Efficiency analysis of Cloud GNSS signal processing for IoT applications

V. Lucas-Sabola[(1)], G. Seco-Granados[(1)], J.A. López-Salcedo[(1)],
J.A. García-Molina[(2)(3)], and M. Crisci[(2)]
(1) Department of Telecommunication and Systems Engineering, IEEC-CERES, Universitat Autònoma de Barcelona (UAB), Spain
(2) *European Space Agency (ESA/ESTEC), The Netherlands*
(3) *HE-Space, The Netherlands*

BIOGRAPHY (IES)

Vicente Lucas-Sabola received the B.Sc. in telecommunication systems engineering in 2015 and the M.Sc. in telecommunication engineering in 2017, both from UAB. Since 2015 he is involved in the development of a Cloud GNSS receiver, a project funded by ESA. Since 2017 he is pursuing the Ph.D. at the Signal Processing for Communications and Navigation (SPCOMNAV) research group, dealing with topics related to cloud GNSS signal processing.

Gonzalo Seco-Granados holds a Ph.D. degree from Universitat Politècnica de Catalunya (UPC) and an MBA from IESE, Universidad de Navarra. Since 2006, he is an associate professor at the Department of Telecommunication and Systems Engineering, Universitat Autònoma de Barcelona (UAB) and head of the Signal Processing for Communications and Navigation (SPCOMNAV) research group.

José A. López-Salcedo received the M.Sc. and Ph.D. degrees in Telecommunication Engineering in 2001 and 2007 from Universitat Politècnica de Catalunya (UPC). He is associate professor at the Department of Telecommunication and Systems Engineering, Universitat Autònoma de Barcelona (UAB), and member of the Signal Processing for Communications and Navigation (SPCOMNAV) research group.

J. A. Garcia-Molina is a Radio Navigation engineer at ESA/ESTEC in Noordwijk, The Netherlands, where he leads several R&D projects and internal research activities on GNSS receiver technology and signal processing techniques for ground and space applications in the context of different ESA programs (including Galileo). His main research interests include signal processing and estimation theory, GNSS/Galileo receivers and signals, hybrid and cooperative/collaborative positioning, array signal processing, identification and localization of jammers/spoofers, SDR receivers, and cloud positioning techniques and applications.

Massimo Crisci is the head of Radio Navigation Systems and Techniques Section at the ESA. He is the technical domain responsible for the field of radionavigation and the head of a team of engineers providing radionavigation expert support to the various ESA programs (EGNOS and Galileo included). He holds a Ph.D. in automatics and operations research from the University of Bologna and a Master's degree in electronics engineering from University of Ferrara.

ABSTRACT

The advent of services and applications that rely on Global Navigation Satellite System (GNSS) to procure reliable, accurate and ubiquitous positioning and timing information is forcing conventional GNSS sensors' computational resources to its limits. In addition, such information is expected to be provided at low energy consumption and at a low cost. These features have led to the implementation of novel techniques and architectures such as the Cloud GNSS receiver. In this architecture, the migration of the computational tasks (e.g. GNSS signal processing) from the device to a cloud server where high-scalable and high-performance computing resources are available is carried out with the aim of reducing the energy required by the sensor or device in order to obtain a position fix. This paper analyzes the energy efficiency and the accuracy performance of the Cloud GNSS receiver. To prove its feasibility, the energy consumption of a conventional GNSS sensor is analyzed, confirming the GNSS module as one of the most consuming components together with the transmission Radio Frequency (RF) front-end. Then, the energy required by the alternative cloud GNSS sensor is addressed and compared with conventional GNSS sensors. The obtained results reveal the feasibility of cloud architectures for GNSS signal processing in energetic and positioning

accuracy terms. Costs derived from data transmission and the use of cloud resources are also assessed within the framework of an Internet-of-Things (IoT) application.

INTRODUCTION

During the last years, the number of applications and services that require of position measurements, usually provided by mean of Global Navigation Satellite System (GNSS) technologies, has increased significantly. Such applications must offer reliable, accurate and ubiquitous positioning without compromising the computational resources required for the implementation of the computational tasks. Nevertheless, devices or sensors are tending to be low-powered in order to achieve a larger autonomy, with limited computational resources and low cost. That is the case of Internet of Things (IoT) applications, where physical objects (e.g. vehicles, buildings, furniture, etc.) are connected within a network with a unique IP address, with the purpose of providing some information (e.g. positioning). Such is the impact of IoT that its market is estimated to generate from €1 trillion to 7€ trillion in revenues in 2020, depending on the definition of IoT, being manufacturing, supply chain and Smart Cities the three dominant markets [1]. Following this tide, big data applications are also converging in the use GNSS technologies for the collection of GNSS data by the establishment of GNSS data centers in different locations. Then, the processing of the collected GNSS data provides information that later can be used for different kind of applications [2]: integrity and ionosphere monitoring, interference detection, etc.

