Performance Evaluation of a Satellite Communication-based MEC architecture for IoT applications

M. Luglio, M. Marchese, *IEEE Senior Member,* F. Patrone, *IEEE Member,* C. Roseti and F. Zampognaro*

***Abstract*—New scenarios and use cases are raising following the birth of the fifth generation of mobile communications (5G). The Internet of Things (IoT) is one of the main use cases which are growing, leading to a massive amount of data that needs to be exchanged throughout the Internet. Satellite Communication (SatCom) networks are essential in remote and isolated environments and can support fully connected environments by offloading the terrestrial infrastructure concerning delay-tolerant traffic flows. However, satellite network resources are limited and expensive, so they need to be carefully used in order to avoid waste and satisfy the required user performance. The Multi-access Edge Computing (MEC) concept can be exploited in this context to allow data pre-processing at the edge, i.e., close to the users, so reducing the amount of data that has to traverse the backhaul satellite link and, in some cases, reducing data delivery times.**

This paper analyses the performance of a satellite architecture in the IoT framework highlighting the advantages brought by MEC, also including data aggregation and compression techniques.

***Index Terms*—Internet of Things (IoT), Satellite Communications, Multi-access Edge Computing (MEC), MQTT**

I. Introduction

WHEN looking at the use of the Internet, the landscape of applications and communication protocols has changed substantially in the last years. Non-real-time information exchange represented most of the Internet traffic in the past, with predominance in the use of Web technology and file transfers (e.g., HTTP, e-mail, FTP, Peer-to-peer solutions). Nowadays, real-time interactions are fundamental for a whole new set of applications, from web gaming to VoIP and chat services, from conferencing platforms to cloud and Internet of Things (IoT) services. In particular, massive IoT applications show a relevant growth and are already representing a significant portion of Internet traffic, as discussed for instance in the latest Cisco Annual Internet Report [1]. Many kinds of sensor-based services fall in this category and are applied in a set of scenarios such as home automation, smart cities, support to logistics, wearable health devices, industrial automation, environmental sensing, and critical infrastructures monitoring and protection. In all these environments, it is often important to consider large coverage and high mobility requirements. An example may be a sensor network which spans across many administrative domains and covers remote areas.

A massive growth and spread of devices, such as sensors and actuators, has been envisioned for 2021 and beyond, with a consequent big mass of data that need to be collected, processed, and, in some cases, displayed to users remotely connected to the Internet.

Satellite-based approaches, such as the Space-Air-Ground Integrated networks [2], are considered as viable solutions thanks to their ubiquitous coverage and broadcast/multicast capabilities. Considering the IoT context and the associated connectivity requirements, satellite access can play a significant role in enabling IoT services [3]. SatCom networks are particularly suitable to offer connectivity to a high number of devices located in wide areas, complementing terrestrial infrastructure thanks to the large coverage. They can support cross-the-borders IoT scenarios properly managing their available network resources to guarantee the required Quality of Service (QoS) [4]

In particular, [3] emphasize the role of satellites in the foreseen spread of IoT devices and applications in the future 5G ecosystem focusing on the challenges related to the massive Machine Type Communication (mMTC) service provision, the air interface, and the network aspects related to the system design. Energy efficiency is another important aspect to take into account. A Network Coding (NC)-based approach has been proposed in [5] to properly manage the available network resources and offer reliable IoT multicast services in a 5G/satellite network. Key network challenges for satellite exploitation in future 5G and IoT satellite-based sensor networks are also presented in [6].

Anyway, assuming a massive amount of data generated by an IoT sensor network composed of thousands of IoT devices, the overall volume of raw data can lead to the rise of several issues. An example is the security and data privacy issue, which involves both legal [7] and technical matters [8] related to different aspects, such as how the network is managed and how data is stored and processed. Another example is related to how the network resources are efficiently exploited to offer the required service and avoid unpleasant situations, such as the saturation of the satellite link bandwidth.

The presence of Multi-access Edge Computing (MEC) functionalities close to the sensor network allows saving satellite communication resources and offering enhanced performance

M. Luglio, C. Roseti and F. Zampognaro are with the Electronics Department of the University of Rome "Tor Vergata", Rome, Italy, e-mail: {luglio|cesare.roseti|francesco.zampognaro}@uniroma2.it

M. Marchese and F. Patrone are with the Electrical, Electronics and Telecommunication Engineering and Naval Architecture Department (DITEN), University of Genoa, Genoa, Italy, e-mail: mario.marchese@unige.it, f.patrone@edu.unige.it

* *Authors in alphabetical order.*

to the users thanks to smart management of large set of data by enabling local processing and in-network functions (i.e., micro-services) tailored on the specific use-case.

The MEC concept, formerly Mobile Edge Computing, has been introduced with the main aim to move the intelligence, or at least part of it, from the network core (centralized datacenters and cloud platforms) to the network edge (distributed micro-datacenters and edge platforms). MEC contributes shaping the next generation network architecture and infrastructure [9], [10] thanks to its applicability in different scenarios and use cases, including IoT [11], [12].

Concerning MEC application in satellite networks, different possible network architecture configurations, challenges, and open issues have been discussed [13]. Possible solutions have been implemented and tested focusing on the integration within the 5G environment [14], based on different kinds of satellites (including nanosatellites [15]), and aimed at improving the user perceived QoS [16], [17].

IoT applications implementing satellite edge computing solutions are the object of [18]–[20], also considering the severe resource limitations that affect IoT devices. Latency and energy consumption reductions are the main aims of the studies [21]–[23], which also adopt Machine Learning (ML) techniques to make resource management more effective.

This paper addresses the integration of satellite and terrestrial networks to interface IoT sensors and efficiently enable IoT services through the Internet. The considered solution includes a satellite backhaul link based on a Geostationary Earth Orbit (GEO) satellite, to offload the terrestrial network of delay-tolerant data, and the employment of the MEC paradigm, to achieve and optimize a fully integrated and global IoT service, i.e., to offer an architectural and operational solution able to guarantee the typical performance requirements of IoT applications worldwide, overcoming limitations of different kinds, such as geographical and data management on vast scale. This solution is assessed in order to quantify and highlight the performance improvement obtained implementing the MEC concept to enhance the satellite transmission, by both performing data compression to save satellite bandwidth, as well as distributing and aggregating data in an efficient way in relation of the domain where it is useful (i.e., local or remote).

The remainder of the paper is organized as follows. Section II highlights the roles that SatCom networks can have in the IoT applications. Section III presents the used MEC-enabled satellite architecture and the considered network configurations. The obtained results are reported in Section IV. Conclusions are drawn in Section V.

II. Role of satellites in IoT applications

Satellites can have two main alternative roles in a typical satellite/terrestrial integrated network for IoT applications: direct-access (fronthaul) or backhaul.

