Web Intelligence (2018)
IOS Press

Energy Efficient Opportunistic Edge Computing for the Internet of Things

Teemu Leppänen and Jukka Riekki
Center for Ubiquitous Computing, University of Oulu, P.O. Box 4500, FI-90014 University of Oulu, Finland

Abstract. Edge computing in Internet of Things enhances application execution by retrieving cloud resources to the close proximity of resource-constrained end devices at the edge and by enabling task offloading from end devices to the edge. In this paper, edge computing platforms are extended into the data producing end devices, including wireless sensor network nodes and smartphones, with mobile agents. Mobile agents operate as a multi-agent system on the opportunistic network of heterogeneous end devices, where the benefits include autonomous, asynchronous and the adaptive execution and relocation of application-specific tasks, while taking into account local resource availability. In addition to the vertical edge connectivity, mobile agents enable horizontal sharing of the information between end devices. Use cases are presented, where mobile agents address challenges in current edge computing platforms. An edge application is evaluated, where mobile agents as a multi-agent system process sensor data in a heterogeneous set of end devices, control the operation of the devices and share their results with system components. Mobile agents operate atop a REST-based mobile agent software framework that relies on embedded Web services for interoperability. A real-world evaluation and large-scale simulations show that energy consumption is reduced significantly in the edge application execution.

Keywords: Internet of Things, Multi-agent systems, Mobile agents, Representational state transfer, Embedded Web services

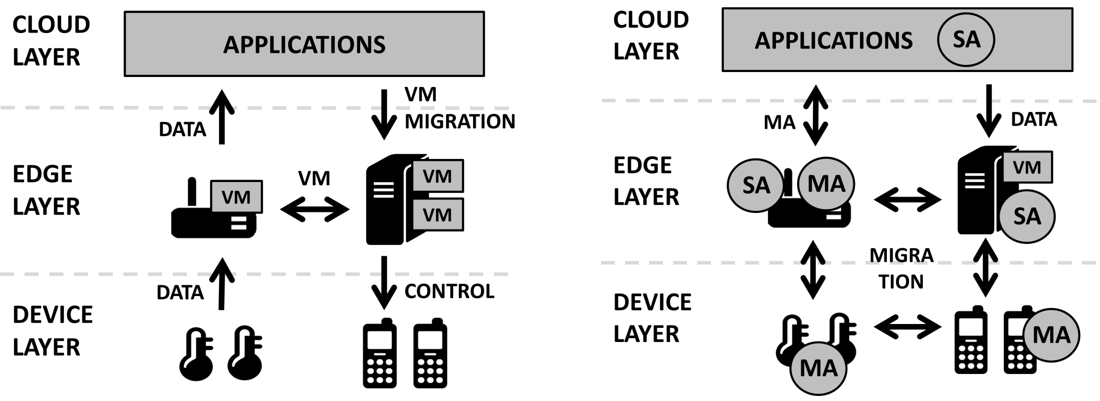
1. Introduction

Internet of Things (IoT) has emerged as the next-generation global distributed application platform. IoT applications rely on large-scale networks of heterogeneous things. IoT system architectures interconnect networked things to applications and services, allowing large-scale contributions of information about real-world phenomena. Things have a sensing, processing, actuation and communication capabilities that enable interactions with both the physical and virtual environment with a diverse set of communication technologies. Deployments include stationary things, such as wireless sensor networks (WSN), and mobile things, such as sensor-equipped smartphones. To handle the heterogeneity of participating things, infrastructure and application components, standardized means for interoperability are required. Currently, the Web technologies are the universal solution that provide standardized protocols and interfaces for communications.

Today, common IoT system architectures are based on hierarchical organizations of system components into layers [8], as illustrated in Figure 1. Applications at the top layer rely on cloud computing infrastructures that provide virtually unlimited data storage and data analysis capabilities with global connectivity. In the middle, a network layer contains middleware and network infrastructure components to handle bi-directional communications, i.e. data traffic to the cloud and application-specific system control to the lower layers. At this layer, the components typically have computational power that can be harnessed for example for edge computing [44], hence the layer is also called edge layer. The lowest layer is a device layer, where the data producing things, i.e. low-power resource-constrained embedded devices and smartphones, operate.

This cloud-centric layered approach is well justified from the point-of-view of client applications and end users. Layers hide the heterogeneity of technologies and complexities from the application developers at the cloud-side. However, the cloud-based application execution model is not without challenges [44,66].

This is the accepted version of the work.
The final publication is available at IOS Press through https://www.iospress.nl/journal/web-intelligence-and-agent-systems/



(a) VM-based edge computing architecture (b) Software agent (SA) and mobile agent (MA) use cases

Fig. 1. Distributed approaches for IoT system architectures.

Operational latencies and bandwidth requirements for the core network increase with large-scale data upload to the infrastructure side. Data upload is resource consuming for battery-operated and mobile devices, let alone with slow and unreliable connections. IoT systems are inherently dynamic as devices at the device layer join and leave the systems at will. Moreover, access to the system services can't be guaranteed, which degrades the expected quality of service for applications. Lastly, the data producing end devices are largely considered external to the application execution. When the end devices are both data producers and consumers, the inherent resource consuming data round-trip to the cloud is to be justified.

Edge computing [44,66] proposes a partial solution to these challenges. Application-specific cloud resources are leveraged to the resource-rich infrastructure devices at the logical edges of the IoT networks. Existing edge computing solutions, e.g. [6,13,53,62] rely on virtual machines (VM) to bring tailored application resources to the edge through code mobility. VM-based applications are then executed in close proximity to end users and devices, generally improving quality of service and providing high bandwidth and low latency. See Figure 1a for the illustration of the VM-based edge architecture. Furthermore, at the device layer, opportunistic computing [11] partitions application execution between the infrastructure components and end devices. The end devices are pooled as a dynamic application execution platform and their resources, e.g. sensors, data and connections, are opportunistically shared. End device mobility can be exploited in information delivery in a dynamic network topology.

In this paper, we extend IoT edge computing platforms with mobile agents to distribute application execution at the device layer as a complementary technology that further improves the energy efficiency in an edge application execution. Mobile agents are software agents that are capable of autonomously controlling their own execution and migration between devices in distributed systems, based on on their observations and interactions on the operating environment [32]. As a multi-agent system (MAS), a number of mobile agents can cooperate and collaborate through the sharing of their data and results. The novelty of this work lies in the integration of mobile agents as a MAS into IoT edge computing platforms. This approach enables reactive and adaptive edge application execution as application-specific tasks autonomously move with regard to the local resource availability in the layered architecture horizontally and vertically. Benefits are shown for the energy efficiency of resource-constrained IoT devices and in distributed opportunistic application execution based on sharing of information between system components. Figure 1b illustrates software and mobile agent operations, that are addressed later in this paper, on the different layers of the same IoT system architecture.

The rest of paper is organized as follows. In Section 2, the current edge computing solutions are reviewed and main challenges identified. In Section 3, it is discussed how mobile agents are beneficial for edge computing at the device layer. In Section 4, use cases for mobile agents in IoT edge computing are presented. A Web-integrated software framework for mobile agents [34] is presented for IoT edge computing in Section 5. An illustrative IoT edge application based on mobile agents as a MAS is evaluated in real-world settings in Section 6 and the evaluation results discussed in Section 7. Lastly, Section 8 concludes the contributions of the paper.

2. Edge computing

When IoT systems scale up, challenges in the cloud-based application model include dynamicity in the system environment, latencies in application execution and increased core network load. Edge computing aims to reduce application execution load and latencies in end devices in two ways. First, by bringing application resources, e.g. data and computational power, into the network infrastructure close to the end devices. Second, by enabling the end devices to offload heavy computations to the edge layer for execution.

The architectural design in edge computing is vertical, as illustrated in the Figure 1a. This vertical approach including the edge layer is beneficial for separation of concerns between layers and for abstracting heterogeneous communication technologies and device capabilities. As defined in [66], any resource-rich device between the cloud and data sources can be considered as an edge device. Typically, edge devices include access points, routers, gateways, cellular base stations and VM-based data centers. Generally, end device operation on the lower layer is controlled by edge devices. The end devices upload data, offload tasks to the edge or request application-specific resources to their proximity. These resources are provided as VMs that are either pushed by the cloud, pulled by an edge device or invoked by synthesis at the edge. At the edge layer, VMs may migrate between edge devices based on user movement.

A number of edge computing solutions have been proposed. In this Section, these solutions are outlined and their main challenges identified.

2.1. Cloudlets

Cloudlets are self-managing VM-based "data centers in a box" in edge devices that have Local Area Network (LAN) connections [62,63]. Cloudlet-based applications are pre-partitioned as VMs that form a distributed cloud infrastructure, where each VM shares its resources to a few end users at a time. The deployment of cloudlets is open, e.g. bottom-up by business owners or top-down by service providers. VMs can be deployed in a number of ways: (1) VM migration, where existing VM is suspended and transferred to a cloudlet near the user, (2) VM handoff, i.e. the adaptive live migration of the memory state of a VM based on user movement [21] and (3) VM synthesis, based on a small VM overlay offloaded from the end device. A base VM in the edge device takes care of cloudlet platform setup, launches user VM's and cleans up resources after use. After VM-based application execution, only the results are uploaded to the cloud, which reduces the core network load.

