Interconnection of Systems with Cloud-Fog-Edge architectures: Concept and Challenges

Cosmina Corches*, Mihai Daraban, Ovidiu Stan*, Szilárd Enyedi*, Liviu Miclea***

**Automation Department, Technical University of Cluj-Napoca (UTCN), 400027 Cluj-Napoca, Romania (e-mail: Cosmina.Corches@aut.utcluj.ro)*
***Applied Electronics Department, Technical University of Cluj-Napoca (UTCN), 400027 Cluj-Napoca, Romania (e-mail: Mihai.Daraban@ael.utcluj.ro)*

Abstract: The advent of Internet of Things (IoT) has sparked enthusiasm and captured the attention of most industries. However, the development of IoT devices took place without the context of a standardization process regarding the data that these devices can communicate or regarding the protocols through which they communicate. Currently, the desire is to use the information from IoT devices in building the context related to the user actions. The user's data IoT equipment and the results of context analysis, such as road traffic control, can be used in monitoring of environmental conditions or coordinating emergency situations. The paper aims to present the challenges that arise when trying to connect multiple IoT devices or sensors in a Cloud-Fog-Edge architecture.

Keywords: cloud computing; fog computing; edge computing; Internet of Things; heterogeneous sensors; wireless sensor networks; heterogeneous networks; interconnected systems; interoperability

1. INTRODUCTION

Nowadays, almost all the devices that surround us can be accessed or controlled via the Internet. In this paper, we present details associated with the challenges of achieving interconnection and interoperability between IoT devices. Sensors transmitting collected data to a dedicated data center (BILLION, 2018; Gallis, 2018) is one thing, it is another to use the data in a correlated manner to build context or to manage current events (Almeida et al., 2017; Duan et al., 2018).

To help other researchers, we went through different studies and concepts that tackled interoperability in IoT technology. The goal is to identify the technologies or approaches to achieve interconnection and interoperability for heterogeneous IoT devices.

2. CLOUD-FOG-EDGE ARCHITECTURE

2.1 Cloud Computing

Cloud Computing represents a new phase of industrialization in provision of computing power as a public service, its development being comparable to how power plants' industrialization influenced the distribution of electricity (European Commission, 2012).

The literature has numerous definitions of the term Cloud Computing, some of them simplified, others very abstract, but nonetheless there is only one definition unanimously accepted by specialists in the field, formulated by NIST (US National Institute for Standards and Technology).

According to this definition, Cloud Computing is a model that allows, on request, easy remote access via the Internet to a configurable set of shared resources that can be provided quickly, with minimal effort or with minimal interaction from the service provider (e.g. networks, servers, external memory, applications and services) (Mell, Grance, 2011). Depending on user requirements, several Cloud Computing solutions are available. These can be grouped into three service models (Dumitrache, Sandu & Barbu, 2017): SaaS, PaaS, IaaS.

2.2 Fog Computing

Fog Computing is an extension of the Cloud that emerged in the context of IoT development. The number of devices that collect data, and the amount of data processed, is growing exponentially.

The Cloud provides remote servers to process this data, but sending it and getting back the results takes time, which can affect a real-time system. In addition, when the Internet connection is not characterized by high bandwidth, reliance on the remote server becomes a challenge.

Cisco defines Fog Computing as a paradigm that extends Cloud and network edge services and that provides end users with data, computing, storage, and application services (CISCO, 2015). In a practical implementation, Fog Computing represents a geographically distributed computing architecture, characterized by the shared use of resources.

A Fog architecture comprises one or more heterogeneous devices that are ubiquitously connected to the edge of the network and are not supported exclusively by Cloud services (Yi et al., 2015). The Fog level is characterized as a low latency intermediary-bidirectional link, which ensures the transfer of data from the Fog level to the Edge level or vice versa.

2.3 Edge Computing

Edge Computing redefines the computation and analysis of data, as processes which are close to the IoT devices or sensors that generated the data flow. This level improves the security and quality of service (QoS) imposed by new applications (Cao, Zhang & Shi, 2018; Hu et al., 2018; Mois et al., 2010). Besides processing data for Edge-connected devices or sensors, the Edge layer is also responsible for processing data received from the layers Cloud and/or Fog. For example, two of the most requested processes today, face recognition and neural network computation, benefitted from moving the computation from the Cloud layer closer to the device on the Edge layer. For the first application, face recognition, the response time was reduced from 900 ms to 169 ms (Jridi et al., 2018), meanwhile for a portable device that was using neural network computation the response time was kept between 80 ms to 200 ms when moved to Edge (Ha et al., 2014).

For a mobile carrier to offer low-latency access to network resources for a user, the services typically located on the Internet (i.e. Cloud computing) needs to be placed right at the edge of the mobile carrier network (Giust et al., 2018). With Multi-Access Edge Computing (MEC), Edge Computing is enabled at the access network (i.e. mobile). Compared to other “edge computes”, MEC can offer new services (Sabella et al., 2019): extreme user proximity, ultra low latency, high bandwidth, real time access to radio network and context information, and location awareness. It is important to note that the resources within an Edge or Fog node must be split between the core functions of the node (routing and data transport) and running software applications, to serve the needs of IoT systems or to allow sensor interconnection. Therefore, implementing equipment that acts as Edge or Fog nodes, becomes a challenge. These must be implemented as Cloud systems, additionally APIs or SDKs are required to access resources, and to be able to install software needed to run services.