In the first case, sensors are directly linked to the satellites which are the first hop of the communication path between IoT devices and users. Third Generation Partnership Project (3GPP) defined six different reference scenarios [24] changing the altitude of the considered satellites (GEO and Non-GEO), the complexity of the carried satellite payload (transparent or regenerative), and the flexibility of the generated beams (steerable or moving beams) for direct-to-satellite access (fronthaul). The most promising approaches are mainly considering Low Earth Orbit (LEO) and Very Low Earth Orbit (VLEO) satellites, even if intrinsic challenges have still to be overcome, such as the heterogeneity of the IoT solutions and the high energy consumption of IoT devices. Some ongoing activities are investigating these issues to assess the feasibility of direct-to-satellite access, such as [25] and [26].

In the second case, satellites can be adopted as backhaul to the Internet or to a dedicated service center. Sensors data are collected within an Edge Cloud in the terrestrial access network and then made available through a satellite Data Network backhaul by using IP-based protocols. High Throughput Satellites (HTS) and Very High Throughput Satellites (VHTS) GEO platforms operating at Ka or higher frequency bands are the main options. They can considerably reduce the communication costs providing large amounts of bandwidths, making satellites a viable solution to enhance the overall network capability.

The employment of communication satellites for IoT can be beneficial for multiple applications. Several solutions have been developed considering the possible different roles of satellites and tested in different application scenarios [27]. Satellites operating as backhaul support have been considered for the smart city [28] and the road safety and autonomous ship scenarios [29]. Smart agriculture [30], smart grid [31], and maritime IoT [32] are three other typical use cases where the use of satellites has been considered for IoT applications and has been proven to be effective, not only for the lack of communication infrastructures based on different technologies.

From the application viewpoint, data generated by devices need to be transferred to receivers by using a suitable protocol. Two main communication models can be employed: "publish/subscribe" or "client/server", depending on the use-case. Although in principle it is possible to exchange such data in non-IP (NIP)/proprietary formats and associated proprietary transport and application protocols, it is worthwhile to consider the IP-based approach. IP allows enabling interoperability and reusability, leveraging well established and freely available IP-based applications and protocols. The two main IP-based solutions are Message Queuing Telemetry Transport (MQTT) and Constrained Application Protocol (CoAP).

MQTT [33] was originally an IBM proprietary protocol now managed as a standard by the International Organization for Standardization (ISO). MQTT uses a publish/subscribe model and requires an MQTT Broker to manage and route messages among MQTT nodes through point-to-point TCP permanent connections. MQTT has not been designed for "constrained" devices, so power saving optimization and trade-offs among complexity/efficiency and bandwidth usage are usually not considered.

CoAP [34] is a lightweight IP-based protocol for sensor data communication. It is part of the Constrained RESTful Environments (CoRE) IETF working group and it aims to realize a REpresentational State Transfer (REST) architecture in a suitable form for the most constrained nodes. CoRE is a very efficient RESTful protocol based on the classic client/server model. Even if DLTS, SMS, and TCP bindings are defined as transport for CoAP, UDP-based implementations are the most commonly used.

Studies have been carried out to evaluate the employment of MQTT and CoAP in the satellite backhaul case [35]. Efficient configurations for both MQTT and CoAP have been highlighted and some modifications have been proposed. For example, MQTT and CoAP employment to support sensor data exchange on a satellite Random Access (RA) channel based on the DVB-RCS2 standard has been investigated in [36].

III. A MEC-ENABLED IOT ARCHITECTURE

A. Reference Architecture description

The locations where MEC functionalities can be typically implemented in satellite networks are three: on the ground close to the users, on-board satellites, and on the ground close to the satellite ground stations. Implementing MEC on-board satellites require regenerative payload satellites with largely available computation and storage resources and an internal structure designed to make their allocation flexible and scalable. Companies are planning to launch this kind of satellites which will be available in the near future. However, a first step to deploy and test satellite MEC networks could be to exploit the already deployed transparent payload satellites. It is important to quantify the achievable performance improvement in different possible IoT use cases with different and variable network and traffic flow features, such as number and density of IoT devices, distribution of generated data flows, and minimum performance requirements.

We consider exploiting the MEC concept and a satellite link to enable IoT scenarios in remote areas. In particular, we focus on GEO-based communication systems, such as Inmarsat, configured as backhaul. Unmanned Aerial Vehicles (UAVs) could also be considered and included in the reference scenario to help collect data from the sensors. However, we decided to not include them since our main focus is on the backhaul link which, especially for rural and remote areas, can be more realistically assumed as a satellite link, while UAVs can be more useful for the access link. Besides, the presence of UAVs would not affect the application of our proposed MEC functionalities and the effect on the shown performance would be negligible, because the main improvement comes from a local management of data."

Figure 1 shows the considered reference network architecture, which was validated within the European Space Agency (ESA) funded project M2MSat[1]. A large and widespread set of IoT devices, called Sensor Equipment (SE), are the data sources of this sensor network scenario. They are linked to a base station located in the remote area through wired or wireless links. The base station aims to collect the data generated by the SEs in order to send them to the Internet through the satellite backhaul link. An IoT Cloud service, located on the Internet, collects and stores all SE data and offers data access to authorized users, called User Equipment (UE). UEs can be located on the Internet (on the right side of the satellite link, called Core side) or connected directly from within the Service Provider Network (on the left side of the satellite link, called Edge side). In this way, the satellite offers a backhaul solution that can completely replace the terrestrial connectivity to the Cloud Services.

It is assumed as well for the reference service, that not all sensor data have the same scope and priority. In fact, it is possible to consider several use-cases in which some remote data collected into a remote area can just be used for local data processing, whereas only alarms or downsampled values are delivered towards a central node. Improve satellite resource management and computation offloading is the main aim of most studies on this topic, such as [37], [38].

In this context, MEC can foster satellite bandwidth reduction through the implementation of dedicated functionalities, such as data aggregation and local processing at the edge, with a consequent lower cost due to the reduced amount of data that need to traverse the satellite link. The proposed reference architecture may in fact include storage, computation resources, and data local breakout functionalities which are part of the Network Operator infrastructure, but not necessarily present at the Edge of the network and not enabled for MEC. For instance, thanks to the presence of network functions and components for the termination of the IoT protocol stack within the Network Operator's Core Network, a possible edge computing infrastructure and the associated components can be conceived and enabled in a straightforward way. These dedicated components can be optimized for IoT and deployed dynamically, e.g. as Virtual Network Functions (VNFs), allowing to manage and terminate the local data-application protocols before the transmissions over satellite, as discussed in the next sub-section.

