ubiquitous Computing, University of Oulu, Finland
Email: teemu.leppanen@oulu.fi

***Abstract*—In this paper, the Multi-access Edge Computing (MEC) system architecture, as defined by the ETSI standards, is modeled as a multi-agent system. MEC system management services and application execution components are designed as software agents, facilitating distributed artificial intelligence capabilities in their operation and cooperation. Further, the integration of current agent technologies into the standardized MEC system is discussed. Lastly, a case study is presented on how to integrate an existing Internet of Things agent framework and agent-based edge application seamlessly to the MEC system.**

***Index Terms*—Edge computing, Agent-based computing, Adaptive systems, Orchestration, Interoperability**

I. Introduction

The predominant Internet of Things (IoT) system model today is cloud-centric, with virtually unlimited computing resources and data storage for Big Data applications. The resulting architecture is three-tiered with cloud platforms at the top, then middleware connecting cloud-based applications to the data producing IoT devices, at the lowest layer. This model introduces latencies into the application execution, as data travels first upstream for processing and secondly the results, i.e. control commands, travel downstream to to the middleware and to the devices. In the system scale, centralized management at the cloud is no longer feasible from such a distance, due to the opportunistic environment with intermittent connectivity and low bandwidth for the data collecting devices.

Edge computing [1] is seen the next step towards the future IoT systems and networks. Edge computing leverages cloud resources into the network infrastructure components, i.e. middleware, in the close proximity of the data producing devices. This way, the communication latencies are reduced with application-specific computing resources and data at the edge. Moreover, privacy increased with localize data processing. As a result, less data is transmitted in the backbone network, which contributes towards scalability.

Multi-access Edge Computing (MEC) [2], [3] extends the role of cellular network infrastructure components, e.g. base stations, from forwarding the traffic to multi-tenant application execution platforms with local data storage capabilities. A distinct feature in MEC, in comparison with other edge computing solutions, is that it offers real-time radio access network information to enhance system optimizations with fine granularity. MEC has recently drawn both industry and academic attention, as an open multivendor platform, and is becoming one of the key technologies for 5G. MEC is currently under standardization by European Telecommunications Standards Institute (ETSI).

Artificial intelligence (AI), with machine learning (ML), has emerged as a key technology for Big Data based application development in the cloud. Recenty, such technologies are being considered also for communication systems [4]. However, as AI solutions are largely centralized and cloud based, the same challenges in latencies and handling massive-scale data follow. Here, distributed AI technologies such as software agents are one solution to handle these issues with proven technology [5], [6]. Software agents possess well-known capabilities for autonomous and intelligent operation based on reactivity, adaptivity, mobility, reasoning, learning, planning and proactivity. Moreover, such agents collaborate as a multi-agent system (MAS) to solve problems that are outside the capabilities of a single agent. These agent capabilities are seen beneficial for the system components across layers in IoT and for providing established interaction techniques for cooperation [6], [7].

In this paper, we describe a MAS based system architecture for the ETSI MEC system reference architecture as described in [8]. We show how agent capabilities provide distributed artificial intelligence methods for cooperation and resource sharing to handle the dynamicity in the MEC system environment, in both individual components and in the system level, with localized knowledge of operational environment. A case study is presented, showing an existing IoT software agent framework is seamlessly integrated into the MEC system. Lastly, we discuss how current agent technologies could be integrated into the MEC system and utilization of agent-based computing (ABC) at the edge in general.

The rest of the paper is organized as follows. In Section 2, we discuss the related agent technologies for edge and present the ETSI MEC system reference architecture. Section 3 presents the MAS based MEC system architecture and an implementation of the functionality of MEC system components with agent technologies. In Section 4, a case study is presented with an agent-based MEC service. In Section 5, we discuss the resulting MAS and identify challenges in the utilization of agent technologies in MEC and future edge computing. Section 6 concludes the paper.

II. Background

Software agent technologies and solutions that focus on edge computing have already been introduced. The integration and management of distributed services on heterogeneous system components with agents was considered in [9], [10]. Particularly for MEC, the authors in [11] propose interface agents to integrate disparate 3rd party cloud computing platforms into the system and to manage interactions between the platforms. Agent-based coordination of the 3rd party service delivery to IoT systems is addressed in [12]. In [13], control agents are utilized as a MAS to create adaptive and decentralize services for Fog computing. Moreover, Fog computing system orchestration with performance monitoring agents has been introduced in [14]. The optimization of task assignment, with negotiating agents, between cloud and edge platforms is addressed in [5]. Lastly, in [15], collaborative microservices are modeled as a MAS for IoT.

However, this paper specifically focuses on the whole ETSI MEC reference architecture as a MAS and providing agent-based roles for the MEC system components that collaborate using the standardized interactions. As no implementation of the reference architecture currently exists, our focus is on the agent-based design of the expected functionality of the system components.