3. RELATED WORK

The development of IoT has generated interest in a wide range of fields, an example being the interconnection of public transport elements. By interconnecting the sensors from the vehicles and those from the traffic lights, it became possible to develop intelligent traffic control for public transport in certain cities (Bangui et al., 2018). The information collected by the sensors is analyzed at the Cloud layer where various data reports are generated: traffic statistics, vehicle monitoring and maintenance history, traffic management for autonomous vehicles or public parking management.

Thus, as seen in the previous examples, sensor information is transmitted to the Cloud layer by devices, where resources (e.g. computing power, storage space, access to databases) are accessed by multiple users, on the principle of the multi-tenant model. Resources are accessed through communication networks using standard protocols and mechanisms (Roman, Lopez & Mambo, 2018). The multitude of protocols and devices that can be used raises real challenges in achieving interconnection and interoperability between IoT devices in a Cloud-Fog-Edge architecture.

3.1 Interconnection and Interoperability

Interconnection represents the process of communication between networks. This type of communication only deals with the lower levels of the Open System Interconnection (OSI) model: the physical level, the data link and the network level. Interoperability is the communication between two active processes. Therefore, it is treated above the transport level of the OSI model. Thus, interoperability characterizes the ability of two systems to exchange data (Naik, 2017; Rachna, 2001): lossless; in an unambiguous way; in a format that both systems understand/support; in a manner in which data interpretation is similar. Interoperability is divided into four levels (Veer & Wiles, 2008), Fig. 1:

- Technical interoperability – is associated with hardware/software components, systems and development platforms that facilitate machine-to-machine (M2M) communication.

- Syntactic interoperability – treats the influence of the data format on the interoperability.

- Semantic interoperability – guarantees that the information exchange between two systems is understood at both ends.

- Organizational interoperability – ensures the transmission of information between organizations that are using various information systems.

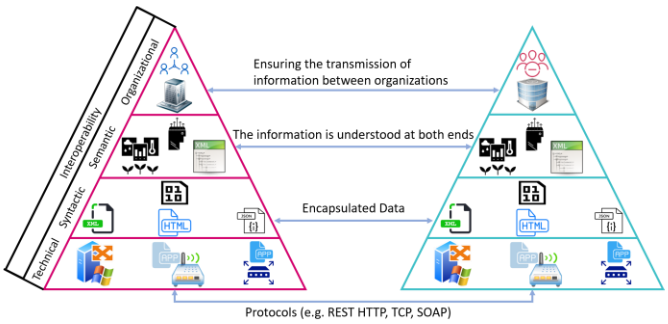

Fig. 1. Interoperability levels.

Although there are solutions for technical and the syntactic interoperability, semantic interoperability remains a great challenge. As there are many solutions for implementing IoT sensors and devices, each manufacturer has its own data transmission protocol and interpretation. Another interoperability issue is that of connecting the devices to the Cloud-Fog-Edge architecture.

4. RELATED WORK

Trying to scale a classical communication network can be a challenge. This process quickly becomes difficult, especially when millions of heterogeneous nodes must be added and some of them are even mobile. At the same time, the quality of services must not be affected, and costs should be kept as low as possible.

In classical networks which use dedicated hardware components for data processing and transmission, the control and data transmission components are unified. By using dedicated components, changes in network scaling will be more time-consuming, especially because specialized personnel is needed and because of hardware performance limitations. In a network with classical architecture, the following limitations were identified: complex network devices, difficulty in managing headers, and difficulty in scaling the network (Bahga & Madisetti, 2014a; Bahga & Madisetti, 2014b).

4.1 Achieving Interconnection in Cloud-Fog-Edge using SDN and NFV

Within the nodes of a Cloud-Fog-Edge architecture, the actions that deal with network traffic management are separated from the software applications that run as services to serve the Cloud-Fog-Edge architecture.

This is done by using Software-Defined Networking (SDN) and Network Function Virtualization (NFV) technologies. SDN and NFV are necessary technologies for next-generation networks, especially for the emerging 5G mobile network (Chairman of ISG ENI, MEC, NFV and ZSM, 2019).

Through NFV, standard IT virtualization technology is used to merge heterogeneous network infrastructure (e.g. physical switches, physical routers, load balancer, firewall) into standard high-volume servers. The result consists in designing network functions in software running on a homogeneous, industry-standard infrastructure.

In such an implementation, the Cloud-Fog-Edge architecture using SDN would be based on the processing power of some servers, to the detriment of specially designed embedded components running a specific set of functions, typical of traditional implementations.

With such a concept, the SDN server platform will be at the Cloud level (Lea, 2018; Wang, 2019), where it serves network services at the software level instead of using an Application Specific Integrated Circuit (ASIC). Without the SDN component, Edge-level routers are not able to provide the services specific to the Cloud-Fog-Edge architecture.

In the literature, there are proposals to install an SDN controller at each node of the Fog layer. The SDN would have to manage the Edge nodes and the devices they are serving. In such a scenario, the Cloud layer will act as a general controller, having a virtual overview of the network. Such an approach enhances security and can isolate compromised nodes, or those nodes that are no longer online, by redirecting data traffic (Kahvazadeh et al., 2019; Sharma, Chen & Park, 2017).

NFV is going to create new network devices, which then will be managed by SDN. As a result, SDN will manage both physical and virtual devices, having the advantage of mixing and matching it well. With SDN and NFV, it is much easier to bring Cloud resources to MEC and achieving the speed and latency improvements promised by 5G (Jun, 2020).