General challenges with cloudlets include resource optimization in varying user and applications requirements, sharing VMs between applications, interactions between cloudlets and the heterogeneity of cloudlet platforms and VM technologies. Cloudlets are large in size [62]: a VM overlay size is 100-200 MB and the size of a full VM is in the range of gigabytes. This makes VM migration highly dependent on the network capability. Therefore, migration can take minutes, but with VM handoff latencies can be reduced to tens of seconds [21,62].

2.2. Mobile edge computing

Mobile edge computing (MEC) extends the role of cellular network base stations, from forwarding network traffic, to running services and storing content in VM's hosted in MEC servers [45,53]. A MEC server aims to reduce end-to-end delays and the power consumption of mobile applications by enabling offloading, tailored local service execution and by content downloading and caching. Locally generated content can be shared through the MEC server within the connected devices. Software updates can be rolled out to the mobile devices from a MEC server.

Generally, challenges in MEC include the handling of user mobility and offloading considerations, resilience and connectivity between MEC servers, security and privacy and service charging policies [53]. MEC framework and architectural specifications are currently progressing towards standardization [45].

2.3. Fog computing

Fog computing provides a virtualized location-aware distributed edge computing platform with content storage and networking services in between end devices and cloud [6,25,74]. A fog platform consists of dense deployment of fog nodes, i.e. edge devices with a LAN connection, in a single-hop distance from the end devices. Fog nodes are organized in a hierarchy, where first-tier nodes operate with short latencies to collect and process data and control the end devices. Second-tier nodes provide virtualization and reporting with latencies up to minutes.

Challenges in fog computing include dynamic network access and fog node utilization due to end device mobility. Stationary end devices offload their data processing to the nearest fog node, but for mobile devices, the VMs are required to migrate between fog nodes. Other challenges include application-aware resource discovery, provisioning and planning VM migration while also trying to predict user movement and minimize latencies. Maintaining quality-of-service depends on the reliability of network connections and capacity with different workloads. Unified communication interfaces are needed for interoperability. Lastly, security and privacy are general concerns in the distributed remote execution of applications.

2.4. Mobile cloud computing

Mobile cloud computing (MCC) allows mobile application offloading to the edge, which provides on-demand resources for application execution [13,15, 75]. With MCC, tasks can be partitioned between end devices and cloud resources [15]. However, this requires the preinstallation of application components and predefined interactions between mobile end devices and edge devices. Moreover, VM migration is centrally controlled and heavy in terms of resource consumption in mobile devices. VM offloading requires careful consideration as VM transport can consume more energy than local execution in the device [15]. The benefits are seen in reduced application execution latences and increased scalability with location-independent resource access.

Challenges are related to static and dynamic offloading, offloading profiling and scheduling, resource monitoring and tracking, context-awareness and the availability of remote services. Typically, mobile devices suffer from low bandwidth connections, where network characteristics often change due to mobility. Such a dynamic environment may require re-offloading that is resource consuming.

2.5. Device layer distributed computing

In the device layer, large amounts of data are generated in real-time and uploaded to the cloud through the network infrastructure. A cloud platform monitors and controls the application execution and lower-layer system component and device operation. This model introduces challenges in system scalability, management and data transportation on the core network [20]. Centralized cloud-based system resource control is achieved in vertical top-down manner in closed application silos, even when in a silo the application execution may be partitioned. In this model, the end devices largely rely on vertical publish-subscribe protocols to receive application-specific data [28]. The other way around, vertical offloading reduces the application execution load of the end devices, but does not address data transport load from the data producing devices.

With in-network data processing at the device layer and proximity based direct device-to-device communications [8,12,31,44,48,68], it becomes possible to exploit the computational power of the end devices. This extends the roles of the devices from data producers to a dynamic set of data producers, consumers and interaction mediators. Application-specific code can be downloaded into the device for in-network processing, that reduces the volume of transmitted data. Now the end devices are invited into and configured at runtime as an application-specific federated computational platform.

Distributed computing on the device layer focuses on collaboration and partitioning of application execution load across the IoT architecture layers. Mist computing [43,55] involves sensor and actuator devices in the application-specific data processing. The goal is to introduce proactivity with self- and context-awareness for edge application execution. However, the presented platforms are centrally controlled and based on the service-oriented architecture paradigm [43]. Other proposed solutions include mobile edge clouds [14], MCC [15,20], femtoclouds [22] and mobile fog [25]. Other tailored solutions, e.g. [18,58,69], rely on centralized control to distribute application execution to the edge devices and resource-rich end devices. In [18], interactions are based on Web technologies. In [57], data is opportunistically preprocessed in the mobile devices and uploaded to the edge based on applications requirements. Whenever resources on the participating mobile devices become limited, the computations are offloaded to the edge. In [69] edge devices independently schedule provisioning application execution. Distinctively, in [1], mobile agents are proposed for MCC for application partitioning between cloud and mobile devices. However, the agents migrate only on the cloud layer.

Mobile devices, as an opportunistic computing platform, rely on infrastructureless spontaneous wire-

Table 1

Challenges in current IoT edge computing

Interoperability	Standardized communication interfaces and protocols are needed to integrate heterogeneous components. Resource sharing originates from the infrastructure-side and is based on utilization of the same virtualization technology.
Energy efficiency	Data upload energy efficiency with resource-constrained end devices is largely not addressed.
Latencies	Data round-trip to the cloud should be avoided. Real-time applications benefit from minimal latencies, where data upload may become a bottleneck.
Scalability	Applications ideally utilize locally dynamically available resources as a shared infrastructure, where availability is not guaranteed. Application component mobility is considered at the edge layer.
Offloading	Centrally controlled offloading introduces increased management load. Static offloading does not consider runtime environment characteristics. Dynamic offloading requires context awareness.

less connections that are not reliable and suffer from low bandwidth and high latencies [12]. The set of participating devices on such a platform can't be predetermined in advance, may not be available at the same time nor know each other. Therefore, application execution is not guaranteed and latencies need to be tolerated. These issues can be controlled with increased application management load, where context- and location-awareness provides knowledge of application-specific resource availability. The integrating technology for participating devices is for example peer-to-peer networks (P2P) [15] that operate without fixed infrastructure in a dynamic topology. Challenges in opportunistic computing include resource availability and the discovery of resources and services. The benefits are that opportunistic deployments are highly scalable and coverage can be extended to areas without fixed infrastructure. However, incentives for opportunistic participation are needed.

Table 1 presents a summary of the main challenges in IoT edge computing, as discussed in this Section.

3. Edge computing with mobile agents

Generally in IoT, distributing the centralized vertical control, partitioning application execution, enabling horizontal interoperability and handling resource dynamicity pave way for software agents and MAS in IoT [8,48]. Software agents operate well on open and dynamic systems, where a priori information availability is not known, a number of participating components is large and new functionality needs to be introduced at runtime [16,17,76]. Resource-aware MAS provides building blocks for smart operation of system components regardless of their placement and functionality, where in large-scale a combination of different technologies are integrated into an IoT platform, e.g. as in hybrid intelligence systems [76]. Operation of such systems is difficult to optimize at design time [46,76]. In IoT, these different technologies can complement each other in different layers of the system architecture, e.g. for data analysis applications.

Mobile agents, as a code mobility paradigm, are capable of controlling their own execution and migration between networked components to execute their encapsulated programs. Already two decades ago, mobile agents were anticipated to bring intelligence and provide benefits in distributed systems [4,26,32]. Mobile agents virtualize the application execution into distributed and loosely coupled components that migrate to relocate application execution with regard to selected application-specific metrics. With mobile agents, it is possible to inject application-specific tasks, such as data processing at the data source, into the system at runtime. Mobile agents provide their results as local services for other system components, including other agents as a MAS. Mobile agents encapsulate device heterogeneity with a common interface for dynamic interoperability. As software agents, mobile agents' operations are autonomous and asynchronous and they have the capabilities to observe their environment and react to changes dynamically and adapt their behavior to the changes. Once injected into the system, there is no need for continuous connection between an owner and the agent, where the owner does not need to control agent operations anymore. With these capabilities, This mobility increases robustness and fault tolerance in distributed application execution.

In this work, the motivation for mobile agent technology in IoT edge computing is to enable autonomous distributed application execution with the opportunistic resources at the device layer. With in-network data processing, mobile agents can address the energy efficiency of the host devices, and simultaneously optimize resource usage in a MAS with regard to local network conditions. Here, mobile agents also implement a mist computing platform [55] by bringing agent-based awareness, interactions, adaptivity and proactivity into the distributed execution of the edge application. Moreover, with mobile agents the execution involves only the devices that directly contribute their data. Mobile agents are considered a complementary technology for the existing edge computing platforms that bring the agent capabilities into the end device operation, based on their capabilities. As discussed later, the standardized means of heterogeneous system component interoperability are provided with Web-integrated mobile agent software framework.