Positioning information is typically procured by means of GNSS, non-GNSS positioning technologies such as 3G, Ultra-Wide-Band (UWB), Wireless Local Area network (WLAN), Inertial Navigation Systems (INS), or a combination (hybrid positioning) of GNSS and non-GNSS positioning technologies. Focusing on GNSS, conventional sensors or devices include a mass-market GNSS module to provide the sensors' localization. In this context, on-site computational tasks are carried out in the sensor itself (GNSS module), thus requiring of computational resources that will consume a significant amount of energy [3],[4], and hence compromising the sensors' lifetime. However, sensors tend to be miniaturized, low cost, with low energy consumption and limited computational power. That is to say, sensors are expected to be cheap and have a large lifetime without compromising its performance. Even though Commercial Off-The-Shelf (COTS) manufacturers are focusing their efforts on reducing the energy consumption of the GNSS module, still the sensors autonomy is compromised. To tackle these sensor constraints, a cloud approach has been proposed in the literature to offload the energy consumption of the sensor [5]. In this paper, the workflow shift proposed in the Cloud GNSS receiver [2] is used, where the heavy computational tasks of the sensor (i.e. GNSS signal processing tasks) are migrated to a cloud server (Cloud GNSS).

The Cloud GNSS receiver structure can by divided in three main elements: front-end, User Equipment (UE), and back-end. The front-end is the interface with which a user or machine, Human-to-Machine (H2M) and Machine-to-Machine (M2M) respectively, interacts with the cloud service or back-end. The UE is the technology to be held by the device or sensor in order to gather any kind of information or data and transfer it to the back-end through the front-end. In this paper, a cloud GNSS sensor is proposed to execute the tasks required by the UE. In this sense, the cloud GNSS sensor would be in charge of just capturing the GNSS signal, digitize it and send it to the cloud server. Therefore, as the computational tasks are remotely performed, it is expected that the energy consumed by the sensor will decrease, thus enhancing its energy lifetime. Not only that, the cost of the sensor is also expected to be reduced as the GNSS module is replaced by just a Radio Frequency (RF) front-end and an Analog-to-Digital Converter (ADC). The back-end is the core of the cloud architecture and where the computational resources are placed. The Cloud GNSS receiver back-end is based on Elastic Cloud Computing (EC2) instances offered by Amazon Web Services (AWS). Instances are virtual machines with computing power (i.e. RAM, CPU, storage, etc.) that can be opened and closed depending on the user demands. To sum it up, in the Cloud GNSS receiver architecture, a UE captures and digitizes a GNSS signal and transfers it through the front-end to the back-end, where the computational resources supplied by EC2 instances will implement the corresponding signal processing techniques with a snapshot-based High-Sensitivity (HS) GNSS Software (Sw) Receiver (Rx) [6].

Nevertheless, the energy consumption and economic cost of accessing the cloud infrastructure are often left aside in most cloud processing studies, when they actually are two critical aspects that must be assessed for determining the feasibility of cloud signal processing. In fact, it is known that the transmission of data is one of the most energy consuming stages of a sensor, even higher than performing the computational tasks locally in the device itself depending on the application. In the Cloud GNSS receiver architecture, the size of the data to be transmitted is directly proportional to the signal length, the sampling frequency of the reception RF front-end and the quantization of the ADC. Therefore, a trade-off is found between the size of the data to be transmitted and the energy consumed by the sensor or device. In addition, the signal length is also related with the sensitivity of the GNSS receiver, as increasing the coherent or non-coherent integration time, and hence increasing the signal length to be gathered, allows the detection of weaker signals. Then, another trade-off is found between the sensitivity and the size of the packet (i.e. raw GNSS samples file) to be sent from the sensor to the cloud. Regarding to the economic cost

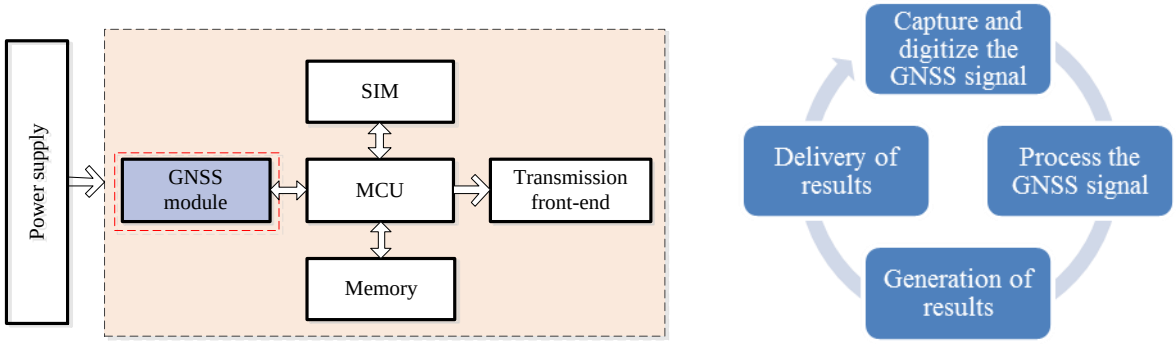

Figure 1. Architecture and workflow of a conventional GNSS sensor.

of Cloud signal processing, the use of computational resources allocated in cloud servers has an additional charge with respect to conventional GNSS sensors. If such cost is considerable, then the economic feasibility of using cloud servers (i.e. Amazon Web Services Elastic Cloud Computing instances) to perform GNSS signal processing tasks may be compromised.