4.2 Protocols for Achieving Interconnection in Cloud-Fog-Edge

For interconnecting IoT devices, the protocol chosen for this task represents an important factor, since it may have a major impact on the device power consumption. Another important criterion used in selecting an interconnection protocol for IoT is communication range. Additionally, the carrier frequency and the communication bandwidth have a greater impact on the protocol’s overall performance, since they will affect the power consumption, communication range, the ability to work in confined spaces, and the data bandwidth. If the protocol is using non-ISM (Industrial, Scientific, and Medical) bands, additional license fees will be applicable.

In terms of connectivity of an IoT device, there is a major change regarding data transfer compared to a typical consumer-based cellular device. The data traffic on a smartphone mainly consists of downloading large amounts of information and real time streaming from the Internet (e.g. video data, music data) (Lea, 2018). However, for an IoT device, the data can be sparse and arrive in short bursts, and in most cases the data will be generated by the IoT device and uploaded to superior levels (e.g. Cloud, Fog, Edge).

When choosing a protocol for interconnection, an advantage is already the presence of a telecommunications network in place. In such a scenario, the users will connect without having to deploy the network, from this point of view mobile networks having a major advantage.

Prior to the IoT devices revolution, there were studies and proposals that contributed to implementing different sensor networks, which allowed to communicate the measured parameters or even exchanging configuration messages (Yick, Mukherjee & Ghosal, 2008; Al-Fuqaha et al., 2015). The proposed concepts used wireless technologies such as Wi-Fi, Bluetooth or Zigbee to interconnect sensors, thus ensuring a local sensor network. Such approaches were used as there was no support for IoT over mobile networks.

However, since the 3rd Generation Partnership Project’s (3GPP) release of CAT-1, CAT-0 and, starting with 2016, of NB-IoT (CAT-NB1) and CAT-M1, things have changed regarding mobile networks.

In Table 1, we can see the advantages brought by these technologies compared to the approaches based on Wi-Fi, Bluetooth and Zigbee (Lea, 2018; Lee, Su & Shen, 2007). Even if at first glance, NB-IoT seems to be the slowest protocol regarding download and upload speeds, it is important to associate these criteria with the channel bandwidth, which translates in low power consumption (i.e. almost 10 years operation from one battery) and the capability to transmit from underground installations (I-SCOOP, 2019a).

Another approach for interconnecting IoT devices over long distances is LoRa and Sigfox. Although communication is carried out using ISM license-free bands, the two protocols require intermediary gateways to allow the connection of devices to the Internet, Table 2 (Lea, 2018).

Comparing Table 1 and Table 2, depending on the transfer speed on the uplink, power consumption, communication range, the most suitable protocols for interconnecting IoT devices are Cat-M1, NB-IoT and LoRa. In terms of latency, the protocol with the fastest response is Cat-M1.

Table 1. Wireless communications protocols for IoT

Criteria	Cat-1	Cat-0	Cat-M1	NB-IoT	Bluetooth	Zigbee	Wi-Fi
ISM Bands	No	No	No	No	Yes	Yes	Yes
Channel Bandwidth	20 MHz	20 MHz	1.4 MHz	180 kHz	1 MHz	0.6 MHz 2 MHz	22 MHz
Downlink Rate	10 Mbps	1 Mbps	1 Mbps 375 Kbps	200 Kbps	1 Mbps	250 Kbps	54 Mbps
Uplink Rate	5 Mbps						
Range	30-100 km	30-100 km	~4x Cat-1	~7x Cat-1	10 m	10-100 m	100 m
Sleep Power	High ~ 2 mA	Low	Very Low ~ 15 µA	Very Low ~ 15 µA	Very Low ~ 9 µA	Very Low ~ 12 µA	Very Low ~ 30 µA
Latency	50-100 ms	50-100 ms	10-15 ms	1.6-10 s	6-250 ms	80-130 ms	150 ms

Cat-M1 and NB-IoT protocols were designed to be supported by legacy mobile technology networks (2G, 3G and 4G) but also by the new release, 5G. As these two protocols were designed to reach their full potential with 5G, since 2019 the introduction of Cat-M1 roaming functionality was deployed.

By the end of 2019, AT&T and Vodafone announced the conclusion of an agreement to support NB-IoT roaming functionality (I-SCOOP, 2019b). Being supported by 5G deployment, Cat-M1 and NB-IoT will begin to be the primary solutions for connecting massive webs of IoT devices (e.g. low-cost devices, low energy, small data volumes), Table 3 (ERICSSON, 2019).

Table 2. LoRa and Sigfox wireless communication protocols

Criteria	LoRa	Sigfox
ISM Bands	Yes	Yes
Channel Bandwidth	125 kHz	100 kHz
Downlink Rate	0.3-5 Kbps	100 bps
Uplink Rate	5 Kbps	600 bps
Range	5 km urban 15 km rural	Up to 50 km
Sleep Power	Extremely Low - 1.5 µA	Extremely Low - 1.5 µA
Latency	500 ms to 2 s	Up to 60 s

Table 3. Cellular IoT connections forecast

Year	Cellular IoT connections by segment and technology (billion)			
	Legacy (2G/3G)	**Massive IoT (NB-IoT/Cat-M1)**	**Broadband IoT (4G/5G)** (e.g. high throughput, low latency, large data volume)	**Critical IoT (4G/5G)** (e.g. ultra-reliability, ultra-low latency, very high availability)
2019	0.8	0.1	0.4	
2020	0.87	0.25	0.6	
2021	0.95	0.42	0.78	
2022	1	0.75	1	
2023	1	1.22	1.18	

Having the devices connected to the Internet, it does not mean we have them interconnected. The difference between a smart IoT device and a sensor on Wireless Sensor Network is given by the possibility of connecting to the Internet and the ability to use or access data provided by different IoT devices produced by other manufacturers.