B. Possible network configurations and MEC operations

In relation to current standardization activities and on the basis of the state of the art and research projects [25], [39], [40], having Figure 1 as a reference, we introduce three possible configurations, leveraging data local breakout. The local breakout means that the Operator Network is supporting the split of data sessions for the sensor data, delivering such data in different configurations according to the enabled MEC options (if any). The goal is to present a progressive performance improvement first due to the enabling of MEC, and then the enabling of satellite-oriented ad-hoc optimizations which are tailored on the data type and meaning. In particular data compression, aggregation, suppression, and differential distribution will be introduced as edge operations in the last configuration:

- *Case 1* (Figure 2): SEs are IoT MQTT Sensors. Due to the large number of involved sensors and users, we consider the use of MQTT because it scales better with a large population of nodes thanks to a broker node which manages the data exchange between data sources and users. Edge Computing is not included in this case. IoT MQTT Sensors generate data destined to the centralized

[1]https://artes.esa.int/projects/m2msat

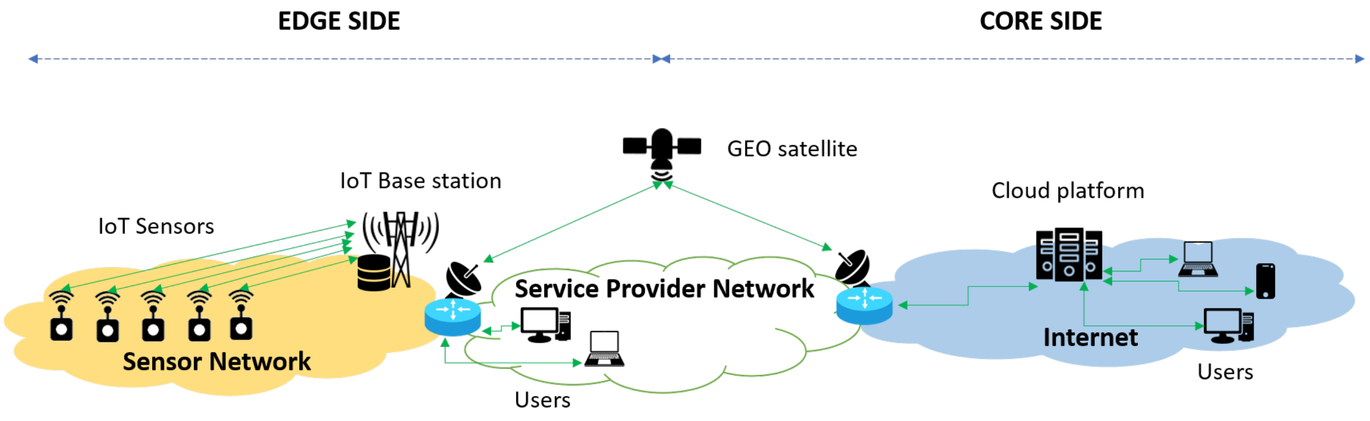

Fig. 1: Reference hybrid network to enable IoT Cloud Services via satellite

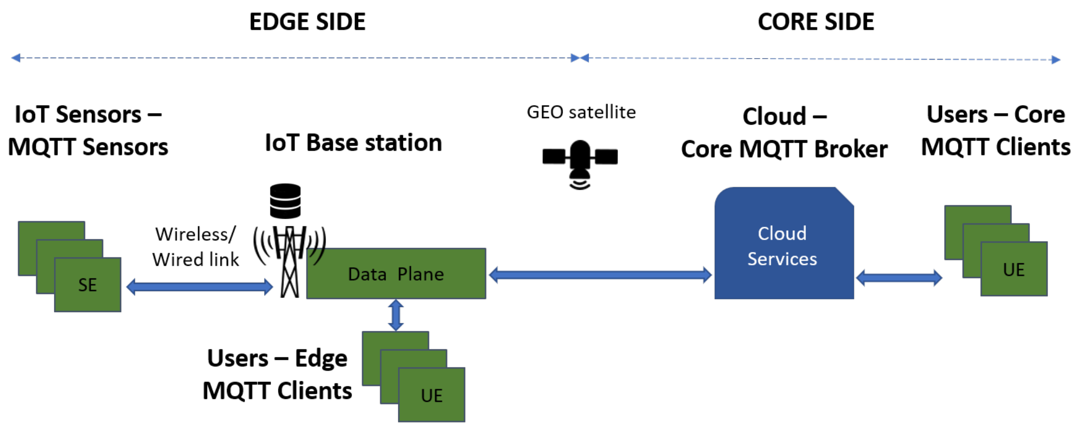

Fig. 2: IoT application level network configuration - Case 1: IoT MQTT native Sensors, No Edge Computing. All MQTT Clients access the Core MQTT Broker, Edge MQTT Clients requests and responses need to traverse the satellite link

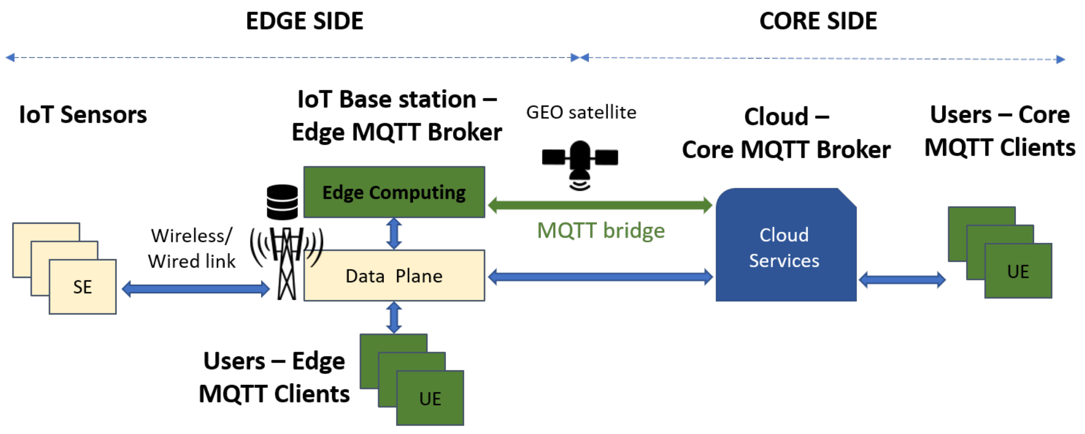

Fig. 3: IoT application level network configuration - Case 2: IoT non-MQTT Sensors, Edge MQTT Computing in the IoT Base station. Presence of an Edge MQTT Broker to store information made available to the Edge MQTT Clients that, in this way, do not need to access the Core MQTT and so forward their request through the satellite link

MQTT Broker (Core MQTT Broker) located on the Core side and hosted by Cloud Services. MQTT Clients, that are located both on the Edge and Core sides (Edge and Core MQTT Clients, respectively), request data to the MQTT Broker. Besides, no additional functionalities dedicated to satellite bandwidth optimization and data pre-processing are included.

- *Case 2* (Figure 3): SEs are IoT non-MQTT Sensors. They get access to the network through different interfaces (e.g., LoRaWAN) equipped with their own application layer protocols. MEC is partially added: an Edge MQTT Broker is introduced at the Edge side with a two-fold aim. On one hand, to "convert" the generated data in the MQTT format in order to let them be stored in MQTT Brokers and, consequently, allow MQTT Clients to access them. On the other hand, to provide local data storage at the Edge side. To guarantee the compatibility with the Core MQTT Broker, the Edge MQTT Broker has to be configured to act as a transparent proxy to the target Core MQTT Broker, i.e., redirecting the traffic to it after locally storing a copy of each content. Edge MQTT Clients can directly ask the Edge MQTT Broker to provide the interested contents so reducing the amount of data that need to traverse the backhaul satellite link in both directions: MQTT requests from Edge MQTT Clients to the Core MQTT Broker and MQTT answers from the Core to the Edge side. Also in this case, no additional functionalities aimed at satellite bandwidth utilization improvement and data pre-processing are included. MEC functionalities are limited to the presence of the Edge MQTT Broker.