In order to fill these gaps, this paper first addresses the energy consumption of both the conventional and cloud GNSS sensor and discusses whether the use of the Cloud GNSS receiver is an energetic efficient procedure. On the other hand, this paper also discusses the performance of the cloud GNSS sensor in terms of positioning accuracy, energy consumption and sensitivity. Next, in the context of an IoT application, the economic cost of hiring the cloud computing servers in order to implement the required GNSS signal processing tasks is addressed. Finally, in the last section conclusions are established in the last section.

CONVENTIONAL GNSS SENSOR

In typical sensor networks or devices aided for positioning applications or services, the sensors are in charge of retrieving their location to a remote monitoring center or station, where a decision is usually reached thanks to the provided information. If we want to reduce the energy consumption of such conventional GNSS sensors, first we have to study the consumed energy by its different components. In this section the architecture and workflow of a conventional GNSS sensor is initially discussed. Then, the energy consumption of each of its components is analyzed, allowing us to find the most consuming ones, which in the end will be targeted for reducing the entire sensors' energy consumption.

Architecture and workflow of a conventional GNSS sensor

The architecture of a conventional positioning sensor is composed by six main elements: power supply, GNSS module, MicroController Unit (MCU), Subscriber Identity Module (SIM), memory and transmission front-end (Figure 1). The power supply, a battery for instance, supplies energy to the sensor. Its duration will determine the lifetime of the sensor. The GNSS module, which usually includes a reception front-end and an ADC, is in charge of capturing the GNSS signal and processing it in order to obtain some measurement such as the Position, Velocity and Time (PVT) of the senor. The output of a GNSS module is typically a National Marine Electronics Association (NMEA) file that contains information regarding to the position of the sensor, visible satellites, measurements, pseudoranges, etc. The MCU is an integrated circuit that contains one or more Central Processing Units (CPUs), a small Random-Access Memory (RAM) and programmable memory, and it is the brain of the sensor. A SIM is used to connect the sensor with a mobile telephony network such as 3G or Long Term Evolution (LTE). The SIM may be replaced by other modules of different wireless telecommunication technologies (i.e. LoRa, SigFox), which may reduce the energy consumption during data transmission. A memory is used for storing the input and output data, such as the NMEA file generated by the GNSS module. There is a wide range of mass-market programmable memories, such as Ferroelectric RAM (FRAM) or Electrically Erasable Programmable Read-Only Memory (EEPROM). Even though FRAM offers a significant speed boost with respect to EEPROM memories (~1000x faster), usually EEPROM memories are used due to their low cost. If the size of the file to be stored is not large, the memory of the sensor may be removed as the MCU usually includes different low-sized programmable memories. Finally, the transmission module is in charge of transferring the data or information from the sensor to the monitoring station. This transmission can be based in many communication standards: Narrow-Band IoT (NB-IoT), LoRa, 3G, etc. The used standard is of paramount importance for reducing the energy required during data transmission, for instance with Low-Power Wide-Area Network (LPWAN) technologies, whose low power and bit rate may extend the battery life of the sensor.

Table 1. Current consumption of the conventional GNSS sensor components.

Component	Manufacturer	Model	Current consumption
GNSS module	u-blox	MAX-M8W	32 mA [7] (Acquisition)
GNSS module	u-blox	MAX-M8W	8.9 [7] (Tracking)
MCU	Texas Instruments	CC1310	1.9-2.5 mA [8]
Memory	Atmel	AT25M02	5 mA [9]
SIM	Infineon	SLM 76CF3601P	10 mA [10]
Transmission front-end	Texas Instruments	CC1310	11.2 mA [8]

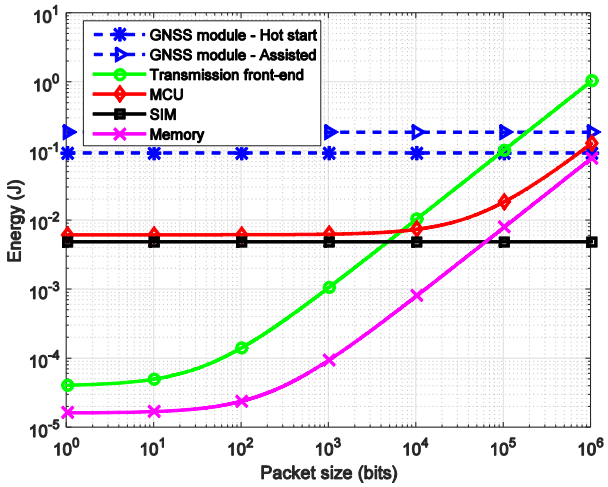

Figure 2. Expected energy consumption of the conventional GNSS sensor components with regard to the size of the packet to be transferred.