When mobile agents are included in edge application design and implementation, the following agent capabilities are enabled in edge application execution:

Mobile code. Mobile agents relocate, both vertically and horizontally, the application execution load. Agents share their tasks, data and results for horizontally distributed application execution.

Decentralized operation. Autonomy and asynchrony enable mobile agents to execute their tasks without centralized control, based on their individual decision-making capabilities, and without continuous connectivity to the edge. Agents can interact with other system components as required in their tasks.

Adaptive operation. Mobile agents are able to observe and react to local resource availability and dynamic network conditions. With migration, the agents' task can be reactively relocated to continue application execution in other devices, e.g. with better network connectivity.

Reducing network load. Communication energy can be saved when only relevant data, processed by agents at the data source, is transmitted in the system. As an example, agent's tasks may include data filtering, event detection and aggregation.

Lightweight migration. In comparison with VM migration, mobile agents are significantly smaller software components to transfer. Minimally, only the task state is transferred.

Security and privacy. Mobile agents are executed in encapsulated execution environments, sandboxes or VM's, which has full control of the agents' operations and interactions. Mobile agents contribute to privacy by sharing only the application-specific data.

In IoT, hardware limitations of resource-constrained devices introduce challenges for their utilization as mobile agent hosts. This issue can be addressed with optimized agent execution environments and interaction protocols, e.g [34,35,36,37,38] that propose an IoT software framework for mobile agents based on standardized Web technologies. The framework is based on standardized IETF Constrained RESTful Environments (CoRE) framework[1] that provides Representational State Transfer (REST) architectural principles for agent-based application design and Constrained Application Protocol[2] (CoAP) for communications. In addition, the work in [34,38] provides agent platforms for resource-constrained stationary and mobile IoT devices. A key technology in the framework realization is embedded Web services [65] that integrate resource-constrained embedded devices, e.g. with 8-bit microcontrollers (MCU), into the IoT.

Architectures and use cases for IoT systems, that utilize software agents, have been presented in [2, 3,7,16,17,19,23,24,27,29,33,47,51,61,64,71]. As presented, an agent-based IoT architecture is multilayered, where the agents operate on different roles on each layer for the management, control, coordination and organization of system components and to connect components into the system. See the Figure 1b for illustration. Software agents represent and provide reasoning and interaction capabilities for things in [30,50].

Related to the utilized mobile agent software framework [34], recent mobile agent solutions based on the Web technologies include [49,70]. Agents are created as HTML documents, but lack autonomous migration and state transfer capabilities. However, a benefit is that HTML documents also provide means to control and visualize the agent. Existing mobile agent frameworks and platforms are mainly based on Java, such as the JADE mobile agent framework [5], and rely on standardized service-oriented Web technologies. The framework in [34] is currently the only REST-complying mobile agent software framework.

4. Use cases for mobile agents in edge computing

In this section, use cases for mobile agents in IoT edge computing are presented, based on the traditional use cases discussed in the literature, e.g. [32, 26]. Comparison of edge computing platforms without agents with mobile agent based application execution is shown in Table 2. As discussed, mobile agents are considered as a complementary technology to extend edge application execution into the resource-

---

[1] `https://datatracker.ietf.org/wg/core/charter/`

[2] `https://tools.ietf.org/html/rfc7252`

Table 2

Edge computing vs. mobile agent based distributed application execution.

Edge computing platforms	**Mobile agents**
Computation and data offloading	Local resource optimization for application execution in end devices
Centrally coordinated vertical interactions	Decentralized autonomous operation with both horizontal and vertical interactions
VMs do not interact with the environment	Agents observe their environment and react to changes autonomously
VM migration centrally controlled in the infrastructure, significant size reductions needed	Lightweight autonomous migration
High bandwidth and network availability requirements	Adapts interactions to local network characteristics
VMs isolated for security with access rights management	Agent execution and interactions controlled by agent platform

constrained end devices. A software framework that follows standardized Web technologies provides methods for interoperability with existing IoT systems.

4.1. Enabling interoperability

IoT deployments are typically based on a heterogeneous set of devices and other application components that operate with a set of different communication technologies. To maximize interoperability, standardized communication protocols and interfaces are required. Here, Web technologies introduce the benefit of standardized integration into existing Internet.

Web-based interoperability requires that IoT devices are simultaneously both clients and servers for each other in a flat system architecture. This is resource-consuming for the constrained devices [65]. Such devices typically also lack capabilities to operate a full-fledged Web server that complies with full HTTP protocol specifications. Moreover, the interactions of such devices are typically short-lived and asynchronous over lossy networks, where the devices are optimized for low power consumption [65]. A solution to these challenges has been proposed with the CoRE framework and CoAP. CoAP is designed with low protocol overhead and efficient payload encoding. The architectural model of CoAP is based on REST and it has largely the same semantics as HTTP.

In comparison with the existing edge computing solutions, generally the application-specific VMs are self-contained and lack interaction capabilities. For MCC, sharing data locally with additional devices extends the application execution capabilities, but the devices have limited energy availability or network connectivity [15]. In fog computing, nodes are progressively deployed to the infrastructure and try to proactively predict information demand to reduce cloud access. However, the nodes do not communicate directly with each other, but only with nodes at the above tier.

Mobile agents are inherently heterogeneous [32], as their architecture and operations are communication technology independent. Mobile agents expose the resources of their hosts, determine how the devices interact and how resources are shared. A traditional use case for mobile agents has been to provide new functionality dynamically into the host devices, for example to introduce a new communication protocol. When a host device is a gateway, interoperability is facilitated beyond the current Internet to resources in disparate non-IP networks. In the Section 5, it is presented how the CoRE framework is extended to include mobile agents in IoT systems in standardized way to build Web-integrated distributed applications.

4.2. Scalability

In IoT, the centralized system management solutions reach their bounds in large-scale deployments, where it is difficult for a centralized mechanism to adapt to environmental changes in real-time across the layers. In IoT, no single component has the capabilities to handle the complete real-time information about the whole system operational state. Fluctuating resource availability and communication latencies reduce the system performance and quality of service. Fog computing addresses scalability by deploying a large number of fog nodes into the infrastructure[25]. These fog deployments are expensive to maintain, but the approach is justified when the aim is increased spatial coverage. MEC supports scalability by performing management actions at the base stations in the edge.

Mobile agents have the benefit of autonomous decision-making based on local information, e.g. user mobility, that enables to efficiently operate on opportunistic networks, discovering and utilizing resources through migration and cooperation in a MAS. Loose coupling supports different ways for resource utilization. With the migration capability, mobile agents transfer application components asynchronously, which improves scalability as connections between components are not needed until results are available. In addition, mobile agents can clone themselves to distribute task execution load and extend application coverage, for example to provide on-demand services in location.

Furthermore, a mobile agent based MAS, executed horizontally at the end devices, distributes the application execution and communication load. A MAS provides a platform for cooperation and collaboration based on agent interactions protocols and sharing of results. Through the decision-making capabilities and interactions of individual agents, MAS adapts its behavior in relation to changes system environment, resource availability, application requirements or selected global measure, such as energy efficiency [40]. MAS makes it possible to logically structure resources for applications, despite of the physical structure of the system or network. Furthermore, mobile agents as a MAS adaptively execute multiple applications concurrently in the same set of devices.

4.3. Reducing communication load and latencies

Generally in IoT edge computing, data processing is offloaded to the infrastructure side at the edge. This reduces the computational load in the participating end devices, but increases the communication energy consumption. For example, VM migration can consume significant resources [62]. For the core network, data processing at the edge reduces large-scale data transmission volume. Fog architectures facilitate hierarchies between edge nodes and the cloud, where each tier has increased capabilities for data processing [6]. However, a challenge is how to to accommodate different workloads in the nodes to optimize system behavior [74]. The central coordination mechanisms in edge computing platforms create communication overhead, which an agent-based operation can improve, e.g. [1]. However, the coordination itself may not address application-specific data transmission overheads in the device layer. Application execution load is related to the amount of information transmitted back to the end device for the application execution.

Both reducing network load and communication latencies with in-network data processing are well-known benefits of mobile agents [32,26]. In general, large volumes of data can be energy efficiently processed by agents at the data producing devices and later aggregated to avoid communication overhead in the layers. Similarly like VMs, mobile agents are designed for application-specific data processing tasks with different granularity of operation. Each agent in its task execution is capable of reacting and adapting to changes in the system, i.e. network connectivity and local resource availability with regard to the application requirements. As an example, simple decision-making regarding transmitted data size in comparison with agent migration and execution overhead already reduces the device energy consumption [9,40]. With mobile agents, both top-down and bottom-up application distribution models are possible. Examples of the bottom-up approach include autonomous operation of user-specific tasks in a dynamic set of IoT system components [39]. Users can interact locally with the environment through their mobile agents without enforced data round-trip to the cloud, which increases real-time responsiveness.