The IoT devices are using the Internet to communicate and receive data from terminals and data centers as diverse as possible from performance and implementation point of view. The common point is the information needed to accomplish the purpose for which the device was designed.

Realizing interoperability between IoT sensors/devices, arises the necessity to encapsulate the messages/data transmitted through a protocol that uses the Internet as a communication support. To achieve this goal, the most widely used protocol in the last 20 years was the Hypertext Transfer Protocol (HTTP) (Bahga & Madisetti, 2014b). Although this protocol has proven to be useful in connecting computers and servers to the Internet, its features are not suitable for networks consisting of IoT sensors and devices.

The challenges in interconnecting IoT devices come from the devices' limited resources, but also because they should be able to work from remote locations. In addition, devices in difficult to access places are suffering from low mobile service coverage, resulting in low data rate. On top of the previous statements, IoT networks need to use well-secured, optimized and scalable protocols to handle the wide range of IoT devices for different network topologies (e.g. mesh network).

The communication between the Edge, Fog and Cloud levels is performed using the Transmission Control Protocol / Internet Protocol (TCP/IP) and User Datagram Protocol / Internet Protocol (UDP/IP) protocols. At the top layers of the Cloud-Fog-Edge architecture, reusing existing protocols and technologies is suitable, because the equipment used is more efficient, allowing the implementation of a packet transmission management system.

In terms of interconnection and interoperability with and between IoT sensors or devices, things are not so simple. One reason is due to the diversity in the IoT environment. An issue is also represented by the large volume of data that is exchanged between devices, but also due to irregular (spontaneous) traffic, based on events and not on planned operations (Bangui et al., 2018; Esposito et al., 2018; Mujica et al., 2018). Due to the previous reasons, a communication between sensors, based on request-response principle, is not suitable, instead a publish-subscribe approach is desired (Eugster et al., 2003).

4.3 Handling Failure Successfully in MQTT

Even though in the last period the MQTT has gained a lot of ground, we must analyze this protocol from the point of view of the faults/defects that may appear and how to manage them. Among the most serious flaws we can identify: a message is not processed correctly, a message can never be processed, the message queue is full.

The first question we must answer is how important the message is and whether it is worth processing even if it is delivered late. If it is not important, then the worker who consumes the message does not have to recognize it, as long as the message was sent to a consumer, then we don't worry about it anymore.

In other scenarios, we want to detect and react to faults/defects that may occur during message processing. By adding a message-state parameter, we can ensure that messages are removed from the queue only when a consumer has accepted the message and notifies that it was successfully processed. If a consumer takes a message and does not acknowledge that it has been successfully processed within an initially established given time, then another consumer receives the message. This can lead to the processing of messages several times by the same or different consumers. For this reason, it is important to make sure that the system can tolerate this behavior and that the expiration settings are set properly during the time that the worker is expected to need to process the message. In addition, this gives us the guarantee of message delivery because we can be sure that the message will be processed at least once.

One of the most important scenarios we should tackle is the case that the message can never be processed successfully. These messages are invalid or simply contain data that the worker does not know how to handle or to interpret it (i.e. semantic interoperability). By always leaving them on the queue, the queue could be blocked with these messages that cannot be processed but are not deleted. One solution to this problem would be to create a dead message bin. Thus, any rejected message is sent to this bin. The dead message bin can be processed later to find out more information, which can help in improving the system.

If it is possible to identify which messages cannot be processed and will never be able to be processed, then we can simply abandon them without further processing and send them to dead message bin. For this, we send a rejection response. It is important to emphasize that because the message cannot be processed this can lead to the collapse of our worker. For this reason, we need to create defensive workers who respond appropriately to failure situations.

Thinking about whether the message should stay in the queue until it can be processed, or if the system should simply work so that it serves the immediate messages as best as possible, is a very important step in queuing. For example, if an application sends both email notifications and a push message to the user's browser, are these types of messages equivalent if they can be delivered late? Does it matter if someone is missed? A common setup would be to add the reload logic to the emails so they are finally sent, even if it is a few minutes late. If the push message to the browser fails, the application may not bother trying to recover, because the notification is less valuable if it is not delivered at the right time.

Typically, the queues are quite small, and their length only increases if there is a problem processing messages. In this case, it is often useful to specify a maximum queue length. Beyond this dimension, messages are either deleted or sent for the exchange of dead letters.

When the queue receives more than the maximum number of messages set, messages are thrown in front of the queue to make room for new messages arriving in the queue. If the queue also has an x-dead-letter exchange, then the thrown messages will go there, otherwise they will be thrown. Another option to maintain queue length is to set a TTL ("time to live") for messages in a given queue. After this time the message has not been processed, then it destroys itself. or is sent to the dead mailbox. For queues where timely delivery is very important, this can be a great way to avoid the situation where the queue quickly becomes larger than can be processed when something goes wrong. In a scenario where the only option is to wipe the queue completely and restart the system, then adding TTL messages can help in keeping only current information and delivering the unwanted messages that clog the queue. A simplified example is presented in Fig. 2.

```
Result: Successfully Processed Message from Queue
i = 0;
while i less than 5 do
    processMessage();
    if processMessage() then
        OK;
    else
        if !(isMessageImportant()) then
            sendMessagetoDeadBin();
            rejectMessage();
        else
            i++;
            addMessagetoQueue();
            if i = 4 then
                sendMessagetoDeadBin();
        end
    end
end
```

Fig. 2. Example failure handling algorithm.