- *Case 3* (Figure 4): SEs are IoT not-MQTT Sensors and the network architecture includes both the Core and Edge MQTT Brokers as in Case 2. The Edge computing component also enables a data compression and data aggregation functionality. In detail, MEC functionalities include also: i) local processing, such as downsampling and duplicated values suppression, to perform data compression; ii) data aggregation, such as aggregating different values by collecting a given amount of messages over time in a single MQTT message to reduce the protocol header overhead. This type of MEC functionalities are already proposed in [41]. As a step ahead with respect to this work, we perform an accurate performance evaluation by using real MQTT software implementation on a Linux-based testbed.

IV. Performance evaluation

A. Testbed setup

The testbed used to perform the validation over the proposed scenarios is composed of a set of Linux Virtual Machines (VMs) as shown in Figure 5. VM3 acts as a router to emulate a GEO Very Small Aperture Terminal (VSAT) SatCom connection. The considered MEC functionalities are implemented as a set of Docker-based applications in VM2. The Docker image deployed to enable the Edge MQTT Broker is based on the Mosquitto image[2]. Additional configurations are added to the network in VM2 to ensure that MQTT traffic can be transparently intercepted and locally processed by the Edge MQTT Broker without requiring sensor reconfigurations.

At the core side, the MQTT Broker is configured in VM4 and acts as the reference backhand on the cloud to collect and dispatch sensor data to the clients that can be located both at the Edge side (VM1) and at the Core side (VM5).

Clients are coded in C by using Open Source Mosquitto libraries. Sensors are modelled with scripts that generate data messages with uniform distribution and settable mean value for data size and inter-generation interval. Data messages from sensors then can be either directly included within MQTT messages by the SEs (Case 1) or encapsulated in MQTT messages by the Edge MQTT Broker before transmitting them to the Cloud (Cases 2 and 3).

In our tests, we define four different sensor categories: real-time, non-real-time, local-only, and global sensors. Real-time sensors generate time critical data (e.g., alarms) that should be delivered as fast as possible without additional processing. On the contrary, non-real-time sensors provide data whose validity is acceptable also in case of deferred or partial delivery (e.g., a non-critical temperature value). Both real-time and non-real-time sensor data can be of local-only type, i.e., they will be required only by Edge MQTT Clients, or global type, i.e., they will be required by both Edge and Core MQTT Clients.

Table I reports the reference values of the configuration variables. Such values are chosen to get realistic configurations. The basic reference is the IoT scenario reported in the Ericsson mobility report [42] with additional assumptions to implement the satellite scenario.

TABLE I: Configuration variables and reference values used in the tests

Parameter	**Variable**	**Reference Value**
Satellite uplink (Edge to Core) datarate	BW_{up}	128 Kbit/s
Satellite downlink (Core to Edge) datarate	BW_{down}	128 Kbit/s
Satellite two-ways average delay	RTT	580 ms
Number of IoT or MQTT Sensors - SE	N_S	5000
Number of real-time sensors	r	2000
Real-time "local-only" sensors	r_l	640
Real-time "global" sensors	r_g	1360
Number of non-real-time sensors	n	3000
Non-real-time "local-only" sensors	n_l	960
Non-real-time "global" sensors	n_r	2040
Mean message size	s_1	150 Bytes
Mean sensor inter-generation message time	t	240 s
Number of Edge MQTT Clients - UE (each interested in 10% of all sensor data)	N_{CE}	10
Number of Core MQTT Clients - UE (each interested in 10% of all sensor data except for "local-only" type)	N_{CC}	40

B. Assessing the proposed optimizations

The obvious expectation is that enabling the Edge MQTT Broker as an edge function will reduce the amount of traffic forwarded through the satellite link. Enabling the additional MEC functionalities should reduce even more the impact of sensor traffic on the satellite link, due to: 1) the reduction of the amount of data that have to reach the Core MQTT Broker and then go back to the Edge MQTT Clients owing to the edge MQTT brokering functions; 2) data compression and aggregation functions. Compression is very important for all real-time and very-low latency services where sensor data must be quickly available avoiding unnecessary data forwarding through the high-latency satellite link; aggregation is significant for delay-tolerant data. The interesting aspect is to quantify these improvements in a realistic environment.

[2]https://hub.docker.com/_/eclipse-mosquitto

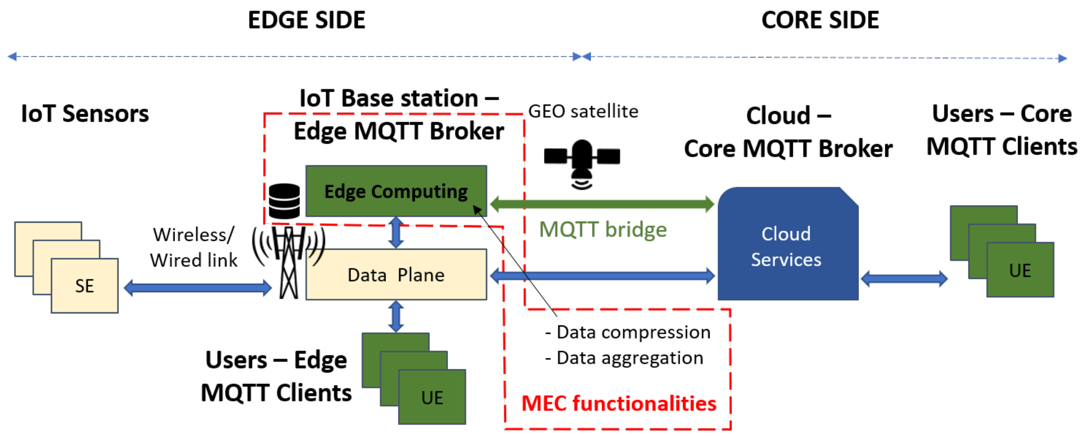

Fig. 4: IoT application level network configuration - Case 3: IoT non-MQTT Sensors, Edge MQTT Computing in the IoT Base station. Presence of an Edge MQTT Broker as in Case 2 + additional edge functionalities to improve satellite bandwidth utilization