4.4. Network connectivity

Due to the opportunistic nature of IoT networks, the availability of resources and services can't be guaranteed. End devices join and part systems any time due to connection issues and network bandwidth may become insufficient due to high utilization. Edge device deployments are commonly location dependent. Fog addresses networking issues with the dense deployment of edge nodes that provide high resource availability, bandwidth and LAN connections. Fog nodes may operate without LAN connectivity, which is required for VM migration.

Mobile agents have the benefit of autonomous local operation, even in an isolated network segments without edge connectivity, where the connected system components still can continue to utilize the results of agents' operations. In unfavorable network conditions, mobile agents adapt their data processing to handle different network characteristics, e.g. less data is transmitted with low bandwidth or an agent can migrate to devices with better connectivity. Asynchronous operation and migration increases robustness and fault tolerance in application execution.

4.5. Data privacy

Generally, data storage in the cloud requires management and may compromise privacy as data ownership is inherently changed. A particular concern is maintaining data privacy for applications that collect personal data about the users. Fog computing relies on public-key based authentication on the infrastructure, but data ownership is an issue [74].

Mobile agents indirectly address privacy as data is processed at the source. Moreover, agents can negotiate with the application at hand considering the privacy constraints set by the user, e.g. in mobile crowdsensing [40]. At best, mobile agents expose only a set of data features based on a consent, where the data ownership never changes.

5. Enabling mobile agents for edge computing

In this section, the mobile agent software framework in [34,35,37,40] is presented for IoT edge computing. The framework is based on embedded Web services in the CoRE framework, where CoAP is utilized as a communication protocol. The framework follows the principles of resource-oriented architectures [60]. Generally, REST is found more feasible for ad hoc integration scenarios with lightweight message formats and message semantics than service-oriented architecture based solutions [52]. Stateless CoAP based on UDP is found more energy efficient than TCP based HTTP for resource-constrained embedded devices [10,65,73].

Generally, everything that has value for the applications in the system is exposed as resources and accessed with a RESTful uniform interface [34]. This consideration enables to create Web applications or services seamlessly within the CoRE framework services. The following resources of interest are identified for the mobile agent framework: (1) IoT end devices as agent platforms and their components, such as sensors, and generated data, (2) infrastructure components, such as edge devices and external Web services, and (3) mobile agents and their task results.

HTTP methods realize CRUD (create, read, update and delete) functionality in the mobile agent framework. With REST, the semantics of HTTP and CoAP methods and their response codes are universally known. URIs identify the target resources. These components are then combined to form an HTTP or CoAP request, of which semantics are universally known by a system component or an agent. The method GET retrieves a resource representations, e.g. sensor data or the agent state. The method POST enables multiple functionality: (1) to create a new resource, e.g. through agent migration, and (2) to control and actuate resources, e.g. a physical sensor in a device. The method DELETE removes resources from the framework. CoAP provides the same methods with slightly different semantics [65]. Now, with this RESTful unified interface, the system component interactions are enabled with standardized Web interfaces. Additionally, seamless access to agent resources is provided as with any other Web resource and agent-based interactions with any system resource are enabled through the same interface.

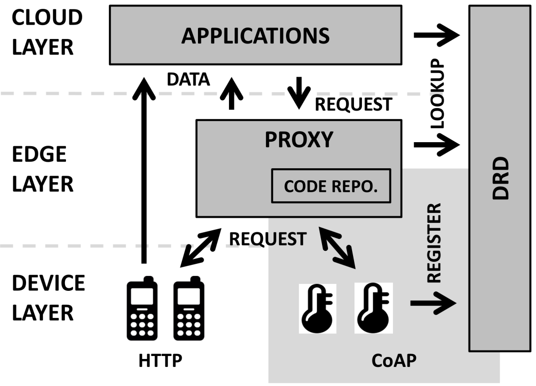


Fig. 2. IoT mobile agent software framework.

See Figure 2 for the illustration of the framework architecture and required software components. At the device layer, mobile and stationary devices that communicate with either HTTP or CoAP, can be integrated into the framework through the distributed resource directory (DRD) [41] that provides resource discovery and runs in a P2P network. Devices register their resources into the DRD to join the system, update their resource descriptions when changes occur and finally unregister resources when leaving the system. When a mobile agent is hosted in a device, it is registered to the DRD as a resource for the device and unregistered after migration away from the device. The DRD interface is described in detail in [41]. A proxy component at the edge layer integrates non-IP devices, such as WSN deployments, into the framework, enabling bidirectional interactions with the exposed resources in the non-IP based network[34]. In the framework, mobile agents are transmitted as CoAP messages or as JavaScript Object Notation (JSON) payloads in HTTP requests, as described in detail in [34]. Lastly, at the cloud layer, applications can perform runtime resource lookups through the DRD, as in [40] and interconnect disparate resources through the proxy.

Table 3

Generic mobile agent architecture that complies with the REST principles

Agent	Agent resource name (URI)
Code	Agent task code and language identifier URL of the remote code resource
Resource	**Local:** Resources in the host devices and targets for migration **Remote:** Remote resources that are retrieved in each iteration of the task **Static:** Remote resources that retrieved once during the agent lifetime
State	Task results that are exposed as the agent state Local variables used in the task execution
Metadata	Agent execution metrics and other metadata

5.1. Mobile agent architecture

The mobile agent architecture [34] utilized within the framework is shown Table 3. The underlying idea is that agents utilize its listed resources as the representation of its operational environment. For example, an agent reads the sensor values in a device and actuates its components or decides to migrate to another device. This enables simple reactive behavior for the mobile agent, which can be executed in an agent platform in a constrained IoT device by taking into account its hardware capabilities. Main features of the architecture are described below. Details can be found in [34,37].

An agent name is the given resource name, e.g. an URI in REST. The URI is also registered into the DRD to provide a unique identifier to address and locate the agent in the system. The architecture enables to implement the agent program, i.e. its task code, in several programming languages that target specific platforms. The program, or its reference can be included in the agent with a specific language identifier, e.g. “python”. Additional benefit of this approach is that through the agent URI and identifier, the agent task code can be retrieved remotely. This way, the architecture supports agents in heterogeneous platforms. Moreover, this allows to minimizing the agent size as a CoAP message.

The agent utilizes three types of system resources. “Local” resources are the resources that the hosting devices expose. These resources also define the migration targets for the agents. “Remote” resources are external to the current host, i.e. hosted in other devices or external Web services. These resources are retrieved in each iteration of the agent task execution. “Static” resources are remote resources, which value does not change, e.g. historical data from a Web service. These resources need to be retrieved only once during the agent lifetime. The different resource types can be modified by the agent in runtime that enables adaptive autonomous operation. For example, when a host device disconnects its URI can be removed. By describing the resources through URIs, loose coupling and dynamic binding are facilitated in a dynamic IoT system environment. For runtime lookup, the resource URIs can be resolved through the DRD.

The state segment exposes the agent task results, i.e. agent program outputs, to the system as a resource that represents the agent. Now, other system components request the agent results through its URI, where the contents of the state segment are returned. The optional metadata segment includes information related to the agent execution, such as execution metrics or historical traces of the agent migration.

Mobile agent platforms for resource-constrained IoT devices, such as 8-bit MCUs (namely ATmega microcontrollers) and Android smartphones, have been developed in [34,40]. The platforms include an HTTP or CoAP server to provide the uniform interface for interactions and to access host device resources. The embedded device platforms handle mobile agents as CoAP messages, where the complete mapping of the agent architecture into a CoAP message is presented in [34]. Generally, each element in the agent architecture has its corresponding CoAP message option. The agent state is included in the message as a payload. In the embedded platform, the agent programs are precompiled into the native machine language of the device due to limited capabilities to handle programs written in general programming languages such as the C programming languages. The Android platform for mobile devices runs agent programs written in either JavaScript or Python that are executed with a 3rd party scripting engine. Agents migrate as JSON data structures in the payload of HTTP requests.

6. Evaluation

A well-known use case for mobile agents is in-network data processing that contributes towards less communication energy consumption by reducing the amount of transmitted data. For example with WSN, less data means less time the transceiver needs to spend transmitting, which gives more time to keep the transceiver in sleep mode [56]. Moreover, in [67], agents can be used to control the sleep cycle of a WSN node. The energy consumption of data producing end devices is also a concern in IoT applications in general, therefore these two methods are adopted in this work to evaluate a mobile agent-based edge application. The application is designed as a MAS based on mobile agents, where a set of mobile agents operate on stationary WSN nodes and an other set of mobile agents operate on smartphones that additionally operate as Internet gateways. The application is illustrated in Figure 3 that additionally shows the required system and agent framework components, i.e. the DRD and a Web application that injects the mobile agents into the system and receives the results of agents' tasks.

The evaluation was conducted by first collecting data of energy consumption in the WSN node and smartphone without and with mobile agents. WSN node energy consumption data was collected in real-world experiments with a Monsoon PowerMonitor device [3]. Smartphone energy consumption data is based on results in [40,59]. Second, based on the collected data, large-scale simulations are conducted to demonstrate the effects of mobile agent-based edge application execution in comparison with straighforward data upload to the edge layer from the same system setup.