A conventional GNSS sensor workflow is mainly divided in four steps, as shown in Figure 1: first, the GNSS signal is captured by means of the RF front-end and digitized with an Analog-to-Digital Converter (ADC), both included in the GNSS module; second, the digitized GNSS signal is processed by the GNSS module; third, the GNSS module generates an output, usually an NMEA file; fourth, the results (NMEA) are delivered from the sensor to a remote control center, where such information is used with some objective (e.g. water or energy savings, asset management, etc.). So, in the conventional approach, sensors obtain their position (among other GNSS and localization information) and transfer the results to a remote server.

Energy consumption of a conventional GNSS sensor

In order to obtain the energy consumption of a conventional GNSS sensor, we first have to obtain the energy consumed by each of its components. The energy model of a conventional GNSS sensor can be defined as $E_{\mathrm{CS}} = E_{\mathrm{GNSS}} + E_{\mathrm{MCU}} + E_{\mathrm{Mem}} + E_{\mathrm{SIM}} + E_{\mathrm{tx}}$, where E_{CS}, E_{GNSS}, E_{MCU}, E_{Mem}, E_{SIM} and E_{tx} are the energy consumed by the conventional GNSS sensor, GNSS module, MCU, memory, SIM and transmission front-end, respectively Basically, the required energy by a component can be obtained by its current consumption in active mode, the amount of time it needs to be in active mode, and the supply voltage.

For this study case, the supply voltage is fixed to 3.3 V. The current consumption of each module has been obtained from datasheets of state of the art components, see Table 1. Regarding to current consumption, notice that the transmission front-end also consumes an additional amount of energy due to the release of power from the antenna, and the GNSS module current consumption may vary depending on the working conditions of the sensor (e.g. open, mixed or obstructed environment). On the other hand, the active time of the modules is obtained as follows: the MCU is active during all the time since the sensor is switched on and begins capturing the GNSS signal until the results (i.e. NMEA) are sent; the memory and transmission front-end active time are dependent on the packet size; the SIM is active since it connects with a base station and obtains the necessary keys to perform the telecommunication or transmission of data; and the GNSS module active time is equal to the Time-To-First-Fix (TTFF) plus the configured tracking time. The TTFF depends on the start mode the GNSS module has to perform when it is switched on. There are four different starts for a GNSS module: cold, warm, assisted and hot. In a cold start, all the possible frequency and code delays are searched and the ephemeris and broadcast time are decoded. In a warm start, only the broadcast time and ephemeris are decoded, as the frequency and code delays are hold as prior information. In a hot

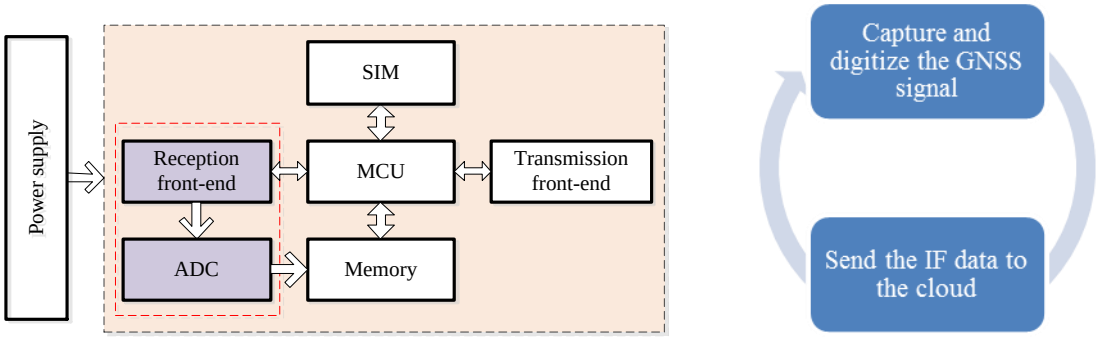

Figure 3. Architecture and workflow of the cloud GNSS sensor.

start, frequency and code delays, broadcast time, and ephemeris data are already known. Novel state of the art GNSS modules include assisted positioning, which allows downloading ephemeris and broadcast time information in order to achieve a faster TTFF. For assisted start, the time and energy consumption required for downloading GNSS assistance data (~1-3 KB) is also considered. In this paper, we have focused in the hot and assisted start only, as it is supposed that the sensors will be able to download assistance data from GNSS Data Centers (GDC) servers or reuse the prior assistance data to ease the computational workload of the GNSS receiver and work with shorter TTFFs.