A. MEC system architecture and operation

ETSI is currently standardizing a reference MEC system architecture, system components, services and their interfaces [8], as illustrated in Figure 1. Detailed information on MEC system operation is available in the specifications, whereas in this paper, an overview is presented for modeling the system as a MAS. General guidelines are given on realizing MEC systems atop the reference architecture and how to implement the system services for uniform MEC operation across application domains. A set of proof-of-concept use cases for MEC are presented, such as device location tracking, IoT gateways, content delivery and distribution and data- and computation-intensive edge applications, such as augmented reality and video analytics.

MEC system operation is based on a set of services for MEC application enablement and lifecycle management, centralized orchestration and management of the whole system, distributed management of MEC platforms consisting of MEC hosts, individual host management and managing the application deployment atop the common virtualization infrastructure. MEC hosts facilitate multi-tenancy, each application accessing services provided in the host and in the platform.

The system operation is divided into system level and host level management. On the system level, MEC applications are initiated in two ways. First, as requests from the users, e.g. through smartphone applications, by creating an application context containing the configuration of application packages, their operational requirements and the required services and resources. Second, 3rd party developers and service providers create contexts including the configuration, usage policies and

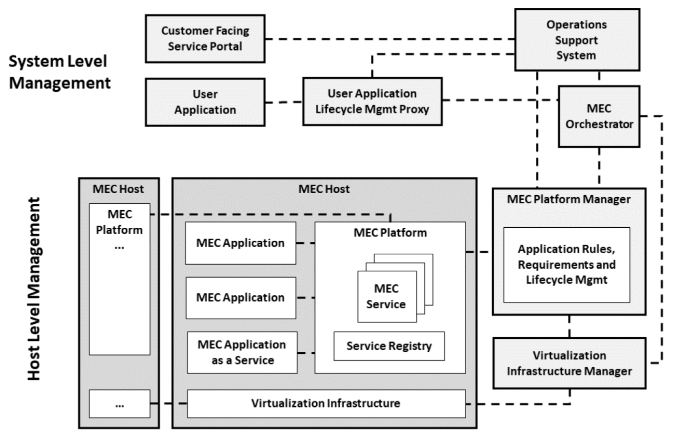

System Level Management
Customer Facing Service Portal
Operations Support System
User Application
User Application Lifecycle Mgmt Proxy
MEC Orchestrator
Host Level Management
MEC Host
MEC Platform
...
MEC Host
MEC Application
MEC Platform
MEC Application
MEC Service
MEC Application as a Service
Service Registry
...
Virtualization Infrastructure
MEC Platform Manager
Application Rules, Requirements and Lifecycle Mgmt
Virtualization Infrastructure Manager

Fig. 1. Outlined ETSI MEC architecture.

billing for provisioning their services. The contexts are authenticated in the User Application Lifecycle Management (LCM) Proxy and, regarding the required application packages, in the Operations Support System (OSS). The OSS represents the mobile network operator, providing information and managing the integration of network services in the host site locations. Through the LCM, user applications receive events related to the application execution and can, for example, request its relocation and termination.

The MEC Orchestrator (MEO) orchestrates the application execution based on the provided contexts. MEO is the central authority to control the system operation and has full visibility in real-time across the whole system, its state and resources. The MEO validates the application contexts, based on the application requirements and host capabilities, e.g. application load, available services and communication latencies. If needed, MEO adjusts the configuration to comply with system resources and operator policies. MEO prepares the onboarding of the application, selects the hosts for application execution and commands by the Virtualization Infrastructure Manager (VIM) to instatiate the packages. During the application execution, MEO can trigger relocation and terminate the application execution. OSS also has a role in the application management by providing access to possible external cloud and network services, e.g. for relocating external resources into the MEC system.

At the host management level, MEC Platform Manager (MEPM) controls the application lifecycle based on the instructions from OSS and MEO. MEPM manages platform operations and resolves resource conflicts between the hosts with regard to data traffic and use of services. The MEPM disseminates application events, such as performance information and faults to the MEO. In parallel, the platform VIM manages and allocates the virtualized resources for applications, their data storage and networking. When the application terminates, VIM releases its virtualization resources. If authorized, VIM also relocates the resources from external clouds by interacting with the remote cloud platform manager.

MEC hosts contain the ME Platform component that pro-

TABLE I
MEC SYSTEM APIS AND GENERIC APPLICATION INFORMATION AS DEFINED BY ETSI

Management and application lifecycle	Services to run the applications on platforms and hosts. Performance data of the virtualization environment with regard to specific host or applications.
End-to-end mobility	Service interruption latencies, out-of-service time, topology of host deployments and application scope aspects.
Location API	User equipment and radio node locations in radio network associated with the host. User equipment movement information. Location can be geolocation, cell identifier, etc., with different granularity.
Radio network management API	Local and external measurement information and statistics, as defined in the 3GPP specifications. Information about user equipments connected to the radio nodes in a host. Provides methods to inspect, modify and prioritize uplink and downlink traffic and configure traffic rules. Granularity can be specified over a period of time for cell, user equipment and QoS.
Bandwidth management API	Dynamic and static bandwidth requirements for each application or user equipment session.
User Identity API	Application and policy information: authorization, access control and traffic rules, information flows and service aggregation patterns.
Authentication	Trusted MEC applications and authorizations to interact with external 3rd party resources.