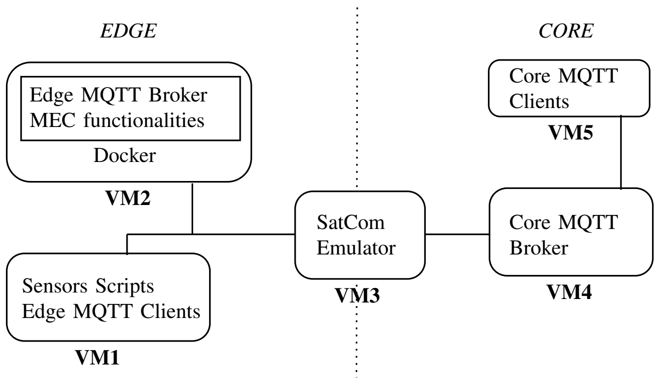

Fig. 5: Testbed configuration by using Linux-based VMs

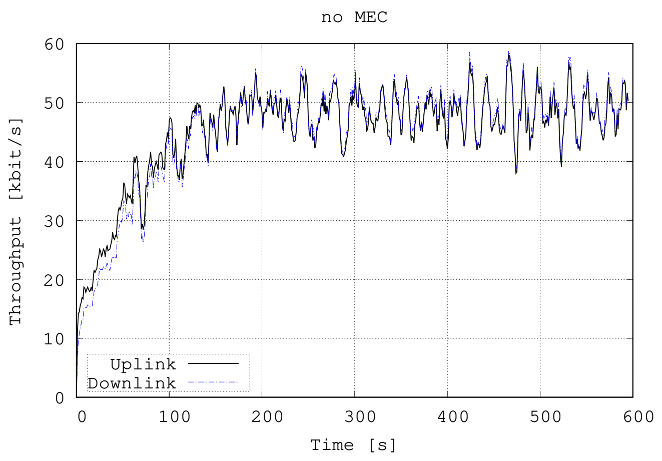

Fig. 6: Satellite link bandwidth utilization for Case 1: MEC is not applied

We have performed real-time tests whose duration is 10 minutes for each of the 3 cases defined in Section III and configured as detailed in Table I. The two considered output performance are:

- message delivery time: the time elapsed between the data message generation by the SE and its reception by the interested MQTT Client.
- satellite link bandwidth utilization, also called throughput: the amount of data that traverse the satellite uplink (from Edge side to Core side) and downlink (from Core side to Edge side).

In each test, the MQTT Clients gradually join the network, which leads to about 200 s of transient time before reaching the regime values of the satellite link bandwidth utilization. For each of the 3 cases, we show the overall satellite link bandwidth utilization and the distribution of the message delivery separate for Edge and Core MQTT Clients.

The first test, referred to Case 1, allows introducing a benchmark reference for the proposed application. The satellite link bandwidth utilization is reported in Figure 6.

We can see an almost symmetric use of the channel since all sensor data must reach the only MQTT Broker available in the network (the Core MQTT Broker at the Core side) and go back to the edge clients. The values for the average throughput after the initial ramp time are 47.9 kbit/s for the uplink and 48.6 kbit/s for the downlink. The throughput over time provides further insights on the traffic pattern: after the gradual entry of the sensors, we can see wide variations that must be taken into account to design and dimension real systems. Even if the sensors transmit at random times, a non-flat channel utilization is obtained.

Figures 7(a) and 7(b) shows the delivery time distribution of data requested by the Core MQTT Clients and by the Edge MQTT Clients, respectively.

Looking at these figures, it is possible to appreciate the impact of the satellite link on the time required to get data from the MQTT Broker without the help of any MEC functionality at the edge. The mean value of the delivery time is 325 ms from Figure 7(a) and 868 ms from Figure 7(b) where a satellite double hop is always required to get data.

When the local processing of MQTT messages is enabled at the Edge side through dedicated MEC functionalities (Case 2), the first aspect worth noting is the significant reduction of

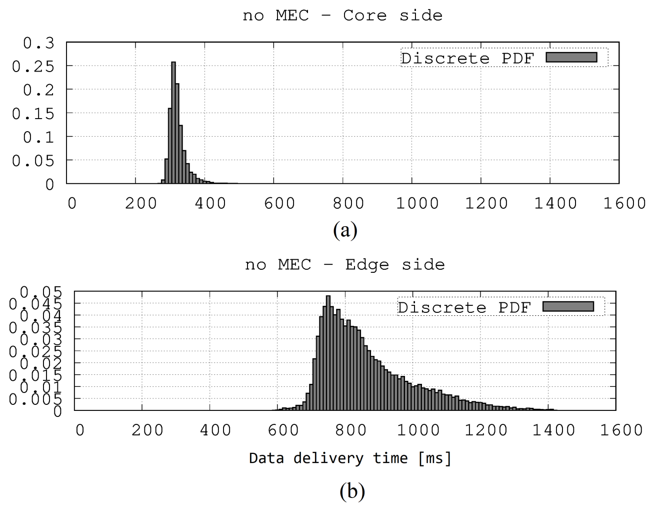

Fig. 7: Delivery time distribution of data destined to the Edge and Core MQTT Clients for Case 1

the satellite link utilization reported in Figure 8.

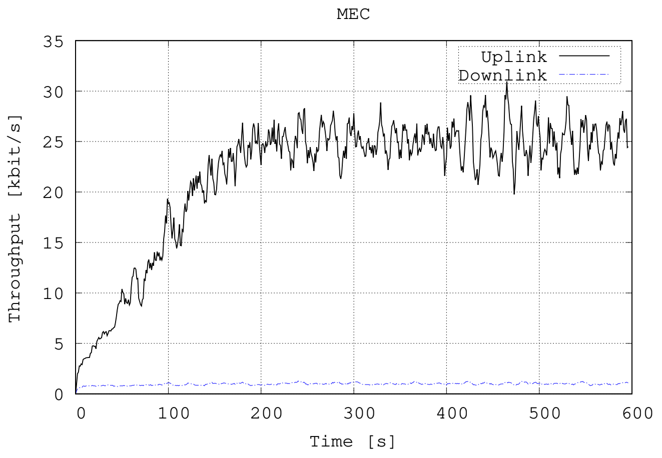

Fig. 8: Satellite link bandwidth utilization for Case 2: MEC is applied introducing an Edge MQTT Broker to handle local data

The downlink is now almost unused since local sensor data are handled by the Edge MQTT Broker without the need to access the Core MQTT Broker. The overall value of about 1 kbit/s is related to the flow of ACK packets destined to the sensors. However, concerning the uplink, the reduction is not proportional to the reduction of the amount of transmitted sensor data. The uplink bandwidth is reduced to about 50% (from 47.9 to 24.6 Kbit/s) while the number of sensors is reduced to 32% (from 5000 to 3400, which are both real-time and non-real-time "global" sensors, since the data generated by the "local-only" sensors do not need to traverse the satellite link but are received and stored only by the Edge MQTT Broker). This additional reduction is due to the implementation of the MQTT bridge by Mosquitto, and in particular due to the Edge MQTT Broker which groups more data messages in bulk before sending them to the Core MQTT Broker through the MQTT bridge. The confirmation of this behaviour is given by the data delivery time distribution shown in Figure 9.