As the WSN nodes, Arduino Mega2560 boards are used as mobile agent platforms that communicate with CoAP using XBee Series 2 radios in API mode. This setup was selected to demonstrate the integration of resource-constrained devices communicating with non-IP protocols into an IoT system. Although this setup can be considered unoptimal for the nodes, as discussed later, it corresponds to a real-world IoT scenario with a set of different devices having different hardware components. Communication power consumption parameters were collected with the node uploading data items of different sizes into the edge layer and executing a single mobile agent. Each test run for two minutes, where the results show the average values of each test run. For uploading, two upload intervals where tested: 1s and 10s. Latencies of data transmission varied from 100ms to 1s with different payloads. For larger playloads the XBee radios require multiple frame transmission to send all the data.

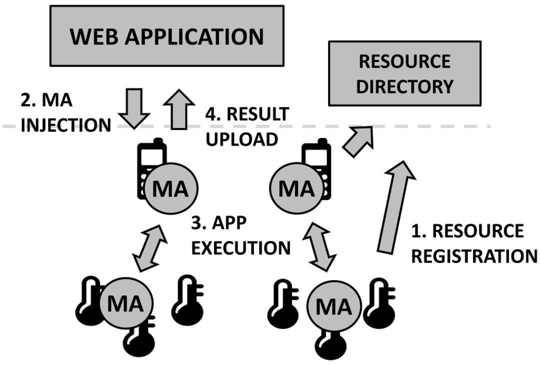


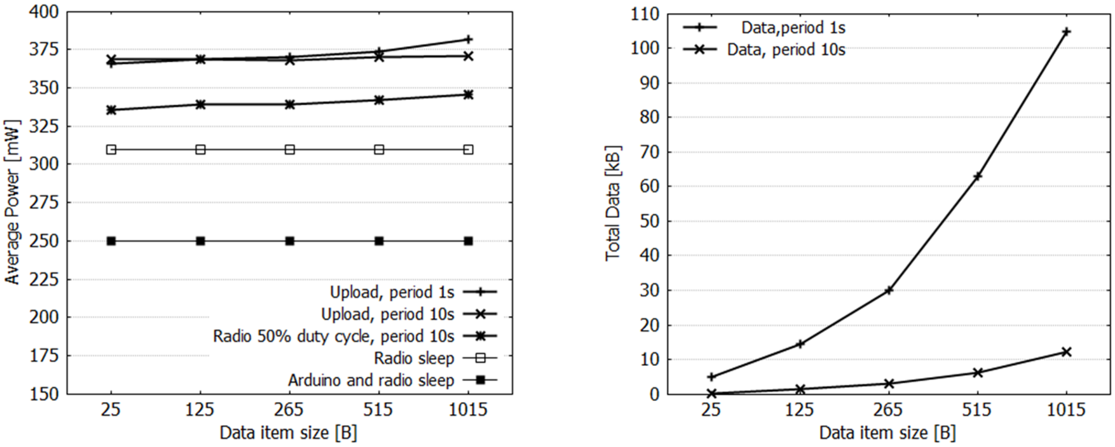
Fig. 3. Evaluated edge application based on mobile agents (MA).

The energy consumption results are shown in Figures 4a and 4b. A significant factor in energy consumption was the high idle power of the Arduino board, about 250mW and including XBee transceiver 310mW. The maximum measured power consumption was 384mW. Between the selected message sizes, from 25 to 1015 bytes, the power consumption is reduced about 15% when the device idle energy consumption is reduced from the measured values. With 50% duty cycling the transceiver, the communication energy is correspondingly reduced about 50%. This result suggests that duty cycling can be significantly beneficial for energy consumption as suggested in [67]. However, the setup with XBee transceivers does not allow smaller duty cycle than 50%, as reconnecting Xbee transceivers to the network takes about 5s after a sleep period. As shown in Figure 4c, mobile agent migration, agent program execution and one remote request for its state introduced overhead (1-2%) to the total device power consumption. With regard to communication energy consumption solely, this overhead was about 7% and with a remote request about 10% of the total communication energy consumption.

The smartphone (Samsung Galaxy S3, Android 4.1) power consumption parameters are based on experiments conducted in [40] and as described in [59]. The results are shown in Figure 5. The mobile devices run the Android agent platform and communicate with HTTP atop Wi-Fi [34]. For the simulation, the mobile devices are equipped with additional XBee radios, enabling operation as Internet gateway for the WSN.

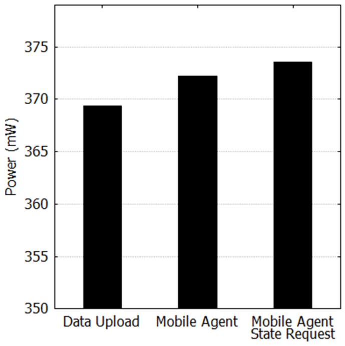
---

[3] `https://www.msoon.com/`



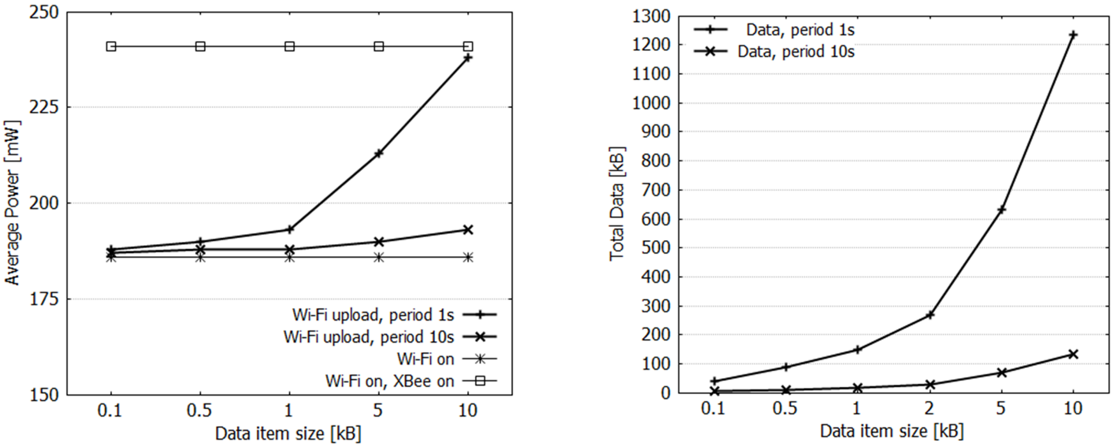
(a) Data upload energy consumption

(b) Amount of uploaded data



(c) Mobile agent execution overhead

Fig. 4. WSN node energy consumption parameters.



(a) Energy consumption parameters

(b) Amount of uploaded data

Fig. 5. Mobile device energy consumption parameters.

Table 4

Simulation parameters

Global	
Time	600 rounds, one round is one second
Mobile devices	
Number of nodes	20
Communication interface	Wi-Fi, unlimited bandwidth, uniform coverage, single-hop distance
Power consumption	Wi-Fi : 164mW, XBee: 55mW Agent EE and sensor: 22mW
Data transmission power	5mW per KB
WSN	
Number of nodes	100
Communication interface	XBee Series 2, unlimited bandwidth, uniform coverage, single-hop distance
Power consumption	Device, sensor and radio: 310mW Agent EE and sensor: 1mW
Data transmission power	2-6mW per message, CoAP message size 25-1015B

Table 5

Simulated edge computing scenarios

No.	**Description**
E1.1	All devices upload all raw data to the edge
E1.2	Same as above, WSN node transceivers 50% duty cycling
E2	Mobile agent based data processing in all devices, data size reduced to 10%
E3.1	Mobile agent based data processing in all devices, data size reduced to 10%, WSN node transceivers 50% duty cycling
E3.2	Same as above, but data size not reduced
E4	Mobile agent based data processing in all devices, processed data in WSN nodes are retrieved by agents in mobile devices

The combined data contains power consumption overhead of the agent platform execution and HTTP and CoAP communications for the hosted mobile agents. The XBee radio increases the device power consumption about 25%, with the power consumption data used from the WSN experiment. For data upload with data item sizes up to 1kB, there are no significant increase in Wi-Fi communication power consumption.

6.1. Simulation results

With the simulation parameters collected in the previous Section, a set of simulations were conducted. NetLogo [72] MAS modeling environment was used to run the simulations. The utilized simulation parameters are shown in Table 4.