During the scope of this work, the studied GNSS receiver is the u-blox MAX-M8X [7] that offers a good performance while maintaining a low energy consumption. The MAX-M8X can implement two different operating modes: continuous and power save [11]. Continuous mode begins with acquisition, the most power-consuming stage of the sensor, where visible satellites are searched and a position fix is performed. Afterwards, the obtained position fix is fed to the tracking engine and the acquisition engine is shut down to reduce the consumption. This process is restarted when new satellites signals are available. Power save mode is divided in two categories: cyclic tracking and ON/OFF operation. The former is suited for applications in which positions fixes are required in short periods of time (between 1 and 10 seconds). The latter is used for applications that require of position fixes in a larger period of time (minutes, hour or days). For this case study, an ON/OFF operation is used, as the period of time between position fixes is considered to be larger than 10 seconds. In ON/OFF mode, the GNSS receiver is shut down between position fixes with exception of the Battery Backed RAM (BBR) and the Real-Time Clock (RTC), thus achieving a minimum energy consumption between fixes. Therefore, in ON/OFF operation, when the receiver is turned on, the acquisition state is enabled until obtaining a first position fix. Afterwards, the receiver changes from acquisition to tracking state, where produces position fixes at a given rate during the configured time. Finally, after finishing the tracking state, the receiver is switched off during a given period until the next fix. The acquisition time of a GNSS module is equal to its TTFF, whereas the tracking time is configured by the user. The TTFF of the GNSS module studied in this work, a u-blox MAX-M8W, is equal to at least 1, 2, 29 seconds for hot, assisted and cold start, respectively [7] in benign environment conditions. Likewise, the tracking time typically ranges from 1 to 10 seconds. In the framework of this work the tracking time is configured to just 1 second in order to achieve lower energy consumption. Note that the acquisition time may vary depending on the working conditions of the receiver, being larger in harsh environments and hence increasing the energy consumption of the receiver. If the GNSS receiver is not able to obtain a valid position fix during the acquisition time, tracking cannot be performed, and then, the receiver must re-start the acquisition state, thus increasing the energy required to obtain a position fix as the TTFF is increased, and hence the active time of the whole sensor. Furthermore, the current consumption of the GNSS module varies depending on the working conditions. However, in this case study is considered that the receiver is working in an open environment (suitable working conditions) and hence the current consumption is assumed to be uniform.

The expected energy consumption of state of the art components of a conventional GNSS sensor with respect to the size of the packet to be sent from the sensor to a remote server is provided in Figure 2. It can be seen how the GNSS module and the SIM are independent of the packet size as their time in active mode is an approximately uniform value, whereas the MCU, memory and transmission front-end energy consumption is directly dependent with the packet size. For packets of small size, the most energy-consuming component is the GNSS module even with hot or assisted start: from roughly an order of magnitude up to almost four orders of magnitude in comparison with other components. As the packet size increases, the energy required by the memory, MCU, and above all, the transmission front-end, becomes dominant on the sensors' energy budget and, indeed, outnumbers the energy consumption of the GNSS module. Nonetheless, the size of the output from the GNSS module, namely the NMEA, is in the range of values from 1 KB (~10^4 bits) to 5 KB (~$4{\cdot}10^4$ bits), which is why the GNSS module typically is the most consuming component of a conventional positioning sensor instead of the transmission front-end. Therefore, if the energy consumption of a conventional GNSS sensor aided for positioning purposes must be reduced, Figure 2 reveals that the components to be targeted are the GNSS receiver and the transmission front-end.

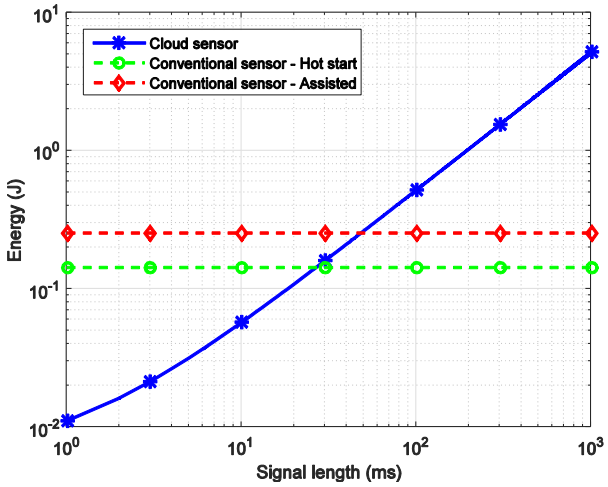

Figure 4. Expected energy consumption of the cloud GNSS sensor against the conventional GNSS sensor with regard to the captured signal length.

CLOUD GNSS SENSOR

Studying the energy consumed by the components of a conventional GNSS sensor reveals that the GNSS module typically is the most consuming element of the whole sensor (see previous section). In this context, the Cloud GNSS receiver is proposed in [2] to overcome the energy consumption hurdle imposed by GNSS modules. In the Cloud GNSS, the signal processing tasks conventionally carried out by the GNSS module are performed in a cloud server instead, thus easing the computational resources required by the sensor and, supposedly, diminishing its energy consumption. This is achieved by sending a raw GNSS samples file (IF data) from the sensor (UE) to a cloud server (the Cloud GNSS back-end), where a snapshot-based HS GNSS SwRX will first read the file and then apply the proper signal processing techniques in order to obtain the PVT or any other measurement as an output. In this section, the architecture and workflow of the proposed cloud GNSS sensor (the proposed UE) and its energy-efficiency with regard to conventional approaches is discussed, together with the position accuracy performance.