4.4 Publish-Subscribe Model

A network where messages are exchanged via publish-subscribe communication is data-centric. The recipient of a message or package is identified by associating the content of the message with the interest expressed by the recipient for certain topics. Such communication has the advantage that it allows the grouping of sensors and nodes according to the interest shown for a certain message flow (Esposito et al., 2018). Publish-subscribe services are middleware solutions that provide two APIs to the application, depending on the role of the middleware:

- publisher role, responsible for generating notifications related to the events occurring in the system;

- subscriber role, that addresses the application which receives and processes the notifications for which it has been subscribed.

In addition to the two roles mentioned above, there is also the role of Notification Service. The role of the Notification Service is to mediate between the publisher and the subscriber. It also fulfils the following three roles: storing subscriptions from subscribers, managing notifications from the publisher (e.g. data, services) and distributing notifications to subscribers based on subscriptions made.

For the interconnection between IoT devices or routers at the Edge layer, message-oriented communication protocols (Message Orientated Middleware - MOM) have been proposed. For the implementation of MOM-based protocols, either the presence of a broker (mediator) or an intermediary node for organizing the sequence of messages is required.

One of the most popular and widely used protocols for message and data transport between sensors and Cloud layer is Message Queuing Telemetry Transport (MQTT data centric) (Bangui et al., 2018; Mujica et al., 2018). Although this protocol is based on MOM, being of publisher-subscriber type, the protocol has been designed to separate the client transmitting the information from the one receiving it. Therefore, the devices/sensors do not have to know the IP, respectively the communication port on which the device published or consumed the data from the broker's stack.

For properly exchanging messages between the MQTT broker and publisher or subscriber, all participants need to know the topic and the format of the data they want to publish/access. Even if MQTT does not store the messages received in a stack, it keeps the last message received from a publisher in order to be able to provide data to any new subscribers (Lea, 2018). Although MQTT is TCP based, the connection between the broker and the data providers may end because of network reasons or because of hardware failure associated with the IoT sensors or devices.

For wireless sensor networks, MQTT for Sensor Networks (MQTT-SN) was developed. Although the protocol is derived from MQTT, it is optimized for connections between sensors with low data rates, prone to communication loss, which require exchanging short messages and operating on hardware with limited resources. Being a simplified protocol, it does not require the use of a TCP-IP stack, it can be used in a Bluetooth, Zigbee, serial (Serial Peripheral Interface, Inter-Integrated Circuit) or UDP transmission.

The MQTT-SN protocol facilitates communication with the Fog layer mainly because of how the gateways are implemented. Edge-layer routers will communicate with Fog-layer gateways (i.e. Fog servers) via the MQTT-SN protocol. These servers will be responsible for further processing or transmission of data to the Cloud layer. The interconnection with the Cloud layer will be done through MQTT. At the Fog layer, the conversion from MQTT-SN to MQTT will be carried out. This can be achieved by simply transforming the data flow using a transparent gateway, either by processing it and organizing the traffic to optimize the communication between Fog and Cloud by using an aggregating gateway.

4.5 RESTful Model

An alternative to protocols based on the MOM principle is the RESTful model. A RESTful web service is a web API implemented using HTTP and Representational State Transfer (REST) principles (Bahga & Madisetti, 2014a). In this implementation, a server is the one who knows the status of a certain resource, and the information about it is not transmitted further to the client during the exchange of messages.

Another protocol capable of transmitting messages and data in a Cloud-Fog-Edge architecture is CoAP (Constrained Application Protocol). The protocol was implemented to perform communication between machines (M2M), more precisely between nodes at the Edge layer. As the protocol evolved, the possibility of HTTP mapping using proxies was added. Through this improvement, Internet data transfer can be achieved. CoAP has evolved to offer similar functionality to HTTP, but with reduced overhead and power consumption, [39][40]. In some implementations, CoAP can even outperform HTTP when running on the same hardware configuration (Lea, 2018).

While for MQTT and MQTT-SN, the presence of a broker is required, for the CoAP protocol the central server may be missing. Therefore, two IoT devices or sensors can communicate using CoAP even in the absence of a server (i.e. a network of sensors without infrastructure).

As presented in the previous paragraphs, the publisher-subscriber services are not characterized by space, time and synchronization (Kang, Kapitanova & Son, 2012), which results in a reduction of the time and effort needed to connect the sensors and performing network scaling. Also, in the case of a sensors network based on brokers and uses of selective notifications, a reduction in the data flow and energy consumption was observed (Bangui et al., 2018). This last aspect is very important in IoT, where reducing battery consumption is hoped to prolong the sensors' activity in the network.

Protocols such as Data Distribution Service, Extensible Messaging and Presence Protocol or Advanced Message Queuing Protocol are just a few of the protocols that can be used to interconnect IoT devices with limited resources (Nastase, Sandu & Popescu, 2017).

Studies have shown that in the field of IoT, a communication based on both publisher/subscriber events needs fewer hardware and electrical resources, compared to a request/response approach (Esposito et al., 2018).

Although these protocols can ensure device interconnection, interoperability is not ensured. Messages can be exchanged, but encapsulated data cannot be understood and used without a common representation.

5. INTEROPERABILITY OF HETEROGENEOUS SENSORS

When it comes to the interconnection of IoT sensors or devices, in most cases, wireless technologies are used, so that the devices' mobility is not affected.

5.1 Technical Interoperability

To ensure technical interoperability for IoT devices and maintain compatibility with current technologies, protocols such as TCP/IP or UDP/IP at the transport layer, and MQTT at the application layer should be used. By using such an architecture, compatibility with current networks is maintained (EMQ, 2019).