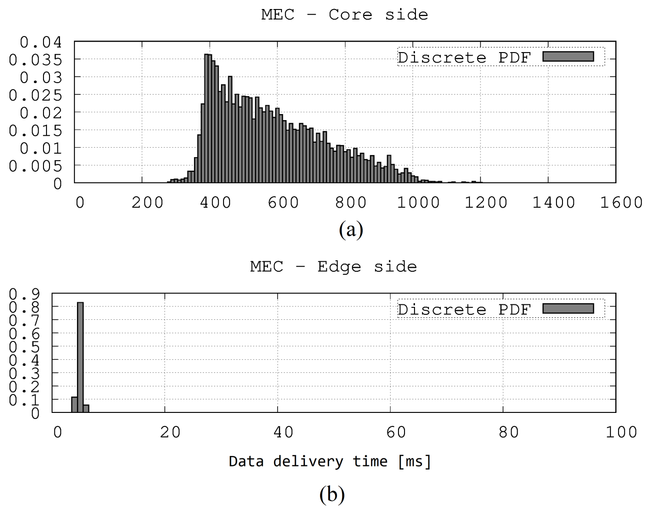

Fig. 9: Delivery time distribution of data destined to the Edge and Core MQTT Clients for Case 2

In particular, for the Core MQTT Clients (Figure 9(a)), the distribution is much shifted towards higher values (with an average of 587 ms). This happened because sensor messages are queued and aggregated with other messages at the socket/TCP level. On the other hand, data destined to Edge MQTT Clients (Figure 9(b)) are now available locally, significantly reducing the data delivery time to a few ms and providing the main desired benefit of MEC at the edge.

This second case shows how bandwidth reduction can be achieved by adding an MQTT MEC function without specific optimizations, at the cost of slightly increasing the Core side data delivery time.

Since some sort of socket-level aggregation was already performed in Case 2, enabling also specific data aggregation and compression mechanisms at the Edge side (Case 3) leads to marginal advantages, lower than what initially expected. The obtained results are shown in Figures 10 and 11.

In this third case, the Edge MQTT Broker is aware of the real-time or non-real-time nature of the global sensor data, as well as of the global or local-only scope of the messages. In this way, it is able to perform selective additional processing. In detail, the data aggregation functionality involves the aggregation of all non-real-time data messages to be sent through the MQTT bridge, received within a time window of fixed duration (1 s in our tests), in one single message, reducing the overall protocol header overhead. The data compression functionality includes: i) discharge of duplicate values: if a sensor generates the same data multiple times, such as the same temperature value, the data is just sent once, i.e. only the data variations are sent instead of always sending each data; ii) data reformatting: data message format is changed from text to binary in order to add some lossless compression. These functions, co-located with the Edge MQTT Broker, are so able to reduce redundant information and aggregate sensor data in a much more compact form by using larger TCP packets. These additional actions lead to a further satellite uplink utilization

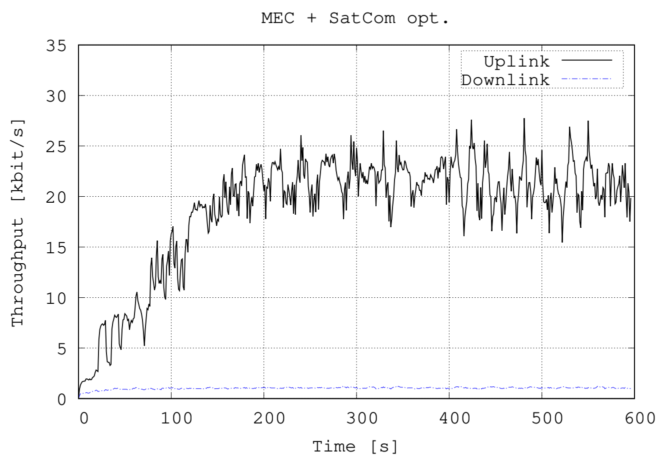

Fig. 10: Satellite link bandwidth utilization for Case 3: MEC is applied, including also data compression to reduce satellite traffic

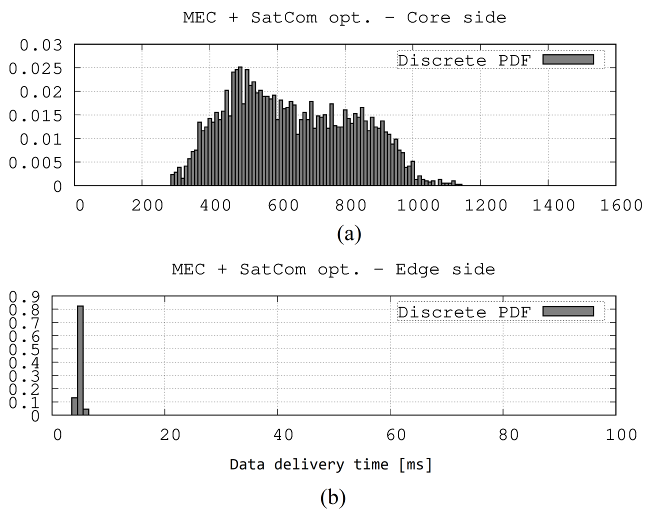

Fig. 11: Delivery time distribution of data destined to the Edge and Core MQTT Clients for Case 3

reduction of 12%, with an average value of 21.1 kbit/s. At the same time, Core side data delivery time is not so much affected, with an average value increased to 647 ms.

C. Further testing configurations

We performed additional tests to determine the effectiveness of the MEC architecture considering alternative configurations. These configurations have been designed with the aim of presenting some limit/stress cases, in which we want to show the scalability of the service, and the validity of the approach when the number of local-only sensors or clients is reduced. Furthermore, even when local data is managed at the edge and distributed only locally, processing delays and enqueuing times for the MEC processing and MQTT are still present, and are interesting for a large-scale deployment of the proposed system in real networks.

The following parameters have been changed, keeping all the other parameters set as reported in Table I:

1) N_{CC} = 100 (higher number of Core MQTT Clients)
2) N_{CE} = 0 (no Edge MQTT Clients)
3) N_S = 50000 (10x sensors, keeping the same percentage of real-time and non-real-time, global and local-only sensors defined in Table I)
4) $r_l = n_l$ = 0 (no local-only sensors)

Table II includes the obtained mean results of the additional tests for each of the proposed MEC configurations. The mean results of the tests already described in Section IV-B, labelled as *default*, have been included too for comparison.

Concerning the increase of N_{CC}, there is no significant change in the results compared to the default configuration due to the publisher-subscriber nature of the MQTT protocol. The amount of sensor data collected at the Core MQTT Broker through the satellite link does not depend on the number of Core MQTT Clients. Therefore, the results of this configuration are not included in Table II.

Without Edge MQTT Clients interested in sensor data (N_{CE} = 0 and, consequently, no Edge side data delivery time results), there is a significant reduction of the satellite downlink bandwidth utilization for Case 1. Since there is no MEC processing, the absence of Edge MQTT Clients reduces the amount of traffic in the core to edge direction since there is no more requests. For Cases 2 and 3, there is no significant difference in the satellite link bandwidth utilization since all traffic at the edge is handled locally, regardless of the number of interested Edge MQTT Clients. Besides, in Cases 2 add 3, the amount of traffic delivered through the MQTT bridge from the edge to the core side is the same. For this reason, also Core side data delivery time is not affected.