Architecture and workflow of the cloud GNSS sensor

The architecture of the cloud GNSS sensor is depicted in Figure 3, where seven different modules are shown: power supply, reception front-end, ADC, SIM, MCU, memory and transmission front-end. One of the main advantages of the Cloud GNSS approach is the capability of getting rid of one of the most expensive and most energy-consuming components of a conventional GNSS sensor: the GNSS module. In its place, just a reception front-end suitable for capturing GNSS signals and an ADC to digitize the analog signal is included. Thanks to this, the energy consumed by the sensor is expected to be reduced as no heavy computational tasks are performed in the sensor itself. Likewise the SIM, MCU, memory and transmission front-end modules still perform the same tasks than in a conventional GNSS sensor.

The workflow of the cloud GNSS sensor is divided in two steps (Figure 3): first, the reception front-end captures the desired GNSS signal and digitizes it by means of an ADC; then, the generated IF data file is sent from the sensor to a cloud server, where it will be read and processed by high-scalable and high-performance instances (machines that include computational power). The cloud GNSS sensor generates an IF data file that contains the digital samples of the captured GNSS signal, instead of generating a NMEA file. As it will be seen in the following sub-sections, the size of the IF data or raw GNSS samples file to be transferred will be of paramount importance in terms of energy and thus cannot be randomly large.

Energy consumption comparison between the cloud GNSS sensor and conventional approaches

One of the first questions that arise when introducing the Cloud GNSS and in particular, the cloud GNSS sensor is its energetic feasibility with regard to conventional approaches [5]. As we have previously seen, instead of locally processing the IF data, the cloud GNSS sensor only has to transfer it to a cloud server. This process is expected to be more energy efficient than the conventional approach as the GNSS signal processing tasks are migrated from the sensor to the cloud server. Nevertheless, in Figure 2 is demonstrated that the transmission of data may require an extensive amount of energy for large packet sizes.

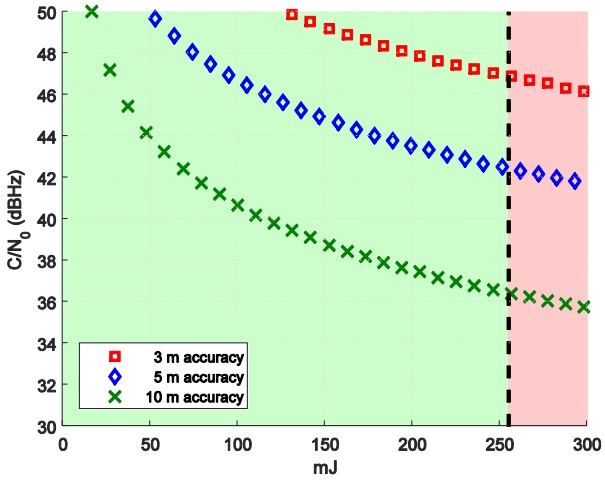

Figure 5. Expected energy consumption and accuracy performance of the cloud GNSS sensor under different C/N0 conditions.

Therefore, if the energy consumed by the cloud GNSS sensor must be remained as low as possible, then the size of the packet must be limited without jeopardizing performance of the position accuracy or any other measurement.

Estimates of the energy consumption of a cloud GNSS sensor can be derived from its energetic model: $E_{\mathrm{CGS}} = E_{\mathrm{rx}} + E_{\mathrm{ADC}} + E_{\mathrm{MCU}} + E_{\mathrm{Mem}} + E_{\mathrm{SIM}} + E_{\mathrm{tx}}$, where E_{CGS}, E_{rx}, E_{ADC} , E_{MCU}, E_{Mem}, E_{SIM} and E_{tx} are the energy consumed by the cloud GNSS sensor, reception front-end, ADC, MCU, memory, SIM and transmission front-end, respectively. The energy required by the MCU, memory, SIM and transmission front-end modules has been previously addressed based on state of the art components (Table 1). On the other hand, the reception front-end and ADC must be carefully chosen to avoid compromising the energy-efficiency of the cloud GNSS sensor. In this sense, the will not only contribute on their own energy consumption, but also in the energy consumed by the other components of the cloud GNSS sensor as they are directly dependent with the packet size. Indeed, the size of the packet that will be generated by the cloud GNSS sensor can be expressed as $L = t_{sl} Q f_{s_{rx}}$, where L is the packet size in bits, t_{sl} is the captured GNSS signal length in seconds, Q is the quantization bits of the ADC, and $f_{s_{rx}}$ is the sampling frequency of the reception front-end (usually twice its bandwidth to fulfill the Nyquist-Shannon theorem). Therefore, there are three alternatives in order to work with small-sized packets: reduce the sampling frequency of the reception front-end, work with an ADC with low number of quantization levels and/or capture a signal of short length. Notice that a larger signal length to be captured will increase the amount of time the reception front-end, ADC, MCU, memory, and transmission RF front-end must be on active mode, and hence their energy consumption is also dependent of the signal length.

