Chapter

Swarm Computing: The Emergence of a Collective Artificial Intelligence at the Edge of the Internet

Laisa Costa de Biase, Geovane Fedrecheski, Pablo Calcina-Ccori, Roseli Lopes and Marcelo Zuffo

Abstract

Billions of devices are interacting in a growing global network, currently designated as the Internet of Things (IoT). In this scenario, embedded computers with sensors and actuators are widespread in all sorts of smart things, transforming the way we live. The complexity produced by the enormous amount of devices expected in the future IoT leads to new challenges. Furthermore, current IoT architectures are highly cloud-centric and do not take advantage of all its potential. To overcome these issues, we propose Swarm computing as the emergence of a collective artificial intelligence out of a decentralized and organic network of cooperating devices. The major contribution of this article is to provide the reader with a comprehensive vision of the key aspects of the Swarm Computing paradigm. In addition, this article addresses technical solutions, related projects, and the Swarm Computing challenges that the research community is called to contribute with.

Keywords: swarm computing, internet of things, collective intelligence, distributed computing, ubiquitous computing, computer architecture

1. Introduction

In this chapter, we present the Swarm Computing paradigm. It is expected to be the evolution of the Internet of Things, the edge of edge computing, in which devices gain protagonism over cloud, leveraging a true open computing resource-sharing infrastructure.

The Internet of Things (IoT) is a global infrastructure that connects objects from the physical world and provides advanced services [1]. Computers with sensors and actuators are embedded in everyday objects, making them smart. In traditional deployments, these devices retrieve huge data that generates knowledge with big-data analysis. The global Internet of Things (IoT) market was valued at US$ 201 billion in 2022 and is projected to reach US$ 410 billion by 2026 [2]. IoT advances are positively impacting diverse areas such as home, manufacturing, medicine, and urban life.

Due to their growing computing power, embedded devices are gaining higher responsibilities that, in the first IoT generation, were exclusively performed in the cloud. This paradigm, called Edge Computing, breaks the centralized model and reduces the amount of data sent to the cloud for processing and the response time for critical applications. Furthermore, the edge computing paradigm increases the security and privacy of data by avoiding the need for transmission and the overall reduction of costs of the cloud infrastructure [3, 4]. Among the main challenges in edge computing is the programmability of the heterogeneous platforms running in the edge, with different computing capabilities, operating systems, and programming languages. The naming of the enormous number of available things is also an unsolved challenge due to the various network protocols available and the lack of standardization. Privacy and security are among the most important challenges in terms of authentication and authorization for the use of devices. Finally, a common model for data abstraction, a comprehensive service management system, and metrics for optimization are among other relevant challenges.

We compare this expected evolution of the IoT to the changes that happened on the Internet. First, the Web was a provider of static content. Then, with Web 2.0, it became participative and democratic; that is, common people assumed a leading role, publishing their content and even trading goods. Finally, Web 3.0 emerged, favoring the culture of sharing, in which individuals exchange resources for mutual benefit (trading an English class for a text review, for example). Similarly, we expect that with the increasing computing power, smart objects will play a major role in the future IoT, by increasing their participation in data processing and establishing cooperative relations. Cooperation will occur through the spontaneous and autonomous organization of devices to solve problems collaboratively, resulting in the emergence of collective artificial intelligence. We call this network of autonomous cooperating devices Swarm computing.

The Swarm is a bio-inspired framework of autonomous smart objects. In parallel with swarms of bees, with specialized bees contributing to a common goal; the Swarm is composed of specialized devices whose interaction solves a common problem. Swarm computing behaves like an organism and shows an organized behavior that results in an emergent collective intelligence. Analogously to specialized honeybees, and the Swarm consists of heterogeneous devices, which might range from high-power processing servers to low-power wearable devices.

Human participation in the Swarm network is also of crucial importance, since the Swarm is at the service of humans. It shall be able to identify and meet real-life needs. Being able to "extract" needs – either inferring from the context or through direct human-interaction. It is especially relevant since the Internet of Things of the future will be composed of thousands of devices per human, new interaction mechanisms avoiding individual configuration and direct control are demanded.

To illustrate an application of the Swarm in our daily lives, we present the following example. Penny, Alice's cat, goes every day for a walk through the neighborhood and is back by dinner time. Once she did not come back, and Alice was really worried. How could the Swarm be used to help Alice to find her cat? Alice could simply ask for Penny, and opportunistically, the Swarm would gather sensors to capture Alice's request. Any microphone could be used, from Alice's smartphone or from a baby monitor. This request could be done at a high semantic level, asking directly "Where is Penny?", and local or remote resources could be used to realize that Penny is a cat and which her attributes are (appearance, weight, etc.). Alice home-network devices could be used to help find Penny: surveillance cameras, a baby monitor, and motion sensors that are usually used to automatically turn the lights on. However, it is not enough if Penny is out of the house. This way, the Swarm allows for actively gathering resources from other networks, such as the surveillance cameras from the street. The images could be processed by a computer vision service available in the cloud. Each usage of a third-party resource has associated retribution in the Swarm economy that will allow other owners to have priority to their own requests. The Swarm could also be applicable in other scenarios where resource sharing is key, such as looking for a missing person or object, in agribusiness with drone sharing, and in smart buildings by enabling automatic sharing of processing nodes, projectors, displays, windows and doors automation systems, HVAC systems, sensors, and many other devices.

The contribution of this paper is to provide a comprehensive description of the key aspects of the Swarm vision, focusing on its principles and challenges. We also review prior work and present an application example. Additionally, we present our initial architecture and implementation.

This is relevant for providing a software infrastructure that allows an open and global network of devices that can collaborate sharing resources with each other. Current Internet of Things implementations are restricted to niches and proprietary networks. An example is the home-network standards, such as UPnP which connects devices in the household environments, connecting TVs, computers, and smartphones, among others. On the other hand, the Swarm paradigm aims to provide technologies and strategies to connect billions of heterogeneous devices in a flexible and scalable way.