TABLE II
MEC SYSTEM KPIS AS DEFINED BY ETSI

Functional KPIs (per service basis)	Service continuity, energy efficiency, end-to-end / one-way latencies, throughput / goodput, loss rate, jitter, out of-order packets, QoS and MOS.
Non-functional KPIs	Lifecycle, boot-time, scalability, elasticity, availability, fault tolerance, MEC host load, number of API requests, request delay, failed requests.

vides computational, network and persistent data storage resources for the hosted applications and services, which are executed in its own virtualization infrastructure. MEC hosts support multi-tenancy for multiple applications and services within a host, but also multiple and shared instances of the applications. MEC host provides a service registry to discover, access, offer and advertise the services across the platform. The virtualization infrastructure includes a data plane, that is controlled by the ME Platform, for routing the application and service data across 3GPP, local and external networks. MEC application images are provided as virtual machines or containers, run on the virtualization infrastructure in the hosts. The applications consume the services across the platform, while interacting directly with the ME Platform and MEPM regarding their lifecycle.

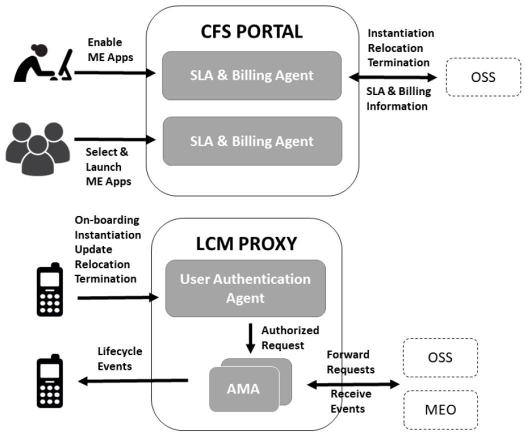
The standards specify a set of Application Programming Interfaces (API) that provide mechanisms for application enablement, deployment, mobility and lifecycle management. Information is needed from two different perspectives: (1) the whole system and operational environment state and resource usage and availability and (2) the requirements of applications with regard to location, computation and data. For this, the system management components provide service APIs for the application execution, management and orchestration of the system and monitoring the system performance. A set of APIs is briefly described in Table I, that is not exhaustive, as we focused on the APIs directly used by agents.

ETSI also defines the sets of functional and non-functional Key Performance Indicators (KPI) for evaluating the system performance, as shown in Table II. The functional KPIs include user equipment and other device connection parameters, Quality of Service (QoS) and service continuity. Non-functional KPIs include indicators in resource availability, scalability, application and host load and energy efficiency.

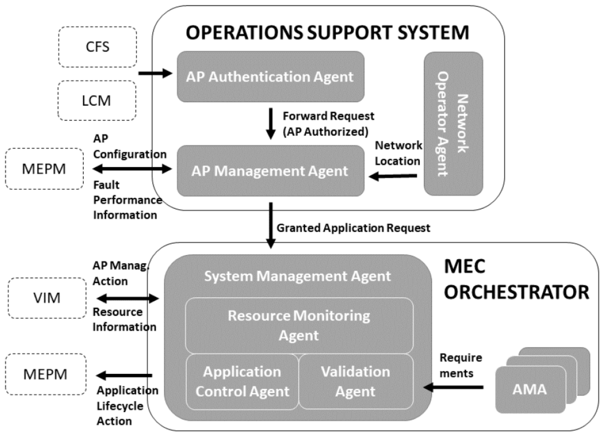
III. MEC AS A MULTI-AGENT SYSTEM

The opportunistic nature of edge computing systems justifies distributed AI approaches. Centralized orchestration and management, let alone optimization, requires a large number of different data sources and complex algorithms for analysing the collected massive-scale data. Such centralized algorithms would be difficult to design, develop, deploy, maintain and evaluate with regard to the system dynamicity. The distributed and partitioned edge application execution is well aligned with the general properties and capabilities of the software agent paradigm and the established methodologies for ABC. Agents provide means to distribute application-specific data analysis and to extract and share relevant context-aware information. In addition, well-known methods and protocols for agent-based negotiation, collaboration, cooperation and competition have been developed for MAS.

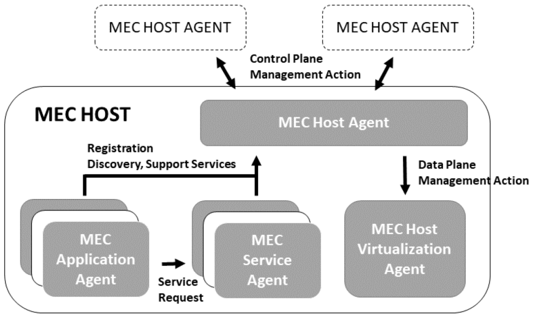
The goal of MEC is to have the applications at the right place at the right moment, aiming to improve the Quality of Experience (QoE) for the users and availability of the distributed system resources. A key is to make the system context-aware and less unpredictable. At the MEC system level, challenges are introduced by multiple applications with dynamically changing application contexts that compete for the same resources. The authors in [2] identify challenges in MEC system orchestration: (1) decision-making on application offloading to MEC and distribution to minimize its resource use, (2) the efficient allocation of the resources within a platform, and (3) mobility management overhead and minimizing application migrations on the backbone. On the network level, MEC is a complementary technology to the Software-Defined Networking (SDN) architecture, reusing the same network infrastructure. Common SDN architecture considers network controllers based on software agents, e.g. to implement application-aware system control [16].