For this test setup, in order to remain the energy consumption of the cloud GNSS sensor as low as possible, the reception front-end bandwidth is fixed to 2 MHz, enough to capture a GPS L1 C/A signal, and the sampling frequency is set to 4 MHz. On the other hand, the ADC uses 1 bit to digitize the GNSS signals. According to state of the art RF reception front-ends and ADCs, the current consumption of such components has been set to 5 mA. A comparison between the energy consumed by the cloud GNSS sensor and a conventional GNSS sensor with hot and assisted start is performed and depicted in Figure 4. With the purpose of implementing a fair comparison between sensors, as they have different workflows, the energy needed by each module of a conventional GNSS sensor for a packet size of ~10^4 bits (meaning that the output NMEA has a size of roughly 1 KB) has been extracted and fixed from Figure 2. It can be seen how as the signal length to be captured by the cloud GNSS sensor is moderately small, the cloud GNSS sensor offers a more energy efficient positioning solution than a conventional GNSS sensor. Therefore, in Figure 4 is demonstrated that for a small signal length (up to approximately 25 and 50 ms for hot and assisted start in this work case) the cloud GNSS sensor provides energy savings of up to one order of magnitude and almost one and a half orders of magnitude in comparison with a conventional GNSS sensor for both hot and assisted start, respectively.

Positioning accuracy performance of the cloud GNSS sensor

In previous sections we have demonstrated that the cloud GNSS sensor outperforms the conventional GNSS sensor in terms of energy consumption under some constraints: the sampling frequency of the reception front-end, the quantization levels of the ADC and the signal length are limited to reduce the size of the packet to be forwarded. In this work case, the theoretical

Table 2. Experimental set-up: EC2 instance characteristics.

Instance type	c3.xlarge
vCPUs/threads	4
RAM capacity	7.5 GB
Storing capacity	2x40 GB (SSD)
Region	Frankfurt (EU)
On-demand price (taxes excluded)	$0.258 per hour
Reserved price (taxes excluded)	$0.11 per hour
Spot price (taxes excluded)	$0.0472 per hour

Table 3. Economic cost (taxes excluded) of the required cloud resources (*c3.xlarge* AWS EC2 instance) by a cloud GNSS sensor which provides one position fix per hour. The used signal length is set-up from 1 to 5 ms.

Payment type	Cost	Days of service	Annual cost
On-demand instance	$0.01	1	$3.65
Reserved instance	$0.01	2.5	$1.46
Spot instance	$0.01	6	$0.61

accuracy of the cloud GNSS sensor is analyzed with regard to its energy consumption, which is directly related with the signal length as we have previously seen in this section.

Low cost and low energy consumption positioning applications (e.g. IoT) usually just need of horizontal positioning information (x- and y-axis, east and north, respectively) and height is not required. The accuracy of these applications is typically provided by the Root Mean Square (RMS) horizontal error [12]:

$$\text{RMS horizontal error} = \text{HDOP} \cdot \sigma_{\text{UERE}}\,, \tag{1}$$

with HDOP the horizontal Dilution of Precision (DOP), a geometry factor that measures the effect of the satellites position or geometry in the solution error or accuracy, and σ_{UERE} is the effect of different error sources on pseudorange measurements or the User-Equivalent Range Error (UERE). In this work, a best-case scenario for the error sources of the pseudorange error is used as only errors incurred by the time delay are considered, whilst and errors incurred by atmospheric propagation models (~5 m), multipath or receiver noise (~0.5-1 m) are left aside in order to obtain a simplistic accuracy error. Note that the impact of these errors can be reduced by GNSS techniques thus obtaining a better positioning accuracy. In this case study, an HDOP of 1.4 with a probability >90% is fixed for a GPS L1 C/A signal assuming an unobstructed view of the sky [12]. The theoretical minimum of the time delay error is obtained by means of the Crámer-Rao bound [13]

$$\sigma_{\text{td}} = \frac{1}{2\,(2\pi)^2\,t_{sl}\,C/N_0\,\int_{-Bw/2}^{Bw/2} f^2 G_s(f)\,df}, \tag{2}$$

where t_{sl} is the signal length to be integrated in seconds, C/N_0 is the carrier-to-noise ratio, Bw is the reception RF front-end bandwidth and G_s is the normalized GNSS signal power spectral density, which for GPS L1 C/A is determined by

$$G_s(f) = T_c\,\text{sinc}^2(\pi f T_c) \tag{3}$$

with T_c the chip-length. Then, in order to obtain pseudorange estimation error in meters from the time delay error in seconds, we have to apply the expression

$$\sigma_{\text{UERE}}(m) \cong \sqrt{c^2\sigma_{td}} \tag{4}$$

where c is the speed of light. As shown in (2), the signal length t_{sl} has an impact on the pseudorange error, and hence, as the signal length is directly dependent with the energy consumed by the cloud GNSS sensor (Figure 4), a relationship between accuracy and energy is obtained. For this experimental set-up, the reception front-end has a bandwidth of 2 MHz, a sampling frequency of 4 MHz, and the quantization levels of the ADC are set to 1 bit. Although sampling frequency and quantization are not included in (2) to obtain the pseduorange error, in [14] is demonstrated their negative effect on the position-time error as they are reduced.