In the second additional test, we introduce a high degree of congestion by increasing the number of sensors by ten times (N_S = 50000). The uplink becomes quickly congested in all cases, but the introduction of MEC provides a substantial difference. Without MEC, also the satellite downlink is highly occupied, with consequent very high mean data delivery time values both at core and edge sides. Several sensors suffered also disconnections due to the excessive packet loss and TCP retransmissions due to RTO (retransmission timeout) expiration. As soon as MEC is enabled, the core side performance is guaranteed, obtaining an Edge side data delivery time mean value comparable with the default configuration. Case 3 shows a further better performance of the Core side data delivery time. This specific configuration allows assessing the reaction of the system in case of congestion: the increase of sensors from 5000 to 50000 does not lead to a proportional increase of the data delivery time, which instead increases of about 100 times. This conclusion is not applicable, as already discussed, for the Edge side data delivery time with MEC enabled, which is not affected.

Finally, without local-only real-time and non-real-time sensors ($r_l = n_l$ = 0), we can observe a proportional reduction of the satellite link bandwidth utilization in Case 1. Concerning Case 2 and 3, there are no significant changes since the local-only traffic flows do not traverse the satellite link. This test allows in particular to confirm the effectiveness of data local management.

TABLE II: Mean results from the additional performed tests

Variable configuration	**Scenario**	**Uplink bandwidth utilization [Kbit/s]**	**Downlink bandwidth utilization [Kbit/s]**	**Core side data delivery time [ms]**	**Edge side data delivery time [ms]**
Default	Case 1	47.9	48.6	325	868
	Case 2	24.6	0.6	587	4.9
	Case 3	21.1	0.7	647	4.9
N_{CE} = 0	Case 1	46.6	12.6	296	n.a.
	Case 2	24.6	0.5	588	n.a.
	Case 3	21.4	0.6	648	n.a.
N_S = 50000	Case 1	128	93.1	56779	53661
	Case 2	128	3.2	71916	4.8
	Case 3	128	2.9	40547	4.8
$r_l = n_l$ = 0	Case 1	33.1	33.6	314	904
	Case 2	24.6	0.5	584	4.8
	Case 3	21.4	0.6	638	4.7

The execution of additional runs showed the main benefit of the MEC-based approach in several configurations. According to the specific applications and associated requirements in terms of satellite link utilization and latency, a service operator can decide which is the best architectural configuration depending on the numerical results shown in the paper.

V. Conclusions

The huge amount of raw IoT data could stress the limited and expensive satellite communication resources which have to be carefully managed. Equipping the edge of the network with local processing MEC functionalities, such as data compression and aggregation, allow improving end-to-end performance and QoS in communications when satellite links are exploited.

This paper has analysed a Satellite communication-based MEC network architecture where a GEO satellite acts as a backhaul node between an IoT sensor network and the Internet. A distributed MQTT framework is defined where local instances of MQTT Brokers and data aggregation and compression techniques are deployed as Virtual Functions at the edge of the network. Performance evaluation is carried out through an ad-hoc designed testbed comparing the obtained performance with three different network configurations in terms of the amount of data that need to traverse the satellite link and data delivery time. The employment of MEC functionalities allows obtaining a significant reduction of the satellite link bandwidth utilization and of the achievable data delivery time in most of the considered cases. The use of data aggregation techniques leads to a higher delivery time in some cases which can affect delay-tolerant data, so implying a careful design.

Considering the rapid growth of LEO satellite communication systems, a second performance evaluation will be performed considering an additional reference scenario based on a LEO satellite constellation. This further analysis aims to show the impact of aspects such as satellite handover and possible link disruptions between the IoT base station and satellites on the shown performance in all the considered cases, so to highlight the improvement achievable with the proposed MEC functionalities also in case of LEO satellite backhaul links.

Acknowledgment

The results presented in this paper are part of the outcomes of the VIBeS project, ESA Contract N.: 4000122991/ 18/ UK/ ND [39]. Responsibility of the content resides with the authors.

References

[1] Cisco, "Cisco Annual Internet Report (2018-2023)," White paper, 2020.

[2] J. Liu, Y. Shi, Z. M. Fadlullah, and N. Kato, "Space-Air-Ground Integrated Network: A Survey," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 4, pp. 2714–2741, 2018.

[3] S. Cioni, R. De Gaudenzi, O. D. R. Herrero, and N. Girault, "On the Satellite Role in the Era of 5G Massive Machine Type Communications," *IEEE Network*, vol. 32, no. 5, pp. 54–61, 2018.

[4] N. Kato, Z. M. Fadlullah, F. Tang, B. Mao, S. Tani, A. Okamura, and J. Liu, "Optimizing Space-Air-Ground Integrated Networks by Artificial Intelligence," *IEEE Wireless Communications*, vol. 26, no. 4, pp. 140–147, 2019.

[5] F. Chiti, R. Fantacci, and L. Pierucci, "Energy Efficient Communications for Reliable IoT Multicast 5G/Satellite Services," *Future Internet*, vol. 11, no. 8, pp. 164–176, 2019.

[6] M. Bacco, F. Davoli, G. Giambene, A. Gotta, M. Luglio, M. Marchese, F. Patrone, and C. Roseti, "Networking Challenges for Non-Terrestrial Networks Exploitation in 5G," in *IEEE 5G World Forum (5GWF)*, 2019, pp. 623–628.

[7] L.-A. Turner and H. Jahankhani, "An Investigation into an Approach to Updating the Governance of Satellite Communications to Enhance Cyber Security," *Cybersecurity, Privacy and Freedom Protection in the Connected World*, pp. 23–33, 2021.

[8] M. Lin, Q. Huang, T. de Cola, J.-B. Wang, J. Wang, M. Guizani, and J.-Y. Wang, "Integrated 5G-satellite networks: A perspective on physical layer reliability and security," *IEEE Wireless Communications*, vol. 27, no. 6, pp. 152–159, 2020.

[9] Q.-V. Pham, F. Fang, V. N. Ha, M. J. Piran, M. Le, L. B. Le, W.-J. Hwang, and Z. Ding, "A Survey of Multi-Access Edge Computing in 5G and Beyond: Fundamentals, Technology Integration, and State-of-the-Art," *IEEE Access*, vol. 8, pp. 116 974–117 017, 2020.

[10] S. Kekki, W. Featherstone, Y. Fang, P. Kuure, A. Li, A. Ranjan, D. Purkayastha, F. Jiangping *et al.*, "MEC in 5G networks," *ETSI White Paper No. 28*, 2018.

[11] P. Porambage, J. Okwuibe, M. Liyanage, M. Ylianttila, and T. Taleb, "Survey on Multi-Access Edge Computing for Internet of Things Realization," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 4, pp. 2961–2991, 2018.

[12] T. K. Rodrigues, K. Suto, and N. Kato, "Edge Cloud Server Deployment with Transmission Power Control through Machine Learning for 6G Internet of Things," *IEEE Transactions on Emerging Topics in Computing*, 2019.

[13] R. Xie, Q. Tang, Q. Wang, X. Liu, F. R. Yu, and T. Huang, "Satellite-Terrestrial Integrated Edge Computing Networks: Architecture, Challenges, and Open Issues," *IEEE Network*, vol. 34, no. 3, pp. 224–231, 2020.