2. Other initiatives

While the IoT term refers to a large network of connected devices, some of those ideas were anticipated with other names. Already in 1994, the term ubiquitous computing (UC) [5] was used to describe a vision where computing devices are widespread at all scales throughout everyday life. While vastly present in human activities, this computation is almost invisible and does not draw attention to itself. This paradigm is part of the IoT current vision, embedding intelligence in everyday objects.

In 2001, Autonomic Computing (AC) [6] was proposed as a solution to the capital challenge that complexity represented for the future of computing systems, whose management goes beyond human capabilities. The AC approach deals with complexity reproducing the human autonomic nervous system, consisting of self-configuration, self-optimization, self-healing, and self-protection. This need for self-managed systems gains renewed importance in the context of the increasing complexity of the IoT, although this previously proposed solution was too complex to deploy.

The Swarm was inspired by a work that made an analogy between biological and digital ecosystems, in 2007 [7]. It briefly mentions the term swarm to refer to a set of computing agents interacting and engaged in solving a common problem. Four aspects were highlighted for digital ecosystems: interaction and engagement, balance, domain clustered and loosely coupled individuals, and self-organization. In addition, semantic Web technologies were recommended for information exchange, attribute modeling, and integrity check.

The Social IoT approach [8] looks forward to advancing the current IoT vision presenting an alternative to the producer-consumer paradigm, by collaborating with other counterparts toward a common goal, as the Swarm does. This approach, however, proposes the implementation of social-like capabilities to the objects, enhancing trust between "friends" objects. The inter-object relationship is related to the human social network.

In the last decade, the cloud has emerged as the principal responsible for data storage and processing that is provided and consumed by personal computers and mobile devices. The IoT adds a communication layer between the physical and logical (cyber) world. Initially, in the IoT first generation, devices are used as data providers. In the second IoT generation, devices are empowered, distributing the processing. This new paradigm is exploited in the Fog Computing [9] and Edge Computing [3] approaches. These definitions use the general idea of performing processing closer to the devices. In Fog Computing, network equipment, and PCs execute part of the processing. In the Edge computing approach, devices do it by themselves. The Swarm vision is aligned with the Edge Computing paradigm, as it aims to make devices less dependent on the cloud and favors a more decentralized IoT.

The Swarm term, applied in the IoT context, was first mentioned by Jan Rabaey in 2008, as a sensory swarm, connecting trillions of sensing and actuating devices connected through a single abstraction platform at the edge of the cloud [10, 11]. Subsequent work led to a more concrete definition of the Swarm [12, 13], proposing an initial architecture. They also outlined a common framework for devices to communicate and share resources, called Swarm OS [14], that was later developed.

3. Major challenges

We identify five main challenges to achieve the Swarm realization: *communication and cooperation* among devices, *human-interaction* with the network, support for *resource-constrained devices*, *security,* and the inherent *network complexity*.

3.1 Communication

Communication is the first step to establishing cooperation among devices. The Swarm is a highly heterogeneous environment that does not pose restrictions to its participants, configuring an open system. Traditional standards enable communication in open systems, but they generate niches (e.g., digital home, industry, etc.), limiting system evolution. This flexibility issue comes from static documents that generate products in which it is necessary to run firmware updates to accommodate standards reviews. Open systems pose security risks as well. Since, in IoT, many participants are resource-constrained, traditional security solutions are not applicable. Furthermore, at the Edge of the Internet, there is a fragmentation of IP and non-IP protocols, IoT technologies such as 6LoWPAN, Bluetooth, LoRa, and Sigfox, which causes fundamental interoperability problems. In summary, this challenge is related to how to achieve an open system with flexibility, broad scope, and security. Also, how to perform interoperability among Swarm agents, considering their different computing capabilities and network protocols.

3.2 Cooperation

In the Swarm context, cooperation consists of sharing resources among participants to accomplish high-level tasks. The result of this cooperation will manifest as a collective intelligence that emerges from the Swarm. Aspects that have to be dealt with are the discovery of resources spread around the globe and the autonomous and spontaneous orchestration of the shared resources, that is, how to use and "embed" this resource inside the consumer's business logic in execution time (in opposition to programming time) without human intervention. This cooperation must be balanced, satisfying objectives from individuals as well as from the Swarm participants as a whole. In this context, new concepts arise, such as trust, virtual currency, billing, reputation, and a full virtual economic system. This phenomenon represents a perfect parallel to a growing trend in the current world economy called the sharing economy, which favors the sharing of physical resources over the acquisition of new ones. Examples are Uber and Airbnb. The resource sharing in the Swarm represents its digital equivalent. A consequence is consumption reduction and better global use of resources.

3.3 Resource-constrained devices

Resource-constrained devices are an essential segment of the current IoT participants, whose operation has a strong focus on energy saving and miniaturization. The energy consumption of these wireless technologies has a significant impact on battery life. The device consumes energy to collect data by sensing the environment and processing and communicating the data. Therefore, all system parts, including software and hardware parts, should be considered to optimize energy consumption. Wireless energy harvesting from environmental sources like solar power is one of the best ways to supply energy for many sensors from the hardware perspective. It is also essential to consider the way how the IoT devices communicate to improve the efficiency of existing power sources in the device considering the data rate of IoT terminals and distances. These specifications should be considered in a communication system to efficiently use power and spectrum. Energy-efficient devices are imperative to make the existent applications greener and more environmentally friendly. On the other hand, to achieve a decentralized swarm of devices, increasing computer activity is expected to be delegated to the edge of the network, allow more efficient use of resources, provide highly responsive services, and enforce a privacy policy. This conflict represents a significant challenge to implementing the Swarm network.

3.4 Human-interaction

Although the Swarm consists of a nonbiological network, human beings play a central role in the Swarm, as it assists humans to interact with the physical world and with other humans. The complexity of the Swarm, involving billions of devices, makes it unmanageable by a person. Thus, a significant challenge is to develop interfaces with high-level semantics and proactive behavior. An interface with high-level semantics abstracts the network's actual resources allowing humans to focus on the intended result instead of focusing on the resources management and, in the process, to make this goal to be achieved. Additionally, the Swarm has the potential to explore an opportunistic gathering of available interfaces exploring the diversity of devices capable of interacting with people. Proactive behavior emerges from past interactions, extracting policies automatically. For example, an agent can infer that, for a given person, comfort takes priority over power saving, which will leverage a policy where the air conditioning will work almost continuously. A person that prioritizes power saving will have a home where the temperature oscillation is more tolerated so the air conditioner will often be off. Additionally, known preferences may be shared among agents to support this proactivity. For example, personal ambient temperature preference may be shared with occupants in a room with an HVAC (Heating, Ventilation, and Air Conditioning) system to maximize comfort.