(a) User and client side components



(b) System management and orchestration components


MEC HOST AGENT
MEC HOST AGENT
Control Plane Management Action
MEC HOST
MEC Host Agent
Registration Discovery, Support Services
Data Plane Management Action
MEC Application Agent
Service Request
MEC Service Agent
MEC Host Virtualization Agent
(c) MEC host components

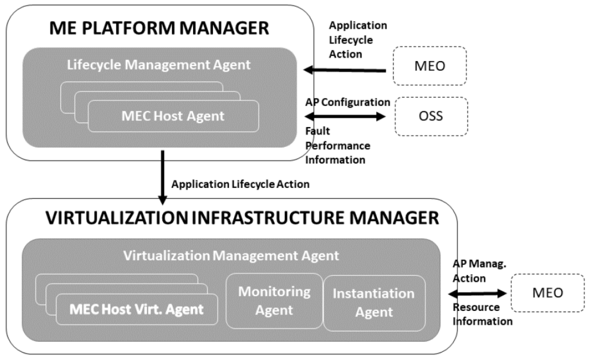

ME PLATFORM MANAGER
Application Lifecycle Action
Lifecycle Management Agent
MEO
AP Configuration
MEC Host Agent
OSS
Fault Performance Information
Application Lifecycle Action
VIRTUALIZATION INFRASTRUCTURE MANAGER
Virtualization Management Agent
MEC Host Virt. Agent
Monitoring Agent
Instantiation Agent
AP Manag. Action
Resource Information
MEO
(d) MEC platform management components

Fig. 2. The MEC system as a multi-agent system.

ETSI models the MEC applications as a set of autonomous and loosely-coupled microservices [17], based on containers. The containerized application functionality can be partitioned and easily distributed in the system, as each microservice can be developed with different technologies by different stakeholders and deployed independently. As in [15], microservices can be considered as agents.

The standardized interaction mechanism in MEC is the Representational State Transfer (REST) architectural principles [8]. The system services, applications and 3rd party components are required to provide REST-based API. In REST, the main abstraction is a resource that is named and given a unique identifier and an address (an URI). The resource has representation of its current value, e.g. a service content. REST then provides a simple interface to access and manipulate the resource, e.g. with Web technologies the standardized HTTP methods (e.g. GET, POST, PUT and DELETE). REST-based interactions follow the client/server model, as in the modern Web. Therefore, the agents are expected to follow the REST paradigm in their interactions with the system components.

A. Agent-based MEC system architecture

Next, we outline the MEC reference system architecture based on a MAS, where the system components are modeled as software agents. These agents interact through the MEC APIs to facilitate system-wide cooperation and expose their own MEC service APIs. The ETSI defined KPIs provide system-wide information for the agents to plan, evaluate and improve their behavior towards individual and common goals with regard to the expected system operation. However, we dont assume that all MEC system components are to be designed as software agents. Instead, we envision that agent capabilities are utilized wherever distributed AI is seen beneficial by the stakeholders.

To describe the MEC system as a MAS, we present the agents realizing the roles of MEC system components and their interactions. The software agents are deliberative, with capabilities for reactive and adaptive behavior, reasoning and planning towards their individual goals, evaluating their behaviors and learning to exhibit proactive behavior. The agent behaviors are derived from the MEC specifications, based on the agent's role in representing the particular system component. As this paper presents an initial architectural design, security and trust in ABC are not considered. For reference, the utilized agent abbreviations are listed in Table III.

Figure 2a shows the user- and client-side components, where we place the User Authentication Agents that authorize the requests for application execution and control. After authorization, the Application Management Agents (AMA) create and/or maintain the application contexts. Here, it is beneficial for the agents to evaluate the requests and application contexts collaboratively, e.g. if multiple users access the same application or multiple applications are accessed by the same user. In such a case, agents could operate as a representative to the aggregated requests in the system-side, maintaining the contexts provided by the users and operating as an intermediate towards the OSS and MEO. Such an agent reduces the need for synchronous connections with the user application and further operates as an intelligent proxy in the opportunistic environment.

For the application or service provisioning, Service-Level Agreement (SLA) and Billing Agents represent the service providers that, with the client-side AMA agents, make agreements for their service usage. The benefit, in comparison with non-agent approaches, is the capability to utilize the well-known negotiation and for example auctioning protocols developed for agents. The service providers could then be easily abstracted as brokers in such scenarios.