Most IoT devices are powered by batteries or from renewable energy sources, which makes them incompatible with traditional wireless technologies. As presented in Table 1, low power consumption solutions like NB-IoT and Cat-M1 can be used. These protocols will reach their full potential with the deployment of 5G technology (I-SCOOP, 2019a; I-SCOOP, 2019b). There is also the problem of already deployed sensors that work on non-IP wireless technologies (Bluetooth, Zigbee, RFID).

To solve this issue, a proposed solution was to add an intermediary node between Edge and sensors or IoT devices to facilitate the conversion. Mujica et al. propose a platform that realizes the transition from non-IP to IP protocols or from technologies associated with wire sensors to wireless data transmission (Mujica et al., 2018). Through the proposed platform, the authors move the problem of technical interoperability from the Edge layer to a node closer to the sensors, which instead allows the Edge nodes to deal with tasks such as processing or packet routing.

Messages propagation between sensors is another important aspect of technical interoperability. A solution for this topic proved to be publisher/subscriber protocol combos (MQTT and MQTT-SN) (EMQ, 2019; Esposito et al., 2018). As the central node is responsible for adding a new node in the network, a problem with such an organization emerges from having the node placed at the top layer. At such a position in the infrastructure, complex computations should be performed, not network management. One solution is to introduce broker-type nodes at the Fog layer, so that the central node no longer needs to communicate with each individual sensor, but with nodes delegated to manage the data packages. Another approach is not to deploy broker-type nodes, but instead the communication between the nodes to be based on routing lists that are constantly updated within the network between the connected devices (Esposito et al., 2018).

5.2 Syntactic and Semantic Interoperability

For achieving syntactic interoperability, the challenges consist in identifying a mapping scheme that allows the exchange of messages between sensors, regardless of the protocol used and the way of transmitting the data bits.

Creating mapping schemes for every data transmission protocol and technique would be far too complex when updates are needed, but also because of the resources involved. A proposed solution is to transmit the mapping scheme from the central nodes to the sensors (publishers), so they know what data format is recognized by the receivers (subscribers). The technique suits an architecture centered on a broker, but also advanced implementations, where the mapping table is received from the neighboring nodes (Esposito et al., 2018).

A common technique is to encapsulate the sensors' messages in XML or JSON. Through an intermediary platform such as the one in (Mujica et al., 2018), it is possible to convert or arrange the received data from the sensors into JSON messages, which will then be understood by all the subscribers interested in the subject. The challenge with syntactic interoperability comes from the tight relationship between device hardware development and the software application. Without a close communication between the two teams, the software application cannot properly map the data from the sensor to the data format recognized by subscriptions.

Another approach in achieving syntactic interoperability is through Action Oriented Programming (AcOP) (Mäkitalo et al., 2018). With AcOP, the ways of interaction between the devices and their operating environment is revised. The AcOP model is realized in JavaScript and includes constructs derived from Social Devices concept (Mäkitalo et al., 2018):

- sensation – an abstract representation of the input coming from the physical, cyber or social world (e.g. a sensor changing value);

- capability – physical objects are identified as JavaScript objects, that describe how a certain device can interact with other devices or humans;

- action – modular unit describing in JavaScript how several devices interact with each other over a certain period of time;

- collective execution – in AcOP a set of devices form a coalition based on trust negotiations with the purpose of sensing and acting toward a common goal.

Beside syntactic interoperability, AcOP can also serve in achieving semantic interoperability through the abstraction of inputs, capabilities of an object, actions performed. To reach syntactic and semantic interoperability in IoT, one needs to develop APIs to support all the devices interconnected. To keep up with device development, APIs should be available to download or update from data centers as a kind of webstore for application services. The implementation and deployment of MEC will help in facilitating API download and managing collective execution.

In conclusion, intermediary platforms, like the one proposed by Mujica et al., and MEC with AcOP solution, will bridge the gap between non-IP sensors (both wireless and wired ones) and the more capable IoT devices. The purpose of single board computers as intermediary platforms will be to map the data from non-IP heterogeneous sensors/devices and to implement AcOP.

6. DISCUSSION

This paper presents the complexity of connecting heterogeneous IoT sensors or devices in a Cloud-Fog-Edge architecture. The study focused on presenting interconnection technologies based on wireless solutions so that the device mobility is not affected. From the protocols analyzed, Cat-M1 and NB-IoT proved to be the best solutions as having the highest data bandwidth, largest coverage and small energy consumption. The deployment of 5G will only help in reaching the full potential of these two protocols. A feature that distinguishes mobile network protocols from the previous technologies is the introduction of IoT roaming by network providers. Although roaming has been recently introduced also in LPWAN (Low-Power Wide-Area Network) technologies like LoRa, even keeping the lower-level data encrypted, the sheer size of telecom operators will grow 5G-based IoT beyond what LoRa or Sigfox can commercially offer. An analysis of real-life roaming support offered by old and new technologies would be interesting, but again, when they are not only technologically, but also commercially mature, deployment-wise.

Achieving interoperability in a heterogeneous environment, not only from the transmitted information point of view but also from the perspective of hardware and the technologies used, can prove to be a real challenge. The solution is to use innovative techniques (Esposito et al., 2018; Mujica et al., 2018) and to resort to domain standardization (Sabella et al., 2019; Chairman of ISG ENI, MEC, NFV and ZSM, 2019) which will bring benefits not only from the communication point of view, but also for security, process and time required on the development side. As 5G and MEC will be deployed, attention should be paid on API development to support interoperability through techniques as AcOP. For IoT interoperability to become a reality, mobile carriers, equipment vendors, and software developers will have to work together to build a standard for APIs intended for MEC.