3.5 Security

A large number of devices collecting and sharing data will open questions about what kinds of data are being shared, who has the right to perform this sharing, and how this data can be protected. Since the devices composing the Swarm will communicate openly across different networks, they will be exposed to a diverse array of cyber threats. Therefore, guaranteeing the privacy and trustworthiness of data in transit will represent a significant challenge. This is aggravated by the fact that, while many devices in the IoT are resource-constrained, cryptographic algorithms, such as those based on asymmetric cryptography, require significant processing and memory resources. Furthermore, since the messages will likely traverse different kinds of networks, the protocols for message security must be able to cope with such a heterogeneous environment. While a possible solution lies in protecting messages at the application layer, the currently accepted protocol for Internet security (Transport Layer Security—TLS) works at the transport layer. Another important challenge concerning security is device identification. Network identifiers such as MAC and IP addresses are easily spoofed, and more secure approaches such as certificates and cloud accounts depend on centralized architectures [15]. This raises questions and challenges regarding the need for secure and decentralized identification solutions for IoT devices [16]. Finally, while Swarm devices are expected to collaborate, they must do so in a controlled manner to prevent security issues. What is needed is a high-level authorization mechanism that device owners can use to specify the collaboration rules. Challenges in this respect arise from the global scale and decentralized nature of Swarm computing.

3.6 Complexity

The Swarm has characteristics of a complex network: autonomy, connectivity, self-organization, emergent behavior, and co-evolution with an environment, and billions of autonomous and interconnected computing devices interact. Among a diversity of resources, there are sensors and actuators. Sensors make the system subject to uncertain and unpredictable events from the physical world, and actuators will impact the physical environment, creating a symbiotic ecosystem. The opportunistic organization of these devices will provide a robust self-organization structure that is perceived as intelligent since it is capable of evolving over time.

4. Swarm principles

The Swarm computing approach can be better understood by examining its two underlying principles: resource sharing and autonomy. These principles can be subdivided into more specific aspects that guide the Swarm evolution, as shown in **Figure 1**.

Autonomy confers devices to the ability to share resources without needing manual input from humans (self-organization). These automatic interactions are not programmed into the system beforehand but emerge naturally during execution time (spontaneity). As a result, the participants may encounter optimized paths and

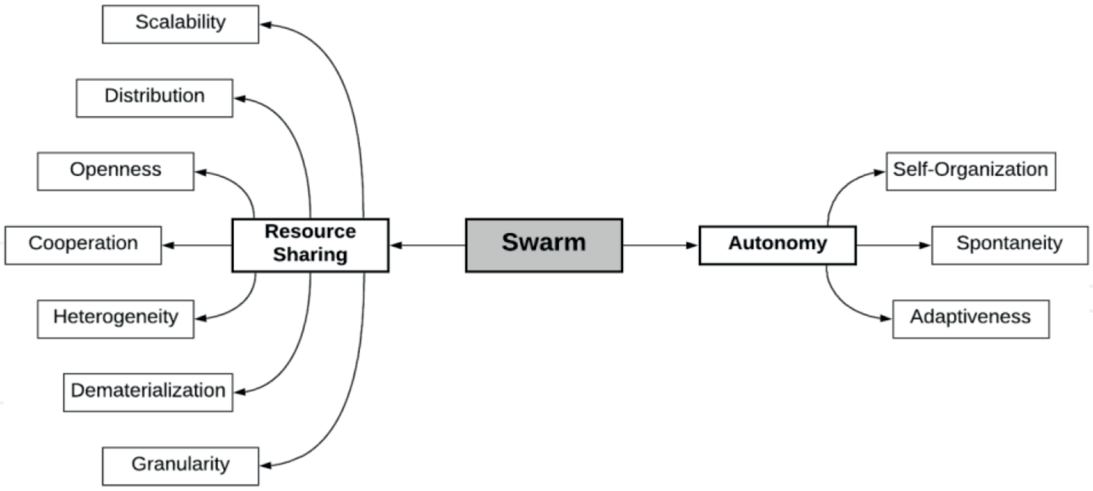

Figure 1.
The Swarm principles and their subsumed aspects.

risky interactions over time, from which they must learn to unexpected situations. Autonomy can be further subdivided into the following aspects:

- Self-organization: when a task or problem needs to be solved, the Swarm devices must be able to organize themselves to achieve the desired results. The devices should perform this behavior, that is, not requiring step-by-step commands from a human operator or a centralized entity.

- Spontaneity: Swarm devices should not necessarily only work in a predefined way. Rather, they should be able to infer changes in their context and react properly to new and unexpected situations.

- Adaptiveness: since each Swarm device can compute only a fragment of the global context, certain operations may be carried out in suboptimal ways, especially when confronted for the first time. Therefore it is important that devices can learn from previous experiences and adapt to perform better in future scenarios.

Resource sharing allows a set of entities (distribution) to provide and consume resources from each other for mutual benefit (cooperation). These entities may have different amounts (heterogeneity) of resources to share, use open and interoperable protocols (openness), and cooperate independently of their physical structure and location (dematerialized). Finally, resources can be composed at different levels (granular), and their scope may range from local to global (scalable). The following aspects are key to enabling resource sharing in the Swarm:

- Scalability: refers to the capacity to support a great number of participants, achieving billions, with global and local communication capability, comprising machine-to-machine and over-the-Internet interoperation;

- Distribution: the participant devices act without a central coordinator;

- Openness: the conditions to connect and interoperate with the Swarm are open, widely accessible, allowing for new products and services to be created by anyone. Despite, is that, security, and privacy risks, it makes systems more powerful (efficient, through sharing of resources), more resilient (by the use of redundant resources through dynamic reconfiguration), and more capable, enabling applications that have not been realized yet;

- Cooperation: allow a set of entities to provide and consume resources from each other, exploring synergies and generating economy;
- Heterogeneity: the Swarm participants are diverse, with different functions, complexity, processing and communication capabilities, operating systems, etc.;
- Dematerialization: participant resources are exposed and shared, making the physical boundaries to lose importance;
- Granularity: the cooperating services could be understood as a set of reusable components that can be organized to compose bigger ones, putting together a hierarchy of components.