The role of the OSS (Figure 2b) is to handle the request from AMA and to authorize the application contexts for the deployment. This is done by the Application Authentication (APA) Agents that collaboratively, and with the AMA agents, assess the multi-user or multi-application requests and the requirements in the contexts. This is beneficial to reduce overlapping and duplicate requests by the same of different stakeholders. The Application Management Agents (APMA) check the rationality of the contexts in collaboration with the Network Operator Agent (NOA). The NOA represents the mobile network resources, e.g. operator policies and host site locations. At best, the cooperation of APMA and NOA agents leads to deployments, where the agents can proactively come up with common plans for providing the functionality for the applications. Agent-based ML can be used to evaluate and optimize the plans further with the performance information received from MEPM. The APMA and AMA agents can be internally organized as MAS.

The MEO is the most complex component in the system. The ETSI specifications allow it to be a distributed component, but no further details are given. Naturally, its functionality could be designed as a MAS, where individual agents or sub-MAS represent the subsystems (Figure 2b). The Validation Agents checks that the application contexts with their KPIs can be met. The System Management Agent (SMA) is the authority to make decisions on the system level regarding application lifecycle, directing the VIM and MEPM and on the system resource usage, with regard to the required KPIs. The SMA needs to contain all the system information as outlined in the specifications. However, the information is to be distributed according to the agent roles in the MAS, each agent providing its own refined context-aware information for the decision-making. This is where the agent paradigm shows its strengths, as a number of organizational, planning, orchestration and learning strategies are available for MAS in general. Therefore, the SMA architecture is hierarchical and hybrid, consisting of sub-MAS and agents with different

TABLE III
LIST OF THE MEC SYSTEM AND AGENT ABBREVIATIONS

ACA	Application Control Agent
AMA	Application Management Agent
APA	Application Authentication Agent
APMA	Application Management Agent
LCM	User Application Lifecycle Management
LMA	Lifecycle Management Agent
MEO	MEC Orchestrator
MEPM	MEC Platform Manager
MHA	MEC Host Agent
MHVA	MEC Host Virtualization Agent
NOA	Network Operator Agent
OSS	Operations Support System
RMA	Resource Monitoring Agent
SMA	System Management Agent
VIM	Virtualization Infrastructure Manager
VMA	Virtualization Management Agent

roles. Due to the internal complexity, learning of SMA is a combined effort, where agent-based ML with limited goals are facilitated for the subsystem optimization. The Application Control Agent (ACA) handles the system level lifecycle management of the applications. The Resource Monitoring Agent (RMA) maintains the overall view of the system performance and resource use, deployed application packages and logical network topologies. The ACA and RMA can be designed in a number of ways as a MAS, e.g. based on application-specific agents, platform- or host-specific agents or resource type-specific agents.

On the MEC platform level, the agent-based MEPM is described as illustrated in Figure 2d. The Lifecycle Management Agent (LMA) receives application execution instructions from the SMA and makes the decisions on their realization. The autonomy of the LMA is therefore limited, but it is responsible for platform-level resource usage, application execution and relocation, service provisioning and data routing. The LMA can also be a hybrid MAS, where MEC Host Agents (MHA) represent hosts to make the decisions collaboratively based on different organizational strategies. The MHAs are depicted as a part of LMA, to emphasize collaboration in a MAS, even when they are considered semi-autonomous authorities on the hosts. The LMA reports the application performance information, e.g. latencies and possible conflicts, to the MEO.

The agent-based VIM (Figure 2d), consists of the Virtualization Management Agent (VMA) including MEC Host Virtualization Agents (MHVA) that represent the platform hosts. The VMA receives the instructions of the deployment and relocation of the application packages from the LMA and SMA, thus it has limited autonomy. Moreover, VMA includes Instantiation Agent (IA) that represents the SMA, controls the deployment of application packages into the virtualization infrastructure with MHVAs. The virtualization infrastructure monitoring is performed by the Monitoring Agent (MA) that reports back to the SMA. It collects the historical information about performance and can suggest optimization and strategies for the deployment within the platform.

The MEC hosts, depicted in Figure 2c, are controlled by the MHA and MHVA. The MHA is a part of the platform-level LMA, but also exhibit limited autonomy based on the on-site deployment rules from the NOA. The MHA collaborates with MHAs on other hosts, as a MAS, to execute the requests as control plan actions. Moreover, MHA controls the host data traffic on the data plane for the applications and services. Additional agents can be deployed to represent the applications and services in the host, which collaborate with the MHA and possibly other agents in their own execution. However, in agent-based MEC host, it is mandatory also to host non-agent applications and services by using the standardized mechanisms.

B. Agent technologies for MEC

Foundation for Intelligent Physical Agents (FIPA) has specified a set of standards[1] for agent system architectures and services, agent platforms and agent communication languages and interaction protocols. The FIPA specifications address message transport services in agent systems, including the use of HTTP as a message transport protocol, where XML-based envelopes are used to describe the agent interactions. This combination allows rich vocabulary and ontologies for agent interactions, that are encapsulated into structured documents. This approach is closely related to the Service-Oriented Architectures (SOA) paradigm and Web service specifications as WS-*. in SOA, XML-based Simple Object Access Protocol messages are transmitted in the HTTP message payload. In SOA-based MAS, the services for agents and exposed by the agents are exposed through a XML-based description language that details the service functionality and access methods. These services in agent platforms are then accessed with RPC mechanisms.