[14] L. Yan, S. Cao, Y. Gong, H. Han, J. Wei, Y. Zhao, and S. Yang, "SatEC: A 5G Satellite Edge Computing Framework Based on Microservice Architecture," *Sensors*, vol. 19, no. 4, p. 831, 2019.

[15] B. Denby and B. Lucia, "Orbital Edge Computing: Nanosatellite Constellations as a New Class of Computer System," in 25th *International Conference on Architectural Support for Programming Languages and Operating Systems*, 2020, pp. 939–954.

[16] Z. Zhang, W. Zhang, and F.-H. Tseng, "Satellite Mobile Edge Computing: Improving QoS of High-Speed Satellite-Terrestrial Networks Using Edge Computing Techniques," *IEEE Network*, vol. 33, no. 1, pp. 70–76, 2019.

[17] T. De Cola, M. Marchese, M. Mongelli, and F. Patrone, "A Unified Optimisation Framework for QoS Management and Congestion Control in VHTS Systems," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 10, pp. 11 619–11 631, 2020.

[18] J. Wei, J. Han, and S. Cao, "Satellite IoT Edge Intelligent Computing: A Research on Architecture," *Electronics*, vol. 8, no. 11, p. 1247, 2019.

[19] Y. Wang, J. Yang, X. Guo, and Z. Qu, "Satellite Edge Computing for the Internet of Things in Aerospace," *Sensors*, vol. 19, no. 20, p. 4375, 2019.

[20] D. Chen, C. Yang, P. Gong, L. Chang, J. Shao, Q. Ni, A. Anpalagan, and M. Guizani, "Resource Cube: Multi-Virtual Resource Management for Integrated Satellite-Terrestrial Industrial IoT Networks," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 10, pp. 11 963–11 974, 2020.

[21] N. Cheng, F. Lyu, W. Quan, C. Zhou, H. He, W. Shi, and X. Shen, "Space/Aerial-Assisted Computing Offloading for IoT Applications: A Learning-Based Approach," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 5, pp. 1117–1129, 2019.

[22] G. Cui, X. Li, L. Xu, and W. Wang, "Latency and Energy Optimization for MEC Enhanced SAT-IoT Networks," *IEEE Access*, vol. 8, pp. 55 915–55 926, 2020.

[23] Y. Cao, Y. Shi, J. Liu, and N. Kato, "Optimal Satellite Gateway Placement in Space-Ground Integrated Network for Latency Minimization with Reliability Guarantee," *IEEE Wireless Communications Letters*, vol. 7, no. 2, pp. 174–177, 2017.

[24] 3GPP, "Solutions for NR to support non-terrestrial networks (NTN)," Technical Report 38.821.

[25] ESA ARTES AT Project, "NB-IoT4Space - 3GPP Narrow-Band Internet-of-Things (NB-IoT) User Sensor Integration into Satellite," 2020. [Online]. Available: https://artes.esa.int/projects/nbiot4space

[26] A. Guidotti, A. Vanelli-Coralli, M. Conti, S. Andrenacci, S. Chatzinotas, N. Maturo, B. Evans, A. Awoseyila, A. Ugolini, T. Foggi, L. Gaudio, N. Alagha, and S. Cioni, "Architectures and Key Technical Challenges for 5G Systems Incorporating Satellites," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 3, pp. 2624–2639, 2019.

[27] M. Centenaro, C. E. Costa, F. Granelli, C. Sacchi, and L. Vangelista, "A survey on technologies, standards and open challenges in satellite IoT," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 3, pp. 1693–1720, 2021.

[28] R. Giuliano, F. Mazzenga, and A. Vizzarri, "Satellite-Based Capillary 5G-mMTC Networks for Environmental Applications," *IEEE Aerospace and Electronic Systems Magazine*, vol. 34, no. 10, pp. 40–48, 2019.

[29] M. Höyhtyä, T. Ojanperä, J. Mäkelä, S. Ruponen, and P. Järvensivu, "Integrated 5G Satellite-Terrestrial Systems: Use Cases for Road Safety and Autonomous Ships," in 23rd *Ka and Broadband Communications Conference*, 2017, pp. 16–19.

[30] J. V. Y. Martnez, A. F. Skarmeta, M. A. Zamora-Izquierdo, and A. P. Ramallo-Gonzlez, "IoT-based data management for Smart Agriculture," in 2nd *International Conference on Embedded & Distributed Systems (EDiS)*. IEEE, 2020, pp. 41–46.

[31] K. Sohraby, D. Minoli, B. Occhiogrosso, and W. Wang, "A review of wireless and satellite-based M2M/IoT services in support of smart grids," *Mobile Networks and Applications*, vol. 23, no. 4, pp. 881–895, 2018.

[32] T. Wei, W. Feng, Y. Chen, C.-X. Wang, N. Ge, and J. Lu, "Hybrid satellite-terrestrial communication networks for the maritime Internet of Things: Key technologies, opportunities, and challenges," *IEEE Internet of things journal*, vol. 8, no. 11, pp. 8910–8934, 2021.

[33] OASIS, "Message Queuing Telemetry Transport (MQTT) version 3.1.1," http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.pdf, 2015.

[34] Z. Shelby, K. Hartke, C. Bormann, and B. Frank, "The Constrained Application Protocol (CoAP)," RFC 7252, 2014.

[35] R. Soua, M. R. Palattella, and T. Engel, "IoT Application Protocols Optimisation for Future Integrated M2M-Satellite Networks," in *IEEE Global Information Infrastructure and Networking Symposium (GIIS)*, 2018, pp. 1–5.

[36] M. Bacco, P. Cassarà, M. Colucci, and A. Gotta, "Modeling Reliable M2M/IoT Traffic over Random Access Satellite Links in Non-Saturated Conditions," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 5, pp. 1042–1051, 2018.

[37] F. Wang, D. Jiang, S. Qi, C. Qiao, and H. Song, "Fine-Grained Resource Management for Edge Computing Satellite Networks," in *IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2019, pp. 1–6.

[38] C. Qiu, H. Yao, F. R. Yu, F. Xu, and C. Zhao, "Deep Q-Learning Aided Networking, Caching, and Computing Resources Allocation in Software-Defined Satellite-Terrestrial Networks," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 6, pp. 5871–5883, 2019.

[39] ESA ARTES AT Project, "VIBES - Implementation of VIrtualised Network Functions (VNFs) for Broadband Satellite Networks," 2018. [Online]. Available: https://artes.esa.int/projects/vibes

[40] ——, "M2MSAT - Demonstrator of Light-Weight Application and Transport Protocols For Future M2M Applications," 2017. [Online]. Available: https://artes.esa.int/projects/m2msat

[41] M. R. Palattella, R. Soua, A. Stemper, and T. Engel, "Aggregation of MQTT Topics over Integrated Satellite-Terrestrial Networks," *ACM SIGMETRICS Performance Evaluation Review*, vol. 46, no. 3, pp. 96–97, 2019.

[42] Ericsson, "Massive IoT in the city," Mobility Report, 2016. [Online]. Available: https://www.ericsson.com/en/mobility-report/articles/massive-iot-in-the-city