Table 5 lists the simulated scenarios that were designed each with 100 WSN nodes and 20 mobile devices. The first scenarios E1.1 and E1.2 are baseline scenarios, where WSN nodes and mobile devices upload all raw to the edge, first without duty cycling and then with 50% duty cycling. This sleep cycle is based on the XBee transceiver sleeping 5s and then reconnecting 5s, where the data upload interval becomes 10s. Scenario E2 demonstrates the effect of mobile agent based in-network data processing in both types of devices. Scenarios E3.1 and E3.2 demonstrate the mobile agent based in-network data processing and controlling the transceiver duty cycling in a MAS operating on a set of heterogeneous devices. The scenarios are particularly designed for the two extremes of the in-network data processing, as in scenario E3.1 90% of the data is removed and in scenario E3.2 no data is removed. The scenario E3.2 enables to observe the mobile agent execution overhead in the devices. Scenario E4 demonstrates the edge application MAS operation (Figure 3), where the agents in different devices interact with each other. The smartphone mobile agent first retrieves the task results of mobile agents from five WSN nodes. The node agents reduce the transmitted data size from 1015B to 25B. Then the smartphone mobile agent reduces the retrieved data size further 90% and uploads one final data item to the edge. The smartphone mobile agent that is utilized in the simulations is shown in Table 6. The mobile agent task code is written in Python.

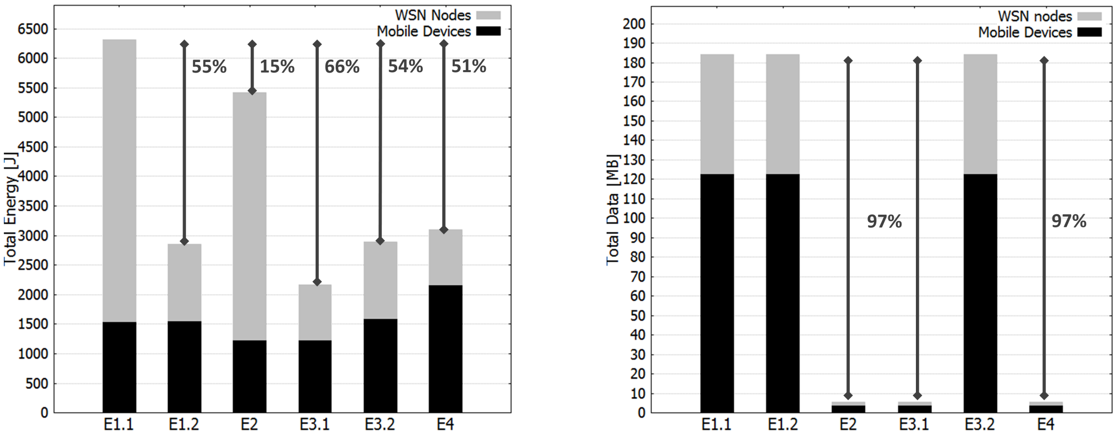
Table 6

The smartphone mobile agent used in evaluation.

Agent URI	light
Code	result = light[0:len(light)/10];

Resource	**Local:** light = coap://node/light;
	Remote: light = coap://remote_node/light;

State	result



(a) Energy consumption (b) Transmitted data

Fig. 6. Simulated application scenarios with the observed energy consumption and transmitted data reduction.

The simulation results, as shown in Figure 6, show that with duty cycling (scenario E1.2), the total application energy consumption can be reduced 55% in comparison with the baseline scenario E1.1. In scenario E2, in-network data processing results in 15% reduction in the energy consumption, even when the total transmitted data size is reduced 97%. The results of scenarios E3.1 and E3.2 indicate that mobile agent operations, i.e. combined data processing and transceiver control, yield the best results with up to 66% reduction in the energy consumption. When comparing scenarios E1.2. and E3.2, it can be observed that the total mobile agent execution overhead was about 1%, where the same amount of data is transmitted. The results of scenario E4 show the overhead of mobile agent interactions when using the external XBee transceiver in the smartphone. With this design, the energy consumption is reduced 51% in comparison with the baseline scenario E1.1. It can also be observed that duty cycling alone (scenario E1.2) is slightly more effective in reducing the total energy consumption while transmitting all data.

7. Discussion

The presented evaluation studied an edge computing application, where mobile agents operated in a MAS on a set heterogeneous resource-constrained IoT devices. This application scenario was selected to demonstrate the energy efficiency of decentralized and autonomous agent-based control of end device operation at the device layer in IoT edge computing. The results indicate that significant reduction in the device energy consumption can be achieved with the presented application design. Although the presented results are application-specific, and many other MAS configurations and IoT device hardware configurations can be found, the evaluated scenario was designed to be unoptimal for the resource-constrained devices and still benefits are shown. An open question in large-scale is the optimal level of agent-based functionality needed in such an application, where the host devices have different hardware capabilities and agents can be introduced in all layers of the system architecture. However, the focus of this paper is to present a method of optimizing the agent operations at the device layer.

For the WSN nodes, the amount of data size reduction achieved with in-network data processing was arbitrarily chosen to demonstrate two extremes: almost all data removed and no reduction at all. The effect of data size reduction by mobile agents to the energy consumption is well-known, e.g for WSN nodes [9,32,56] and recently for mobile devices [40]. The results show that mobile agent-based in-network data processing is a feasible option even for the resource-constrained devices, when significant amounts of data is to be transferred. Both communication energy consumption and the following data transfer time can be reduced. However, mobile agent execution and migration overheads needs to be taken into account, also as shown for mobile devices in [40]. The available network bandwidth affects this result, e.g. with limited bandwidth the benefit of data processing amplifies as the amount of transmitted data can be optimized. Related to the duty cycling, less data means that the transceiver spends less time in sending the data [56]. Mobile agent execution overhead for the resource-constrained WSN nodes was insignificant for a simple data processing task, but the number of agent-based interactions should be optimized in a MAS.

Duty cycling alone can be an efficient method of reducing energy consumption as shown in the results. The time to keep the transceiver on should be minimized, which is a method commonly used with real-world WSN deployments. The operation of such nodes is controlled centrally by a sink node, an application component at the edge or a software agent, e.g. [67]. The method proposed in this paper is that application-specific mobile agents are in control of the participating device operation and not only have access to the device data. This approach has drawbacks, as the interaction possibilities of the nodes are limited while the transceiver is sleeping, but in reality this limitation is widely accepted in WSN when it is expected that the nodes only upload their data to the sink node. This scenario becomes a MAS interaction issue when a number of agents negotiate resource control for their applications.

The non-IP based communications added another source of difficulty into the evaluation application realization and increased energy consumption was observed due to the second radio in mobile devices. Nevertheless, this setup corresponds to a real-world IoT application scenario that requires gateways for Internet connectivity. The XBee protocol imposes severe limitations to the node communications, such as low throughput and small message frame size [54]. The available bandwidth was much less than as presented in the related work on IoT edge computing. As discussed in [54], the maximum measured bandwidth of XBee is 5.4kB/s, thus XBee is suitable technology only for low datarate applications with no strict real-time requirements.

The evaluation results of existing edge computing platforms largely concern application execution latencies, as it is the identified key benefit in IoT edge computing. Partitioning applications into cloudlets is shown to reduce execution latencies and energy consumption in mobile devices [21]. With regard to application design, positioning a large number of small cloudlets at the edge are is better than larger cloudlets in the network devices towards the cloud. However, this increases the system management costs [63]. The communication latencies and VM migration times are reported in the range of tens of seconds up to minutes with different placements of the VM and different available network bandwidths [62,6,21]. With mobile edge clouds, device clusters with distributed datasets and minimal interactions outperform task offloading to cloud with regard to latency [14]. In comparison, results in this paper show that small-sized mobile agents can migrate as a single optimized message. Similar results were obtained in [58], where container based application code was executed on Raspberry PI single-board computers but with execution latencies up to 7 seconds. The DRD was evaluated in [42], where it was shown that resource lookup latencies are below one second.

Overall, mobile agents extend the edge computing platform towards decentralized autonomous and asynchronous application execution, with the additional benefit of energy efficiency that is particularly shown in the operations on IoT device layer. Mobile agents have significantly smaller communication, migration and application execution requirements than VM-based code mobility can achieve. The evaluation considered applications with significant amount of data to be transmitted and the results are supported by previous research [9,56]. Security-wise both VM and mobile agents are remote code that is executed on host computers at risk.

To generalize the presented results, it would be difficult to take into account all the different communication technologies, device hardware configurations, network characteristics and varying application configurations in the context of IoT. Even when an optimization algorithm can be provided, it is often tailored for the particular application and generalization can be difficult due to conflicting sets of parameters. In [46], a theoretical framework is proposed that enable to assess distributed system deployments and find the most appropriate deployment architecture, based on monitoring of multiple operational factors in runtime, e.g. network parameters, quality of service and usage of the provided functionality. In this paper, simplified analysis was conducted based on the energy consumption of the transmitted data, but considering other network parameters, environment characteristics and application requirements remain a future work.

With regard to software agents, a MAS can be designed in a number of ways that each targets the same or different aspects in application execution. In [34,37,38], straightforward metrics is presented that can be used to evaluate both the MAS design and agent architecture from the resource usage point of view. The type of a resource, i.e. local or remote, and its size can be used to approximate the energy consumption of accessing the resource. With this information, it is possible to address different MAS designs, e.g. where the agents should be placed and how the resources are defined in the agent. Moreover, in the presented agent architecture, it is possible to redefine the resource types by the agent, which enables to dynamically optimize the resource usage. Lastly, from the agent-based IoT application design and implementation perspective, its is needed that mobile agent platforms are readily available for a number of common IoT device platforms. The presented mobile agent framework, that utilizes well-known REST principles for the application design, and the mobile agent platforms for well-known resource-constrained device platforms are steps towards this goal.