ACKNOWLEDGEMENTS

This research was funded by the Ministry of Education and Research of Romania, section CCCDI - UEFISCDI, grant number PN-III-P1-1.2-PCCDI-2017-0734 / ROBIN - “Roboții și Societatea: Sisteme Cognitive pentru Roboți Personali și Vehicule Autonome” (Robots and Society: Cognitive Systems for Personal Robots and Autonomous Vehicles) within PNCDI III, and grant SeMed (2933/55/GNaC 2018).

REFERENCES

Al-Fuqaha, A., Guizani, M., Mohammadi, M.; Aledhari, M.& Ayyash, M. (2015). Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications, *IEEE Communications Surveys & Tutorials*, *17*(4), 2347-2376.

Almeida, F.R., Brayner, A., Rodrigues, J.J.P.C.; & Maia, J.E.B. (2017). Improving Multidimensional Wireless Sensor Network Lifetime Using Pearson Correlation and Fractal Clustering, *Sensors*, *17*(6), 1-24.

Bahga, A. & Madisetti, V. (2014). *Cloud Computing - A Hands-On Approach*. CreateSpace Independent Publishing Platform.

Bahga, A. & Madisetti, V. (2014). *Internet of Things: A Hands-On Approach*. CreateSpace Independent Publishing Platform.

Bangui, H., Rakrak, S., Raghay, S. & Buhnova, B. (2018). Moving to the Edge-Cloud-of-Things: Recent Advances and Future Research Directions, *Electronic*, *7*, 1-31.

Billion (2018), Case Study: LoRa Smart Meter, *billion.com* [Online]. Available

<https://www.billion.com/upload/web/Case/Billion-Case-Study-4700ZU-LoRa-Smart-Meter.pdf>.

Cao, J., Zhang, Q. & Shi, W (2018). *Edge Computing: A Primer*. Springer International Publishing, ISBN 978-3-030-02082-8.

Chairman of ISG ENI, MEC, NFV and ZSM (2019). *Network Transformation; (Orchestration, Network and Service Management Framework)*, ISBN No.979-10-92620-29-0.

Cisco Systems (2015). Fog Computing and the Internet of Things: Extend the Cloud to Where the Things Are, *Cisco.com* [Online]. Available <https://www.cisco.com/c/dam/en_us/solutions/trends/iot/docs/computing-overview.pdf>.

Duan, K., Fong, S., Zhuang, Y.; & Song, W. (2018). Improving Oxides Gases for Occupancy Counting and Emergency Control in Fog Environment, *Symmetry*, *10*(3), 1-16.

Dumitrache, M., Sandu, I. E. & Barbu, D. C. (2017). An Integrated Cloud Computing Solution for Romanian Public-Sector Entities: ICIPRO Project, *Studies in Informatics and Control*, *26*(4), 481-487.

EMQ [2019], Smart China Expo – EMQ & Intel MEC IoT Edge Solution, *EMQ* [Online]. Available <https://www.emqx.io/blog/2019-smart-china-expo-emq-intel-mec-iot-edge-solution>.

Ericsson [2019], Ericsson Mobility Report November 2019– *ericsson.com* [Online]. Available <https://www.ericsson.com/4acd7e/assets/local/mobility-report/documents/2019/emr-november-2019.pdf >.

Esposito, C., Castiglione, A., Palmieri, F., Ficco, M., Dobre, C., Iordache, G. V. & Pop, F. (2018). Event-based sensor data exchange and fusion in the Internet of Things environments, *Journal of Parallel and Distributed Computing*, *118*(2), 328-343.

Eugster, P. Th., Felber, P. A., Guerraoui, R. & Kermarrec, A. M. (2003). The many faces of publish/subscribe, *ACM Computing Surveys*, *35*(2), 114-131.

European Commission (2012), Unleashing the Potential of Cloud in Europe, *Eur-lex.europa.eu* [Online]. Available <https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:52012DC0529&from=en >

Gallis, R., (2018), First Sigfox enabled bollard. A case study to better monitor water outlets, *Thinxtra The IoT Telco* [Online]. Available <https://library.web.thinxtra.com/2018/03/first-sigfox-enabled-bollard/ >

Giust, F., Verin, G.; Antevski, K., Chou, J., Fang, Y., Featherstone, W., Fontes, F., Frydman, D., Li, A., Manzalini, A., Purkayastha, D., Sabella, D., Wehner, C., Wen, K.W. & Zhou, Z. (2018). *MEC Deployments in 4G and Evolution Towards 5G*, ISBN No. 979-10-92620-18-4.

Ha, K., Chen, Z., Hu, W., Richter, W., Pillai, P. & Satyanarayanan, M. (2014). Towards Wearable Cognitive Assistance. In *Proceedings of the 12th Annual International Conference on Mobile Systems, Applications, and Services* (pp. 68-81).

Hu, B., Xie, H., Ma, Y., Wang, J. & Zhang, L. J. (2018). A Robust Retail POS System Based on Blockchain and Edge Computing. In *Edge Computing – EDGE 2018. EDGE 2018. Lecture Notes in Computer Science* (pp. 99-110).

I-SCOOP (2019). NB-IoT explained: a complete guide to Narrowband-IoT, *i-scoop.eu* [Online]. Available <https://www.i-scoop.eu/internet-of-things-guide/lpwan/nb-iot-narrowband-iot/ >.