However, from the agent perspective, REST in MEC is fundamentally different paradigm as SOA in FIPA. The agent interactions and their semantics must be reconsidered through the resource abstraction and the small set of HTTP methods. A solution could be to provide wrapper interfaces that translate agent communication language messages into REST-based requests as in [18]. Well-known agent platforms, such as JADE [19], provide these wrappers to utilize HTTP as the agent message transport protocol. Nevertheless, solutions are needed to maintain the semantical interoperability of the messages as in [20]. A step towards full agent-based Web applications is to utilize the native Web technologies as agent architecture, e.g. HTML5 [21]. However, such agents are tightly-coupled into the transferred Web documents.

A fully REST and FIPA compliant agent architecture and ROAgent framework for IoT was presented in [22]–[24]. The resource abstraction is built-in into the agent architecture, the agent platform operation and interactions and to the services provided by the framework. These agents are therefore natively and seamlessly integrated into the Web, through a RESTful API for interactions with agents, services and other Web components. This framework was extended towards edge computing in [25], [26]. However, these agents are largely reactive, without the deliberative capabilities leading to autonomous intelligent operation, e.g. reasoning, planning, making decisions and learning from the system behavior.

[1]http://www.fipa.org/

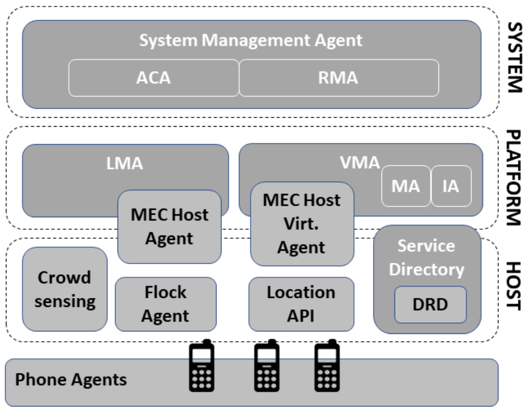


Fig. 3. MAS-based crowdsensing service for MEC.

Solutions towards deliberative agents have been presented. In [27], an example if given how cognitive modules for agents can be integrated into existing agent platforms, i.e. JADE. Integration of such solutions would lead to an MAS based on heterogeneous hybrid agents with reactive and deliberative capabilities as needed for the task at hand and according to the resources available. As an example for agent-based ML, reinforcement learning is widely utilized for online learning in MAS [28]. Challenges include how to define common learning goals and how to distribute and coordinate the learning between agents.

IV. Case study

We demonstrate the MAS-based edge service in MEC system, based on our previous work in [25]. An agent-based edge service provides user mobility traces, based on how their devices are connected to the edge system, and detects flocks of users moving to the same direction. The flock information is provided as service content for crowdsensing applications running at the edge.

Figure 3 illustrates agent-based system architecture design atop the MEC system architecture. As the user devices, i.e. smartphones, are connected to a MEC host through an AP under its domain, their location is known and reported by the MEC Location API. The Flock agent implements the flock detection service logic, based on the information it receives through the Location API. The crowdsensing application is implemented and deployed as a MEC application. Therefore, it needs to describe the Flock Agent as one of the required MEC services on the host. Such applications can then query available flocks that meet their requirements, e.g. user in a geographical area at given time. The application the controls the sensing task execution on the user devices in collaboration with the Phone Agents [29].

Further, the Flock agents can operate as a Flock MAS to provide collaborative large-scale MEC service, i.e. mobility traces across the whole system. Simultaneously with the Flock agent, the information provided by the Location API is utilized by the MEO and MEPM to orchestrate the service, e.g. instantiate or terminate Flock agents in the hosts, based on the requirements of the crowdsensing applications.

As described in detail in [25], the agent-based service is implemented with the ROAgent framework. Therefore, the agents already interact through RESTful interfaces, which provides seamless integration into the MEC system and gives access to its services. The Flock Agent exposes a RESTful API that the MEC system components can utilize. In addition, the Crowdsensing application and Flock Agent are expected to implement the requires contexts and the required standardized MEC APIs. To fully integrate the ROAgent framework into MEC system, the Distributed Resource Directory (DRD) peer [30] is integrated into the MEC host service discovery component. Through the DRD, the agents register themselves to the agent framework as resources and to the MEC system as services.

This straightforward case study shows how REST-based agent framework can be seamlessly integrated into the standardized MEC system reference architecture. The benefit is that interactions between agents and MEC system components are all based on the REST paradigm, with no extra effort in implementing interface wrappers for REST or proxies for protocol translation.

V. Discussion

AI on the edge is seen to have increasing importance in realizing distributed IoT applications [1], [4]. Also the ETSI specifications consider use cases, where AI is seen beneficial in the context of MEC. The challenges of the opportunistic environment needs to be addressed, at best proactively by distributed intelligent system management. Resource conflicts and the evident user mobility in such a dynamic environment needs collaborative intelligence across system layers and components to maintain the required KPIs. Moreover, integration of 3rd party services facilitates widespread MEC resource utilization, increasing the need for content-aware operation.