8. Conclusion

The current edge computing solutions focus on vertical offloading of data or computational tasks from resource-constrained end devices to the resource-rich edge devices. The dynamic nature of IoT calls for the reactive and adaptive behavior of system components, which is facilitated with mobile agents in this work. Adaptivity and self- and context-awareness are also desired features in mist computing platforms. Mobile agents are capable of autonomous operation to distribute data processing tasks into the heterogeneous set of resource-constrained IoT devices. The agent-based sharing of information in a MAS promotes local horizontal execution for applications within the end devices without the need for edge connectivity. In general, agent mobility increases energy efficiency, robustness and fault tolerance in distributed edge application execution.

In this work, the mobile agent based in-network data processing and operational control of data producing devices are shown in real-world to reduce edge application energy consumption significantly for both stationary and mobile resource-constrained IoT devices. Mobile agents in the devices operate as a MAS in the presented unfavourable system setup with additional hardware components. Nevertheless, the amount of transmitted data can be significantly decreased, which contributes towards energy efficiency due to reduced data transmission latencies and increased data utility with local sharing. However, the benefits of in-network data processing are largely application-specific due to different requirements and various MAS setups, but agent-based device resource control has significant benefits. For the IoT system infrastructure layers, the benefits of this approach include less bandwidth needed to transmit the data to the cloud and reduced data processing needs in the backend.

With regard to the edge computing challenges presented in Table 1, mobile agents provide autonomous application execution that adaptively considers local resource availability and network characteristics at the device layer. Local sharing of the data and results facilitates horizontal interoperability, where embedded Web services provide a standardized way for the different system components to interact, including the existing Internet-based services. The presented mobile agent framework is a valuable extension to the IoT edge and mist computing platforms, where the presented results demonstrate benefits for data-intensive IoT applications.

Acknowledgments

The authors would like to thank Mr. Joonas Kataja for the work with the Arduino mobile agent platform.

References

[1] P. Angin, and B. K. Bhargava, An agent-based optimization framework for mobile-cloud computing, *JoWUA*, **4(2)** (2013), 1–17.

[2] R. Aversa, B. Di Martino, M. Rak, and S. Venticinque, Cloud agency: A mobile agent based cloud system, in: *International Conference on Complex, Intelligent and Software Intensive Systems*, IEEE, 2010, pp. 132–137.

[3] I. Ayala, M. Amor, and L. Fuentes, The sol agent platform: Enabling group communication and interoperability of self-configuring agents in the internet of things, *Journal of Ambient Intelligence and Smart Environments*, **7(2)** (2015), 243–269.

[4] P. Bellavista, A. Corradi, and C. Stefanelli, Mobile agent middleware for mobile computing, *Computer*, **34(3)** (2001), 73–81.

[5] F. L. Bellifemine, G. Caire, and D. Greenwood, *Developing Multi-agent Systems with JADE*, Wiley, 2007.

[6] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, Fog computing and its role in the internet of things, in: *1st Edition of the workshop on Mobile cloud computing*, ACM, 2012, pp. 13–16.

[7] F. Bonomi, R. Milito, P. Natarajan, and J. Zhu, Fog computing: A platform for internet of things and analytics, in: *Big Data and Internet of Things: A Roadmap for Smart Environments*, N. Bessis, C. Dobre, eds, Springer, 2014, pp. 169–186.

[8] E. Borgia, The internet of things vision: Key features, applications and open issues, *Computer Communications* **54** (2014), 1–31.

[9] M. Chen, T. Kwon, Y. Yuan, and V. Leung, Mobile agent based wireless sensor networks, *Journal of Computers*, **1(1)** (2006), 14–21.

[10] W. Colitti, K. Steenhaut, N. D. Caro, B. Buta, and V. Dobrota, Evaluation of constrained application protocol for wireless sensor networks, in: *18th IEEE Workshop on Local Metropolitan Area Networks*, IEEE, 2011, pp. 1–6.

[11] Conti, S. Giordano, M. May, and A. Passarella, From opportunistic networks to opportunistic computing, *IEEE Communications Magazine*, **48(9)** (2010), 126–139.

[12] M. Conti, C. Boldrini, S. S. Kanhere, E. Mingozzi, E. Pagani, P. M. Ruiz, et al., From manet to people-centric networking: Milestones and open research challenges, *Computer Communications*, **71** (2015), 1–21.

[13] H. T. Dinh, C. Lee, D. Niyato, and P. Wang, A survey of mobile cloud computing: architecture, applications, and approaches, *Wireless communications and mobile computing*, **13(18)** (2013), 1587–1611.

[14] U. Drolia, R. Martins, J. Tan, A. Chheda, M. Sanghavi, R. Gandhi, et al., The case for mobile edge-clouds, in: *10th International Conference on Ubiquitous Intelligence and Computing and Autonomic and Trusted Computing*, IEEE, 2013, pp. 209–215.

[15] N. Fernando, S. W. Loke, and W. Rahayu, Mobile cloud computing: A survey, *Future generation computer systems*, **29(1)** (2013), 84–106.

[16] G. Fortino, A. Guerrieri, and W. Russo, Agent-oriented smart objects development, in: *16th International Conference on Computer Supported Cooperative Work in Design*, IEEE, 2012, pp. 907–912.

[17] G. Fortino and W. Russo, Towards a cloud-assisted and agent-oriented architecture for the internet of things, in: *13th International Conference of the Italian Association for Artificial Intelligence*, 2013, pp. 60–65.

[18] G. C. Fox, S. Kamburugamuve, and R. D. Hartman, Architecture and measured characteristics of a cloud based internet of things, in: *International Conference on Collaboration Technologies and Systems*, IEEE, 2012, pp. 6–12.

[19] W. W. Godfrey, S. S. Jha, and S. B. Nair, On a mobile agent framework for an internet of things, in: *International Conference on Communication Systems and Network Technologies*, IEEE, 2013, pp. 345–350.

[20] B. Guo, D. Zhang, Z. Wang, Z. Yu, and X. Zhou, Opportunistic IoT: Exploring the harmonious interaction between human and the internet of things, *Journal of Network and Computer Applications*, **36(6)** (2013), 1531–1539.

[21] K. Ha, Y. Abe, Z. Chen, W. Hu, B. Amos, P. Pillai, et al., Adaptive VM handoff across cloudlets, Technical Report CMU-CS-15-113, CMU School of Computer Science, 2015.

[22] K. Habak, M. Ammar, K. A. Harras, and E. Zegura, Femto clouds: Leveraging mobile devices to provide cloud service at the edge, in: *8th International Conference on Cloud Computing*, IEEE, 2015, pp. 9–16.

[23] M. G. Hafez and M. S. Elgamel, Agent-based cloud computing: A survey, in: *4th International Conference on Future Internet of Things and Cloud*, IEEE, 2016, pp. 285–292.

[24] M. E. P. Hernandez and S. Reiff-Marganiec, Towards a software framework for the autonomous internet of things, in: *4th International Conference on Future Internet of Things and Cloud*, IEEE, 2016, pp. 220–227.

[25] K. Hong, D. Lillethun, U. Ramachandran, B. Ottenwalder, and B. Koldehofe, Mobile fog: A programming model for large-scale applications on the internet of things, in: *2nd ACM SIGCOMM workshop on Mobile cloud computing*, 2013, pp. 15–20.

[26] A. R. Hurson, E. Jean, M. Ongtang, X. Gao, Y. Jiao, and T. E. Potok, Recent advances in mobile agent-oriented applications, in: *Mobile Intelligence*, L. T. Yang, A. B. Waluyo, J. Ma, L. Tan, and B. Srinivasan, eds, Wiley, 2010, pp. 106–139.

[27] 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*, IEEE, 2016, pp. 88–93.

[28] V. Karagiannis, P. Chatzimisios, F. Vazquez-Gallego, and J. Alonso-Zarate, A survey on application layer protocols for the internet of things, *Transaction on IoT and Cloud Computing*, **3(1)** (2015), 11–17.

[29] A. Katasonov, O. Kaykova, O. Khriyenko, S. Nikitin, and V. Y. Terziyan, Smart semantic middleware for the internet of things, in: *ICINCO-ICSO*, 2008, pp. 169–178.

[30] E. Kazanavicius, V. Kazanavicius, and L. Ostaseviciute, Agent-based framework for embedded systems development in smart environments, in: *International Conference on Information Technologies*, FISITA, 2009, pp. 194–200.

[31] K. Kumar, J. Liu, Y.-H. Lu, and B. Bhargava, A survey of computation offloading for mobile systems, *Mobile Networks and Applications*, **18(1)** (2013), 129–140.

[32] D. B. Lange and M. Oshima, Seven good reasons for mobile agents, *Communications of the ACM*, **42(3)** (1999), 88–89.