The Swarm principles act as guidelines for the research and development of the Swarm computing paradigm and its applications. For example, imagine a set of cameras and smart doors that cooperate. When a person approaches the system, the cameras will tell the doors whether to open or not. Or in an emergency scenario, a car passing by accident can automatically share a danger alert with surrounding devices. Both examples work without human intervention (autonomy) and create value through cooperation (resource sharing). By combining these two principles in a cohesive model/platform/abstraction, the Swarm creates a new era for the Internet of Things.

5. Architecture

In this section, we propose our architecture for the realization of the Swarm Computing vision.

5.1 Processing balance

In the Swarm, devices are used not just for sensing and actuation, but also for hosting applications and business logic. This leads to architectural differences between the Swarm and other IoT solutions. Although the Swarm shares principles with the Edge Computing vision, many Edge Computing architectures still run applications on remote servers in the cloud, while edge devices are limited to preprocessing tasks.

Most IoT architectures are organized into three layers: perception/actuation, transport, and application. The perception/actuation layer is responsible for gathering information from the physical world and acting on it. The perception layer comprehends the devices with embedded electronics, particularly sensors and actuators. The transport layer connects devices to the cloud; it includes gateways and the network infrastructure (i.e., the Internet). The application layer includes IoT middleware and applications [17]. Other works split the application layer into three: Business Intelligence, Application, and Middleware [18]. These layered models evidence that despite the generic definitions of the IoT term, their architectures and solutions are centralized in servers in the cloud. In general, the application runs on a server in the cloud and devices become information providers.

Current Edge Computing efforts promote the participation of sensors and gateways in data processing and analysis [18]. The layered architecture in Edge Computing puts preprocessing between the layers of perception and transport. This preprocessing at the edge includes monitoring, storage, and security. While this approach aims to reduce the strong centralization of cloud computing, it still relies on the cloud and centralized applications.

The Swarm meets the goal of decentralization better, empowering devices in the autonomous composition of new services. In the Swarm, devices seek resources from other devices to cooperate and achieve complex goals. In addition, applications run on the devices themselves, thus creating a large and scalable network of distributed processing.

5.2 Distribution of resources

Cooperation in the Swarm is achieved through resource sharing. We use a microservices approach to expose device resources. Microservices are an evolution of traditional service-oriented architectures, to provide better scalability, performance, loose coupling, functional independence, reusability, resilience, and cost [19].

In the past decade, Service-Oriented Architecture (SOA) became a popular paradigm for integrating distributed services across organizations. SOA is a software design pattern based on the dynamic selection of services to other applications. The most serious drawback of SOA is its centralized integration, based on a service bus. Microservices move intelligence to the endpoints, eliminating this centralized integration. In addition, each service is either independent or broken into smaller independent services.

We propose two kinds of microservices in the Swarm: platform and application. The platform microservices consist of common microservices that the Swarm participants use to support the interaction among them. An example of a platform microservice is the discovery service, which helps to locate resources in the Swarm. In addition, swarm devices offer application microservices to share their resources, such as a service to read a temperature sensor.

While an application in the Swarm may run in isolation, the true potential of the Swarm lies in service composition. When a service uses other services, it creates a graph of resources. Devices that own those resources form groups of interacting participants in the Swarm.

5.3 Swarm OS

IoT frameworks usually provide a software module in the cloud, called Broker, that helps service consumers to access service providers. Our Swarm OS is a Broker enhancement. So the IoT Broker is a Swarm Broker or, finally, a Swarm OS.

MQTT [20] is a popular IoT framework based on the publish/subscribe protocol. The MQTT Broker is a central entity that manages data publications and subscriptions. The IoT Broker Generic Enabler is a component of the Fiware middleware [21] that interfaces with devices providing publish/subscribe managements and associations between device-level and things-level descriptions. The Fiware middleware resides in a server in the cloud.

The Swarm Broker is conceptually different. It is a software agent installed on each device, turning it into an "insect," that is, a member of the Swarm. The Swarm Broker is responsible for providing the platform microservices that provide service discovery; a distributed registry of services; access control to resources; protocol binding; policy management; service-level agreements; semantic mediation; and optimization.

As the Swarm is an open and heterogeneous environment, we propose a minimum Swarm Broker that includes a core set of the platform microservices to be installed on every device [22]. Advanced features are provided by more complex Brokers that run on more capable devices. Proxy servers can serve legacy or less capable devices that are unable to run the minimum Broker.

Device interaction in the Swarm happens in three phases: registration, cooperation initiation, and interaction support. In the Registration phase, services such as cameras, temperature sensors, and smart doors register themselves in a Swarm Broker, becoming available to be shared within the Swarm. The Cooperation Initiation phase starts when a Swarm participant demands a resource and is concluded by a service-level agreement establishment with the best available resource at the moment. Finally, the interaction support phase occurs when cooperation is already established and running. A direct link between the service consumer and the provider is established, while the Swarm Broker acts as a helper by providing contract maintenance, protocol adaptation, optimization, authentication, and access control.

The Registration phase consists of the check-in of a service provider to the Swarm. As each device has at least a minimum Broker inside, service providers will communicate with their own local Swarm Broker (i.e., in the same device), registering its resources. A service description is registered, containing all the necessary information to verify the resource's suitability and then access the service, such as the functional description, quality of service, and required retribution. Once the resource is registered, it is available to the Swarm for sharing. The registration is done in a local registry of the Broker and may be sent to another Broker that would act as a service directory. This strategy allows for an opportunistic registry that can be centralized, totally or partially distributed accordingly to the capabilities of the available Brokers at the moment.