The utilization of agent technologies in IoT requires efforts at agent standardization and technologies for interoperability between agents and IoT systems. Recent solutions have been proposed, e.g. [22], that addressed interoperability between a standardized IoT framework and well-known agent standards, e.g FIPA. Moreover, steps are already taken towards utilizing the deliberative capabilities of agents in edge computing [11]. ML is an important agent capability towards autonomous operation, facilitating proactivity. MEC standards offer a set of KPIs for the functional evaluation of application execution with focus on network parameters, such as latencies. The non-functional KPIs, such as elasticity, offer means to evaluate the cognitive capabilities of agents. Lastly, the underlying communication technologies in edge computing are utilizing agents in similar roles, as with SDN [16]. Integration of these network level agents into the MEC system opens up possibilities for fine-grained optimization based on cooperation between agents.

In [31], the requirements for future AI-enhanced edge computing platforms are outlined. In this respect, the agent based MEC system facilitates partial autonomy and intelligence for the individual system components with local and global data, standardized means for interoperability across hosts, platforms and systems, on-demand (logical) network topologies for distributed applications and context-awareness to address challenges in environment dynamicity.

VI. Conclusion

This paper proposed the modeling of the ETSI MEC system reference architecture as a MAS, that brings distributed AI capabilities for MEC system operation and optimization, particularly for the system orchestration. With the software agent properties, such as autonomy, reactivity, adaptivity, cooperation, learning and proactivity such intelligence can be embedded in the system components. A case study demonstrated how an existing agent framework can be integrated seamlessly to the MEC system through the REST architectural principles.

Software agent capabilities contribute towards future AI-enhanced edge computing, where ETSI MEC is now considered as a part of 5G standardization. Our future work aims at real-world prototypes, atop our real-world 5G edge computing testbed in the premises of the University of Oulu [32], for studying the different agent and MAS capabilities in distributed MEC systems.

Acknowledgment

This work is finincially supported by the Academy of Finland 6Genesis Flagship (grant 318927) and the Future Makers program of the Jane and Aatos Erkko Foundation and the Technology Industries of Finland Centennial Foundation.

References

[1] C. Li, Y. Xue, J. Wang, W. Zhang, and T. Li, ”Edge-oriented computing paradigms: A survey on architecture design and system management,” ACM Computing Surveys, vol. 51, no. 2, article 39, 2018.

[2] P. Mach, and Z. Becvar, ”Mobile edge computing: A survey on architecture and computation offloading,” IEEE Communications Surveys & Tutorials, vol. 19, no. 3, pp. 1628-1656, 2017.

[3] Y. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, ”Mobile edge computing - A key technology towards 5G,” ETSI White Paper, no. 11, 2015.

[4] Q. Mao, F. Hu, and Q. Hao, ”Deep learning for intelligent wireless networks: A comprehensive survey,” IEEE Communications Surveys & Tutorials, vol. 20, no. 4, pp. 2595-2621, 2018.

[5] T. Suganuma, T. Oide, S. Kitagami, K. Sugawara, and N. Shiratori, ”Multiagent-based flexible edge computing architecture for iot,” IEEE Network, vol. 32, no. 1, pp. 16-23, 2018.

[6] G. Fortino, W. Russo, C. Savaglio, W. Shen, and M. Zhou, ”Agent-oriented cooperative smart objects: From IoT system design to implementation,” IEEE Transactions on Systems, Man, and Cybernetics: Systems, vol. 99, pp. 1-18, 2017.

[7] C. Savaglio, G. Fortino, M. Ganzha, M. Paprzycki, C. Badica, and M. Ivanovic, ”Agent-based computing in the Internet of Things: a survey,” In: International Symposium on Intelligent and Distributed Computing, pp. 307-320, 2017.

[8] ETSI, ”Mobile Edge Computing (MEC): Framework and Reference Architecture,” ETSI GS MEC 003, 2016.

[9] Y. Jararweh, A. Doulat, A. Darabseh, M. Alsmirat, M. Al-Ayyoub, and E. Benkhelifa, ”Sdmec: Software defined system for mobile edge computing,” In: IEEE International Conference on Cloud Engineering Workshop, pp. 88-93, 2016.

[10] V. Bumgardner, V. Marek, and C. Hickey, ”Cresco: A distributed agent-based edge computing framework,” In: 12th International Conference on Network and Service Management, pp. 400-405, IEEE, 2016.

[11] V. Shah, ”Multi-agent cognitive architecture-enabled IoT applications of mobile edge computing,” Annals of Telecommunications, vol. 73, no. 7-8, pp. 487-497, 2018.

[12] J. Wang, Q. Zhu, and Y. Ma, ”An agent-based hybrid service delivery for coordinating internet of things and 3rd party service providers,” Journal of Network and Computer Applications, vol. 36, no. 6, pp. 1684-1695, 2013.