I-SCOOP (2019). LTE-M roaming for North America and several European markets, *i-scoop.eu* [Online]. Available <https://www.i-scoop.eu/internet-of-things-guide/lpwan/lte-m-roaming/ >

Jridi, M., Chapel, T., Dorez, V., Le Bougeant, G. & Le Botlan, A. (2018). Improving SoC-Based Edge Computing Gateway in the Context of the Internet of Multimedia Things: Experimental Platform, *Journal of Low Power Electronics and Applications*, *8*(1), 1-18.

Jun, S. (2020), Challenges & Key Issues of Constructing 'MEC-Ready' 5G Bearer Networks for Carriers, *telecoms.com* [Online]. Available <https://telecoms.com/intelligence/challenges-key-issues-of-constructing-mec-ready-5g-bearer-networks-for-carriers/ >

Kahvazadeh, S., Souza, V. B., Masip-Bruin, X., Marn-Tordera, E., Garcia, J. & Diaz, R. (2017). Securing Combined Fog-to-Cloud System Through SDN Approach. In *Proceedings of the 4th Workshop on CrossCloud Infrastructures & Platforms* (pp. 1-6).

Kang, W., Kapitanova, K. & Son, S. H. (2012). RDDS: A Real-Time Data Distribution Service for Cyber-Physical Systems, *IEEE Transactions on Industrial Informatics*, *8*(2), 393-405.

Lea, P. (2018). *Internet of Things for Archtects*. Packt Publishing.

Lee, J., Su, Y. & Shen, C. (2007). A Comparative Study of Wireless Protocols: Bluetooth, UWB, ZigBee, and Wi-Fi, In Proceedings of the *IECON 2007-33rd Annual Conference of the IEEE Industrial Electronics Society*, Taipei, Taiwan, 46-51.

Mäkitalo, N., Ometov, A., Kannisto, J., Andreev, S., Koucheryavy, Y. & Mikkonen, T. (2018). Safe, Secure Executions at the Network Edge : Coordinating Cloud, Edge, and Fog Computing, *IEEE Software*, *35*(1), 30-37.

Mell, P., Grance, T. (2011). The NIST Definition of Cloud Computing, *National Institute of Standards and Technology* [Online]. Available <https://faculty.winthrop.edu/domanm/csci411/Handouts/NIST.pdf>.

Mois, G., Stefan, I., Enyedi, S. & Miclea, L. (2010). Reconfiguration and hardware agents in testing and repair of distributed systems. In *2010 East-West Design Test Symposium (EWDTS)* (pp. 195-198).

Mujica, G., Rodriguez-Zurrunero, R., Wilby, M. R., Portilla, J., Rodríguez González, A. B., Araujo, A., Riesgo, T. & Vinagre Díaz, J. J. (2018). Edge and Fog Computing Platform for Data Fusion of Complex Heterogeneous Sensors, *Sensors*, *18*(11), 1-26.

Naik, N. (2017). Choice of effective messaging protocols for IoT systems: MQTT, CoAP, AMQP and HTTP. In *2017*

IEEE International Systems Engineering Symposium (ISSE) (pp. 1-7).

Nastase, L., Sandu, I. E. & Popescu, N. (2017). An Experimental Evaluation of Application Layer Protocols for the Internet of Things, *Studies in Informatics and Control*, *26*(4), 403-412.

Rachna, P. (2001). Study on the Interconnection and Interoperability of Information Systems. In *CALIBER 2001: Pune*.

Roman, R., Lopez, J. & Mambo, M. (2018). Mobile edge computing, Fog et al.: A survey and Analysis of Security, Threats and Challenges, *Future Generation Computer Systems*, *78*(2), 680-698.

Sabella, D., Sukhomlinov, V., Trang, L., Kekki, S., Paglierani, P., Rossbach, R., Li, X., Fang, Y., Druta, D., Giust, F., Cominardi, L., Featherstone, W., Pike, B. & Hadad, S. (2019). *Developing Software for Multi-Access Edge Computing*, ISBN No. 979-10-92620-29-0.

Sharma, P. K., Chen, M. Y. & Park, J. H. (2017). A Software Defined Fog Node Based Distributed Blockchain Cloud Architecture for IoT, *IEEE Access*, *6*, 115-124.

Siira, M. (2014). Interconnection, interoperability for integration in the Smart Grid, *Consulting-Specifying Engineer (CSEMAG)* [Online]. Available <https://www.csemag.com/articles/interconnection-interoperability-for-integration-in-the-smart-grid/>.

Thangavel, D., Ma, X., Valera, A., Tan, H. & Tan, C. K. (2014). Performance evaluation of MQTT and CoAP via a common middleware. In *2014 IEEE Ninth International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)* (pp. 1-6).

Veer, H. & Wiles, A. (2008). Achieving Technical Interoperability - the ETSI Approach, *European Telecommunications Standards Institute (ETSI)* [Online]. Available <https://portal.etsi.org/Portals/0/TBpages/CTI/Docs/IOP%20whitepaper%20Edition%203%20final.pdf>.

Wang, X., Chen, X., Wang, Y. & Ge, L. (2019). An efficient scheme for SDN state consistency verification in cloud computing environment, *Concurrency and Computation: Practice and Experience*.

Yi, S., Hao, Z., Qin, Z. & Li, Q. (2015). Fog Computing: Platform and Applications. In *2015 Third IEEE Workshop on Hot Topics in Web Systems and Technologies (HotWeb)* (73-78).

Yick, J., Mukherjee, B., & Ghosal, D. (2008). Wireless Sensor Network Survey, *Computer Networks, 52*(12), 2292-2330.