Flexibility is achieved by using semantics in the service descriptions [23]. Semantics uses ontologies to define terms and their relationship, allowing to build standardized protocols for niches that can be expanded or interconnected by linking various ontologies together. For example, the services used in home networking, such as a television and a baby monitor, may connect to services from the city, such as a public surveillance camera pointing to the street. If a company association that deals with home network defines an ontology in which the service "camera" and "display" are defined; other association of companies that deals with smart cities may create an ontology with their own terms, such as "city_cameras," "semaphore," "surveillance_camera" and "biometrics_camera." If an application uses the television inside a house to display images from a street camera, the Swarm may automatically do this mapping linking both ontologies to finding the equivalences.

The Cooperation Initiation phase is started when a Swarm participant, a Service Consumer, searches for a given resource on its local Swarm Broker, that is, the Broker that is running in its same device. The local Swarm Broker that has received this request will then communicate with other Brokers in the Swarm to identify the best suitable resource and then negotiate the establishment of a service-level agreement.

The discovery has four stages. First, the Broker searches in its local registry. Second, the Broker sends a query to other Brokers that act as third-party registries.

Third, a bootstrap mechanism defines the address of the Brokers to forward the request. This mechanism is complemented by a history of past interactions and a heuristic search for Broker' addresses. Third, the request to locate the service is forwarded to the local network, using direct communication, such as multicast. The fourth mechanism uses a mediation service that expands the discovery request to equivalent services, based on a functionality taxonomy, instead of the exact matching of a keyword or hash code.

This discovery returns a set of services that meet the given functional requirements. The resulting set of services is evaluated from the quality of service point of view, verifying policies about access control, priorities, and retribution mechanisms. A contract is then established with the best service provider available, setting a service-level agreement. To achieve fairness, the Swarm supports a microeconomy model with credits exchange for sharing a resource (payments) and a reputation system. Finally, the Swarm Broker returns this contract and the service information to the participant that requested the service.

The discovery process returns directly the operation that the consumer will use to access the service, allowing direct communication between service consumer and provider; since the operation is not predefined and is discovered just as the provider, it is called as automatic execution.

Interaction Support occurs when cooperation is already established and running; thus, there is a direct link between service consumer and provider, and the Swarm Broker may provide some support services: contract maintenance, protocol adaptation, general optimization, authentication, and access control. Currently, just access control and CoAP-HTTP binding are supported. The Swarm Brokers store and process access control policies and rules, having the role of supporting decisions of their local service providers and participating in access control enforcement [22]. Proxies are platform services that are ideally transparent and bidirectional, creating a unified virtual network that merges distinct physical networks. This type of proxy intercepts messages and redirects responses to pass through them [23].

6. Implementation approach

So far, we have presented the Swarm challenges, principles, and architecture, which include the Swarm Broker as a key enabler. This section discusses the implementation aspects of the Swarm OS, including an overview of the technologies employed and the services it implements. As the Swarm is a heterogeneous platform, two versions of the Broker have been developed: a Minimum Broker and a Full Broker. We start by briefly describing the former and then proceed to give more details on the latter.

The minimum Swarm Broker implements only two essential services that enable the device to be discovered by other Swarm devices: Registry and Discovery. It was implemented using Lua programming language, targeting a node with an ESP8266 microcontroller. This implementation has only 2% of the size and uses 0.02% of memory compared to the full Swarm Broker implementation [24]. While this proof-of-concept can be enhanced, particularly by adding access control enforcement to protect its services, it showed that even highly restricted devices could participate in the Swarm network while relying on more powerful nodes for more complex services.

The full Swarm Broker is implemented using the Elixir programming language and has been tested on Linux laptops, servers, and single-board computers. As shown in **Figure 2**, it follows a protocol-agnostic architecture, in which the Broker Core module executes business logic, and is separated from protocol-specific modules, such as Broker HTTP and Multicast. A Broker HTTP Client is also present to help the Swarm Broker call other Swarm Brokers and services independently. As application services may also use the Swarm Broker Client, it is integrated within the Broker as an external library. Modules that implement alternative protocols also exist, such as the Swarm Broker CoAP [25] and its respective Client, as shown in faded colors in **Figure 2**.

The Swarm Broker implements the following platform services:

- **Registry:** responsible for keeping a record of available services in the Swarm. It receives a service description file serialized as JSON-LD via the Broker HTTP interface and saves it into memory. During registration, the Registry will verify whether there is a Semantic Registry Service available, to whom it will forward the service description as well. The Semantic Registry Service enhances the capabilities of the Swarm Broker by allowing service inference during the discovery process.

- **Discovery:** allow the search of services in the Swarm network. It receives a request containing a Query Description and responds with a list of matching services. The Query Description is a JSON-LD file containing parameters that specify the desired services. For example, a query may include "type: camera," to indicate that it is looking for camera services.

- **Distances:** stores the distance between a service and a Bluetooth beacon. It receives periodical updates from services capable of measuring their own distance against beacons. These distances can then be used to refine a Discovery query, for exmaple, search for services that are up to three meters from the TV.

- **Access control:** maintains and enforces access policies among services. It is divided into a Policy Decision Point (PDP), responsible for evaluating policies against requests, and a Policy Administration Point (PAP), responsible for managing policies.

- **Policy sharing:** allows policy sharing among services. The current implementation uses the Discovery service to discover other Swarm devices, and then pulls and pushes the access policies, according to the needs of a policy administrator.