[13] A. Giordano, G. Spezzano, and A. Vinci, ”Smart agents and fog computing for smart city applications,” In: 1st International Conference on Smart Cities, pp. 137-146, 2016.

[14] F. Bonomi, R. Milito, P. Natarajan, and J. Zhu, ”Fog computing: A platform for internet of things and analytics,” Big data and internet of things: A roadmap for smart environments, pp. 169-186, Springer, 2014.

[15] P. Krivic, P. Skocir, M. Kusek, and G. Jezic, ”Microservices as agents in iot systems,” In: KES International Symposium on Agent and Multi-Agent Systems: Technologies and Applications, pp. 22-31, 2017.

[16] Z. Qazi, J. Lee, T. Jin, G. Bellala, M. Arndt, and G. Noubir, ”Application-awareness in SDN,” In: ACM SIGCOMM computer communication review, vol. 43, no. 4, pp. 487-488, 2013.

[17] D. Sabella, V. Sukhomlinov, L. Trang, S. Kekki, P. Paglierani, R. Rossbach, et al., ”Developing Software for Multi-Access Edge Computing,” ETSI White Paper, no. 20, 2nd ed., 2019.

[18] L. Braubach, and A. Pokahr, ”Conceptual integration of agents with wsdl and restful web services,” In: International Workshop on Programming Multi-Agent Systems, pp. 17-34, 2012.

[19] F. Bellifemine, G. Caire, A. Poggi, and G. Rimassa, ”JADE: A software framework for developing multi-agent applications. Lessons learned,” Information and Software Technology, vol. 50, no. 1-2, pp. 10-21, 2008.

[20] P. Pico-Valencia, J. Holgado-Terriza, and J. Senso, ”Towards an Internet of Agents model based on Linked Open Data approach,” Autonomous Agents and Multi-Agent Systems, vol. 33, no. 1-2, pp. 84-131, 2019.

[21] J. Voutilainen, A. Mattila, K. Systä, and T. Mikkonen, ”HTML5-based mobile agents for Web-of-Things,” Informatica, vol. 40, no. 1, 2016.

[22] T. Leppänen, ”Resource-oriented mobile agent and software framework for the Internet of Things,” Dissertation, Acta Universitatis Ouluensis, C Technica, no. 645, University of Oulu, Finland, 2018.

[23] T. Leppänen, J. Riekki, M. Liu, E. Harjula, and T. Ojala, ”Mobile Agents-based Smart Objects for the Internet of Things,” In: Fortino and Trunfio (eds.), Internet of Things based on Smart Objects: Technology, Middleware and Applications, pp. 29-48, Springer, 2014.

[24] T. Leppänen, M. Liu, E. Harjula, A. Ramalingam, J. Ylioja, P. Närhi, et al., ”Mobile Agents for Integration of Internet of Things and Wireless Sensor Networks,” In: IEEE International Conference on Systems, Man and Cybernetics, pp. 14-21, 2013.

[25] T. Leppänen, C. Savaglio, L. Lovén, W. Russo, G. Di Fatta, J. Riekki, and G. Fortino, ”Developing Agent-based Smart Objects for IoT Edge Computing: Mobile Crowdsensing Use Case,” In: 11th International Conference on Internet and Distributed Computing Systems, pp. 235-247, 2018.

[26] T. Leppänen, and J. Riekki, ”Energy Efficient Opportunistic Edge Computing for the Internet of Things”, Web Intelligence, IOS Press, 2018. [To Appear]

[27] L. Braubach, A. Pokahr, and W. Lamersdorf, ”Jadex: A BDI-agent system combining middleware and reasoning,” In: Software agent-based applications, platforms and development kits, pp. 143-168, Birkhäuser Basel, 2005.

[28] L. Busoniu, R. Babuska, and B. De Schutter, ”A comprehensive survey of multiagent reinforcement learning,” IEEE Transactions on Systems, Man, and Cybernetics, Part C, vol. 38, no. 2, pp. 156-172, 2008.

[29] T. Leppänen, J. Álvarez Lacasia, Y. Tobe, K. Sezaki, and J. Riekki, ”Mobile Crowdsensing with Mobile Agents,” Autonomous Agents and Multi-agent Systems, vol. 31, no. 1, pp. 1-35, 2017.

[30] M. Liu, T. Leppänen, E. Harjula, Z. Ou, A. Ramalingam, M. Ylianttila, et al., ”Distributed resource directory architecture in Machine-to-Machine communications,” In: 9th IEEE International Conference on Wireless and Mobile computing, Networking and Communications, Workshop on the Internet of Things Communications and Technologies, pp. 319-324, 2013.

[31] E. Peltonen, T. Leppänen, and L. Lovén, ”EdgeAI: Edge-native Distributed Platform for Articial Intelligence,” In: 6G Wireless Summit, Levi, Finland, 2019.

[32] J. Haavisto, M. Arif, L. Lovén, T. Leppänen, and J. Riekki, ”Open-source RANs in practice: an over-the-air deployment for 5G MEC,” In: European Conference on Networks and Communications (EuCNC2019), June 18-21, 2019. [To Appear]