[33] P. Leong and L. Lu, Multiagent web for the internet of things, in: *International Conference on Information Science and Applications*, IEEE, 2014, pp. 1–4.

[34] 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*, 2013, pp. 14–21.

[35] T. Leppänen, J. Alvarez Lacasia, A. Ramalingam, M. Liu, E. Harjula, P. Närhi, et al., Interoperable mobile agents in heterogeneous wireless sensor networks, in: *11th ACM Conference on Embedded Networked Sensor Systems*, 2013, pp. 6–7.

[36] T. Leppänen, and J. Riekki, A lightweight agent-based architecture for the internet of things, in: *IEICE workshop on Smart Sensing, Wireless Communications, and Human Probes*, IEICE, 2013, pp. 2–4.

[37] T. Leppänen, J. Riekki, M. Liu, E. Harjula, and T. Ojala, Mobile Agents-Based Smart Objects for the Internet of Things, in: *Internet of Things Based on Smart Objects. Internet of Things (Technology, Communications and Computing)*, G. Fortino, and P. Trunfio, eds, Springer, 2014, pp. 29–48.

[38] T. Leppänen, and J. Riekki, Moving computation to the edges of iot, *Internet of Things Magazine Finland*, **1** (2015), 28–31.

[39] T. Leppänen, I. Sanchez Milara, J. Yang, J. Kataja, and J. Riekki, Enabling user-centered interactions in the internet of things, in: *IEEE International Conference on Systems, Man, and Cybernetics*, IEEE, 2016, pp. 1537–1543.

[40] T. Leppänen, J. Alvarez Lacasia, Y. Tobe, K. Sezaki, and J. Riekki, Mobile crowdsensing with mobile agents, *Autonomous Agents and Multi-Agent Systems*, **31(1)** (2017), 1–35.

[41] M. Liu, T. Leppänen, E. Harjula, Z. Ou, A. Ramalingam, M. Ylianttila, et al., Distributed resource directory architecture in machine-to-machine communications, in: *International Conference on Wireless and Mobile Computing, Networking and Communications*, IEEE, 2013, pp. 319–324.

[42] M. Liu, T. Leppänen, E. Harjula, Z. Ou, M. Ylianttila, and T. Ojala, Distributed resource discovery in the machine-to-machine applications, in: *10th IEEE International Conference on Mobile Ad-hoc and Sensor Systems*, IEEE, 2013, pp. 3–4.

[43] M. Liyanage, C. Chang, and S. Srirama, mepaas: Mobile-embedded platform as a service for distributing fog computing to edge nodes. in: *17th International Conference on Parallel and Distributed Computing, Applications and Technologies*, IEEE, 2016, pp. 73-80.

[44] P. Garcia Lopez, A. Montresor, D. Epema, A. Datta, T. Higashino, A. Iamnitchi, et al., Edge-centric computing: Vision and challenges, *ACM SIGCOMM Computer Communication Review*, **45(5)** (2015), 37–42.

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

[46] S. Malek, N. Medvidovic, and M. Mikic-Rakic. An extensible framework for improving a distributed software system's deployment architecture. *IEEE Transactions on Software Engineering*, **38(1)** (2012), 73–100.

[47] B. Manate, T.-F. Fortis, and V. Negru, Optimizing cloud resources allocation for an internet of things architecture, *Scalable Computing: Practice and Experience*, **15(4)** (2015), 345–355.

[48] D. Miorandi, S. Sicari, F. De Pellegrini, and I. Chlamtac, Internet of Things: Vision, applications and research challenges, *Ad Hoc Networks*, **10** (2012), 1497–1516.

[49] D. Mitrovic, M. Ivanovic, Z. Budimac, and M. Vidakovic, Radigost: Interoperable Web-based multi-agent platform, *Journal of Systems and Software*, **90(1)** (2014), 167–178.

[50] A. M. Mzahm, M. S. Ahmad, and A. Y. Tang, Agents of things (aot): An intelligent operational concept of the internet of things (iot), in: *13th International Conference on Intelligent Systems Design and Applications*, IEEE, 2013, pp. 159–164.

[51] N. M. do Nascimento and C. J. P. de Lucena, Fiot: An agent-based framework for self-adaptive and self-organizing applications based on the internet of things, *Information Sciences*, **378** (2017), 161–176.

[52] C. Pautasso, O. Zimmermann, and F. Leymann, Restful web services vs. big web services: making the right architectural decision, in: *17th International Conference on World Wide Web*, ACM, 2008, pp. 805–814.

[53] M. Patel, B. Naughton, C. Chan, N. Sprecher, S. Abeta, A. Neal et al., Mobile-edge computing introductory technical white paper, Mobile-edge Computing industry initiative, 2014.

[54] R. Piyare and S.-R. Lee, Performance analysis of xbee zb module based wireless sensor networks, *International Journal of Scientific & Engineering Research*, **4(4)** (2013), 1615–1621.

[55] J. Preden, K. Tammemae, A. Jantsch, M. Leier, A. Riid, and E. Calis, The benefits of self-awareness and attention in fog and mist computing. *Computer*, **48(7)** (2015), 37–45.

[56] H. Qi, S. Iyengar, and K. Chakrabarty, Multiresolution data integration using mobile agents in distributed sensor networks, *IEEE Transactions on Systems, Man, and Cybernetics, Part C Applications and Reviews*, **31(3)** (2001), 383–391.

[57] M. H. ur Rehman, C. Sun, T. Y. Wah, A. Iqbal, and P. P. Jayaraman, Opportunistic computation offloading in mobile edge cloud computing environments, in: *International Conference on Mobile Data Management*, IEEE, 2016, pp. 208–213.

[58] T. Renner, M. Meldau, and A. Kliem, Towards container-based resource management for the internet of things, in: *International Conference on Software Networking*, IEEE, 2016, pp. 1–5.

[59] A. Rice and S. Hay, Measuring mobile phone energy consumption for 802.11 wireless networking, *Pervasive and Mobile Computing*, **6(6)** (2010), 593–606.

[60] L. Richardson and S. Ruby, *RESTful Web Services*, O'Reilly, 2007.

[61] T. Sanchez Lopez, A. Brintrup, M.-A. Isenberg, and J. Mansfeld, Resource management in the internet of things: Clustering, synchronisation and software agents, in: *Architecting the internet of things*, D. Uckelmann, M. Harrison and F. Michahelles, eds, Springer, 2011, pp. 159–193.

[62] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, The case for VM-based cloudlets in mobile computing, *IEEE pervasive Computing*, **8(4)** (2009).

[63] M. Satyanarayanan, P. Simoens, Y. Xiao, P. Pillai, Z. Chen, K. Ha, et al., Edge analytics in the internet of things, *IEEE Pervasive Computing*, **14(2)** (2015), 24–31.

[64] M. Schatten, J. Seva, and I. Tomicic, A roadmap for scalable agent organizations in the internet of everything, *Journal of Systems and Software*, **115** (2016), 31–41.

[65] Z. Shelby, Embedded Web services, *IEEE Wireless Communications*, **17(6)** (2010), 52–57.

[66] W. Shi and S. Dustdar, The promise of edge computing, *Computer*, **49(5)** (2016), 78–81.

[67] R. Tynan, D. Marsh, D. O'Kane, and G.M.P. O'Hare, Intelligent agents for wireless sensor networks, in: *4th International joint conference on autonomous agents and multiagent systems*, ACM, 2005, pp. 1179–1180.

[68] D. Uckelmann, M. Harrison, and F. Michahelles, An architectural approach towards the future Internet of Things, in: *Architecting the internet of things*, D. Uckelmann, M. Harrison and F. Michahelles, eds, Springer, 2011, pp. 1–24.

[69] M. Vögler, J. M. Schleicher, C. Inzinger, and S. Dustdar, A scalable framework for provisioning large-scale iot deployments, *ACM Transactions on Internet Technology*, **16(2)** (2016), article 11.

[70] J.-P. Voutilainen, A.-L. Mattila, K. Systä, and T. Mikkonen, Html5-based mobile agents for web-of-things, *Informatica*, **40(1)** (2016), 43–51.

[71] 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*,

36(6) (2013), 1684–1695.

[72] U. Wilensky, Netlogo, Techincal Report, Center for Connected Learning and Computer-Based Modeling, Northwestern University, 1999.

[73] D. Yazar and A. Dunkels, EfïňĄcient application integration in ip-based sensor networks, in: *1st ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings*, ACM, 2009, pp. 43–48.

[74] S. Yi, C. Li, and Q. Li, A survey of fog computing: Concepts, applications and issues, in: *Workshop on Mobile Big Data*, ACM, 2015, pp. 37–42.

[75] W. Zhang, S. Tan, F. Xia, X. Chen, Z. Li, Q. Lu, et al., A survey on decision making for task migration in mobile cloud environments, *Personal and Ubiquitous Computing*, **20(3)** (2016), 295–309.

[76] Z. Zhang, and C. Zhang, Building agent-based hybrid intelligent systems: A case study. *Web Intelligence and Agent Systems*, **5(3)** (2007), 255–271.