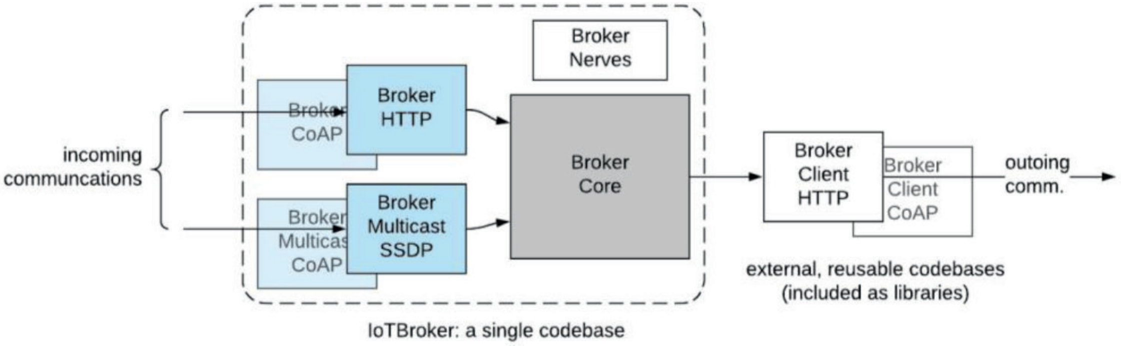

Figure 2.
Swarm Broker support for interaction.

- **Policy management**: provides a GUI for editing access policies. It first connects to the policy sharing services in order to obtain the policies from nearby devices, which are then shown in a Web interface. Next, a user edits and saves the new policies sent to the respective devices through the policy sharing service.

- **Contract:** allows a consumer to establish a service-level agreement with service providers, including the discovery and selection of suitable services, defining usage conditions, paying for it, and evaluating the service. It is one of the most complex Broker services, as it composes the Discovery, Access Control, and Reputation services and implements two blockchain clients. Creating a contract involves eight steps, depicted in **Figure 3**. The first step is to have a registered service provider (1) available to be contracted. Then, a service consumer tries to contract (2) a target service, by sending a query to its Broker, which will use the Discovery service (3) to find services within the Swarm network. After finding a list of services, the best service is selected (4), using price and reputation as the sorting criteria. Then, negotiation (5) takes place, which consists of having the consumer and the provider to accept the proposed contract. If all goes well, a (6) digital currency payment is made through a blockchain. After the payment is confirmed, establishing a service-level agreement (7) will be triggered, causing the service provider to create an access policy allowing the consumer to use its services. Finally, the two Brokers will evaluate each other

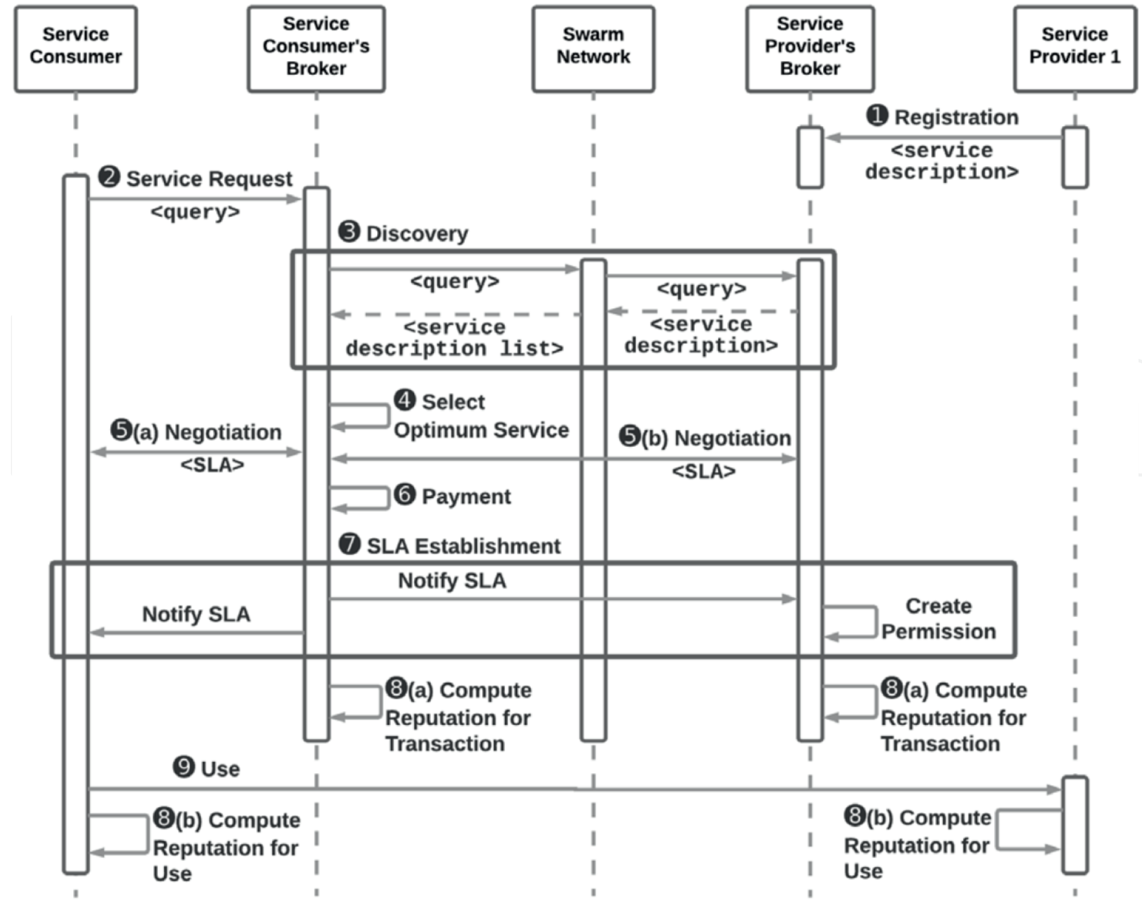

Figure 3.
Swarm broker codebase architecture.

Swarm broker stages	Services involved
Registration	Registry, policy sharing
Cooperation initiation	Discovery, contract, access control
Interaction support	Distances, access control, policy management, reputation

Table 1.
Swarm interaction stages and the main services involved.

and compute a reputation value (8a), the consumer will use the service provider (9) for a limited amount of time, and reputation for service usage will be computed (8b).

- **Reputation:** allows consumers and providers to rate each other. This service receives the reputation computed by service consumers and providers regarding a contract they have established and run. These reputations will be forwarded to a blockchain, which will act as a public ledger for the reputation of all Swarm services.

- **Semantic registry**: in addition to the basic functionality of the Registry module, the Semantic Registry stores the information of available services using a knowledge base that relies on an ontology and has a semantic representation of the services in the Swarm network. As in the Registry module, the Semantic Registry receives a service description based on the JSON-LD format, which natively includes semantic information about the services. Additionally, this module uses an inference engine to expand the original query and bring compatible services according to their semantic service description.

The described Broker services give support to application services, allowing them to be part of the Swarm. **Table 1** relates the three stages of Broker support, described in Section 5, to the services implemented in the Broker.

7. Proof-of-concept

To demonstrate the functionality of the Swarm Broker implementation, we have developed a use case applying the Swarm to a surveillance application, specifically the scenario described in the introduction of this article. To implement the scenario, we use the following services, also shown in **Figure 4**:

- **Personal assistant**: the system frontend, capable of natural language processing to identify commands and parameters. Additionally, the Swarm Assistant reunites information about the person who owns a device group. It might have information that the owner has a pet, a cat called Penny, and even have some pictures of it.

- **Object finder**: a service specialized in finding objects. It discovers and connects Camera Services to Identification Services.

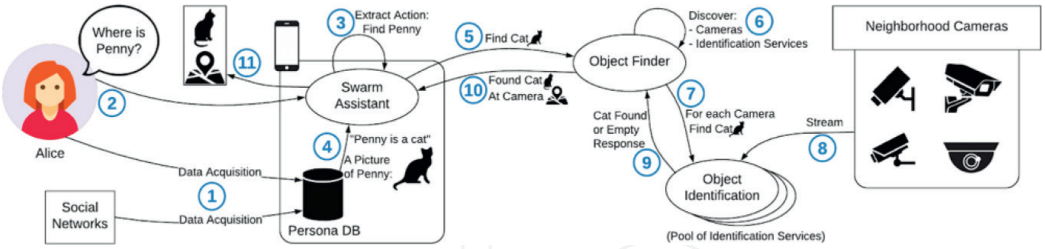
Where is Penny?
Alice
2
11
3 Extract Action: Find Penny
Swarm Assistant
4 "Penny is a cat" A Picture of Penny:
Data Acquisition
1
Data Acquisition
Social Networks
Persona DB
5 Find Cat
10 Found Cat At Camera
Object Finder
Discover: 6 - Cameras - Identification Services
7 For each Camera Find Cat
Cat Found or Empty Response
9
Stream
8
Object Identification
(Pool of Identification Services)
Neighborhood Cameras

Figure 4.
Finding Penny implementation using the Swarm.

- **Camera service**: a service that makes camera frames available to allowed consumers.
- **Identification service**: an image processing service that takes two images as inputs, a picture of a cat and a camera frame, and returns whether the former appears in the latter.

It is important to highlight that, Since the Swarm uses a service-oriented approach, the physical location of devices is not important, except for limiting the scope of the interactions. In our implementation, each service is implemented in a different device to exercise the decentralization principle of the Swarm. Thus, each device has exactly one application service, and one Swarm Broker, that supports the service interactions. The communication flow of this example is described below:

1. **Data acquisition:** In this phase, the Persona Database is pre-populated. The “Persona” is a concept adopted by the Swarm to digitally represent human users. In our scenario, the Persona Database contains data about Alice, such as preferences, friends, and pets. This information may be manually inserted by Alice, or automatically extracted from her social networks, or a combination of both.

2. **Voice command:** issued by Alice, it is processed by the Swarm Assistant, an app that serves as an interface between users and the Swarm.

3. **Extract action:** using a Speech-to-Text service, the assistant obtains the action and other parameters from a voice command. In this case, the action ``Find” and the string “Penny,” an unknown parameter, are extracted.

4. **Parameters qualification:** the assistant consults the Persona Database to get information about the extracted parameters. In this case, it learns that the object that it must Find is a Cat, obtains a picture of the cat, and also gets Alice’s default preferences about the radius of object searches.

5. **Find cat:** the assistant uses the swarm network to discover an Object Finder, a service specialized in finding objects, and then asks the Object Finder to search for a cat, passing the cat’s picture, and a radius relative to Alice’s smartphone position, where the cat must be searched.

6. **Discover cameras and detection services:** based on Alice’s preferences, the Object Finder will find the addresses of available cameras in her neighborhood and one or more Object Identification services.

7. **Identify cat:** for each of the discovered cameras, the Object Finder will call an Object Identification service, passing the address of the camera, the category of the object (i.e., cat), and a picture of the specific object that must be found. If there are less identification services than cameras, the requests will be enqueued and sent when an identification service becomes available.

8. **Stream:** each identification service will gather one camera's stream, and try to detect a cat. If successful, it will compare with the provided cat picture. Finally, if there is a match, the frame with the found cat will be returned, along with the position of the camera that captured the frame.

9. The frame and the location of the found cat are returned to Alice's Swarm Assistant. Now Alice knows where is Penny, her missing cat.

8. Discussion

As pointed out in the challenges section, communication and cooperation constitute the main challenges in the future of IoT. The importance of communication can be observed in two main areas: device-to-device and human-to-device. In the first case, we identify a research opportunity for autonomous intelligent agents to overcome the complexity of the resulting network. Regarding the second case, which involves human-to-device communication, we clearly see the need to deeply explore the understanding of human language, with emphasis on human commands. As the number of devices available to every person grows, a concise way to perform tasks involving several devices is more important. Recent advances in tools such as ChatGPT show the relevance of the convergence between the IoT and Natural Language Processing (NLP).

9. Conclusions

In this chapter, we have presented the Swarm computing vision, a decentralized and self-adaptive approach to overcome the limitations of cloud-centric architecture for the IoT. We presented the principles that have driven the conception of the Swarm and summarized the main challenges to achieving its realization: communication and cooperation of devices, the inclusion of resource-constrained devices, better interfaces for human-interaction, and the complex nature of the network. We also proposed an initial architecture, whose main component is the Broker, a communication mediator that aims to solve the interoperability of devices in the Swarm network. Besides, we listed a selection of technologies that enabled our implementation and described an application example that illustrates the potential of the Swarm network. Our advances in coping with the Swarm challenges can be summarized as follows. We developed four Broker implementations, using different programming languages: C, Lua, Java, and Elixir. We developed a minimum Broker implementation. We adopted open Web semantic technologies that facilitate device communication; we implemented a mechanism of semantic service discovery as a starting point for cooperation; and we implemented a CoAP-HTTP proxy that leverages transparent communication with resource-constrained devices with minimum impact. Additional effort is needed in all fronts of Swarm challenges to concretize the vision.

Acknowledgements

We could not have undertaken this journey without Dr. Jan Rabaey and inspiring work and talks. We would like to express our gratitude to CITI-USP.

This work was partially supported by LSI-Tec; FUNDEP-Rota2030, Stellantis and CEABS; MCTI and BNDES.

Author details

Laisa Costa de Biase*, Geovane Fedrecheski, Pablo Calcina-Ccori, Roseli Lopes and Marcelo Zuffo
Escola Politecnica of the University of Sao Paulo, Sao Paulo, Brazil

*Address all correspondence to: laisa.costa@usp.br

References

[1] International Telecommunication Union. Overview of the Internet of Things. Geneva, Switzerland: International Telecommunication Union; 2012

[2] IoT Analytics. Global IoT market size to grow 19% in 2023—IoT shows resilience despite economic downturn [Internet]. 2023. Available from: https://iot-analytics.com/iot-market-size/

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

[4] Shi W, Pallis G, Xu Z. Edge computing [Scanning the Issue]. Proceedings of the IEEE. 2019;**107**:1474-1781

[5] Weiser M. The computer for the 21st century. Scientific American. 1991;**265**(3):94-105

[6] Abeywickrama DB, Ovaska E. A survey of autonomic computing methods in digital service ecosystems. Service-Oriented Computing and Applications. 2017;**11**(1):1-31

[7] Boley H, Chang E. Digital ecosystems: Principles and semantics. In: 2007 Inaugural IEEE-IES Digital EcoSystems and Technologies Conference, Cairns, QLD, Australia, 2007. pp. 398-403

[8] Atzori L, Iera A, Morabito G. From "smart objects" to "social objects": The next evolutionary step of the internet of things. IEEE Communications Magazine. 2014;**52**(1):97-105

[9] Chiang M, Ha S, Risso F, Zhang T, Chih-Lin I. Clarifying fog computing and networking: 10 questions and answers. IEEE Communications Magazine. 2017;**55**(4):18-20

[10] Maliniak D. Visions Of The Future (Part 1): A Ubiquitous Cloud Of Computing. In Electronic Computer. Sept. 15, 2008. Available from: https://www.electronicdesign.com/markets/mobile/article/21778269/visions-of-the-future-part-1-a-ubiquitous-cloud-of-computing

[11] Lee EA, Hartmann B, Kubiatowicz J, Rosing TS, Wawrzynek J, Wessel D, et al. The swarm at the edge of the cloud. IEEE Design & Test. 2014;**31**(3):8-20

[12] Alippi C, Fantacci R, Marabissi D, Roveri M. A cloud to the ground: The new frontier of intelligent and autonomous networks of things. IEEE Communications Magazine. 2016;**54**(12):14-20

[13] Costa LC, Rabaey J, Wolisz A, Rosan M, Zuffo MK. Swarm os control plane: An architecture proposal for heterogeneous and organic networks. IEEE Transactions on Consumer Electronics. 2015;**61**(4):454-462

[14] Rabaey JM. The human intranet—Where Swarms and humans meet. IEEE Pervasive Computing. 2015;**14**(1):78-83

[15] Fedrecheski G, Rabaey JM, Costa LC, Ccori PCC, Pereira WT, Zuffo MK. Self-sovereign identity for IoT environments: A perspective. In: 2020 Global Internet of Things Summit (GIoTS). Dublin, Ireland. 2020. pp. 1-6

[16] Bartolomeu PC, Vieira E, Hosseini SM, Ferreira J. Self-sovereign identity: Use-cases, technologies, and challenges for industrial iot. In: 2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), Zaragoza, Spain. 2019. pp. 1173-1180

[17] Milić L, Jelenković L. A novel versatile architecture for internet of things. In: 2015 38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), Opatija, Croatia; 2015. pp. 1026-1031

[18] Sethi P, Sarangi SR. Internet of things: Architectures, protocols, and applications. Journal of Electrical and Computer Engineering. 2017;**2017**. Article ID 9324035, pages 25

[19] Salah T, Zemerly MJ, Yeun CY, Al-Qutayri M, Al-Hammadi Y. The evolution of distributed systems towards microservices architecture. In: 2016 11th International Conference for Internet Technology and Secured Transactions (ICITST); Barcelona, Spain. 2016. pp. 318-325

[20] Hunkeler U, Truong HL, Stanford-Clark A. MQTT-S—A publish/subscribe protocol for wireless sensor networks. In: 2008 3rd International Conference on Communication Systems Software and Middleware and Workshops (COMSWARE'08); Bangalore, India. 2008. pp. 791-798

[21] FIWARE Foundation. Fiware. The Internet. 2023. Available from: https://www.fiware.org/

[22] De Biase LC, Calcina-Ccori PC, Fedrecheski G, Duarte GM, Rangel PS, Zuffo MK. Swarm economy: A model for transactions in a distributed and organic iot platform. IEEE Internet of Things Journal. 2018;**6**(3):4561-4572

[23] Calcina-Ccori PC, De Biase LCC, Fedrecheski G, da Silva FSC, Zuffo MK. Enabling semantic discovery in the swarm. IEEE Transactions on Consumer Electronics. 2018;**65**(1):57-63

[24] De Biase LC, Ccori PC, Fedrecheski G, Navarro D, Lino RY, Zuffo MK. Swarm minimum broker: An approach to deal with the internet of things heterogeneity. In: 2018 Global Internet of Things Summit (GIoTS). Bilbao, Spain: IEEE; 2018. pp. 1-6

[25] Esquiagola J, Costa L, Calcina P, Zuffo M. Enabling CoAP into the swarm: A transparent interception CoAP-HTTP proxy for the internet of things. In: 2017 Global Internet of Things Summit (GIoTS). Geneva, Switzerland. 2017. pp. 1-6