The expected accuracy performance of the cloud GNSS sensor with regard to its energy consumption under different working conditions is provided in Figure 5. The sensitivity of the HS-GNSS SwRx has been obtained for a probability of false alarm of

1e-6 and a probability of detection equal to 0.9. It is shown that as the C/N_0 decreases, if we want to keep the same positioning accuracy, the energy required by the sensor increases, as the amount of signal length to be captured and then processed by the HS-GNSS SwRx must be larger. The bound fixed at roughly 255 mJ has been obtained from Figure 4 and indicates when the cloud GNSS sensor stops being more energy-efficient than the conventional GNSS sensor with assisted start. The u-blox MAX-M8W GNSS receiver has an horizontal position accuracy in continuous mode of 2.5 m CEP 50% [7] (roughly equivalent to an RMS horizontal error of 3 meters), assuming a 24 hour static scenario with more than 6 visible satellites. Hence, the accuracy performance of the u-blox module is expected to decrease in an ON/OFF operation mode with limited acquisition and tracking time and under severe working conditions (RMS horizontal error of 5-30 m). Furthermore, in this test setup, the current consumption of the GNSS module is fixed to a uniform value (Table 1) for all working conditions although it can increase as the C/N_0 of the signal decreases.

Therefore, in Figure 5 is demonstrated that in outdoors sky working conditions, the cloud GNSS sensor may offer the same horizontal accuracy performance than a conventional GNSS sensor (3 meters, according to [7] under given conditions) while consuming a less amount of energy. Furthermore, the cloud GNSS sensor may consume up to more than an order of magnitude less of energy than a conventional GNSS sensor for applications in where a very precise position is not mandatory (e.g. 10-5 meters).

ECONOMIC COST OF CLOUD GNSS

Processing the raw GNSS samples file or IF data on a remote server (i.e. cloud server) instead of in the sensor itself as in conventional approaches implies the added cost of hiring cloud computing resources. In this section, the cost of hiring such cloud resources offered by AWS within the framework of an IoT application and the cost of sending the packets from the sensor to the cloud server are addressed.

AWS offers resizable and scalable virtual computing environments (instances) through the Elastic Compute Cloud (EC2) service. Such instances, which are the back-end bulk of the Cloud GNSS receiver, include computing resources such as RAM, CPU, storage, etc. EC2 offers a wide selection of instances types suited and optimized for different applications: general purpose, compute optimized, memory optimized, storage optimized, etc. Furthermore, AWS offers three basic ways to pay for its instances: on-demand, reserved and spot. On-demand instances are paid per hour use as they are opened and closed with a fixed cost. Reserved instances, which are paid in advance to its use, are suited for applications with predictable usage and offer discounts up to 75% with regard to on-demand. In addition, we can bid for EC2 spot instances, whose fluctuating price vary depending on the supply and demand of EC2 instances. The cost of spot instances can decrease down to a 90% with respect to on-demand instances. On the other hand, the price of the different instances may vary depending on the selected region (geographic area where Amazon EC2 servers are hosted).

In this test setup, a c*3.xlarge* instance (Table 2) has been used for processing raw GNSS samples files with signal length between 1 and 5 ms. The allocation of the GNSS signal processing tasks over the different computer resources provided by instances is assumed to be optimum in terms of usage time. That is to say, the computational tasks of a given amount of sensors are sequentially performed in the same EC2 instance with the objective of maximizing the amount of processed snapshots in an hour at the same instance and hence reducing the cost per snapshot. Now, let us suppose an IoT application whose sensor network must provide one position fix per hour. In such application, the expected cost of the cloud resources required by a single sensor is shown in Table 3. By using on-demand instances, 1 day of service per sensor has a cost of $0.01 (taxes excluded) whilst for the same cost reserved instances provide 2.5 days of service. In this aspect, the most cost-effective payment type is obtained when spot instances are used: 6 days of service has a cost of $0.01 (this cost may vary depending on the fluctuating price of the spot instance). Translated into cost per sensor per year (the battery lifetime of an IoT sensor is expected to last years), 1 year of service would cost about $0.61, $1.46 and $3.65 for spot, reserved and on-demand instances, respectively.

On the other hand, the cloud GNSS sensor has to send a packet (raw GNSS samples file) to the cloud server as well, which implies the cost of hiring a telecommunication channel. Using a reception front-end with sampling frequency equal to 4 MHz, an ADC with a quantization of 1 bit, and setting the signal length from 1 to 5 ms, the size of the packet would be of 0.5-2.44 KB, rounded to 3KB to take into account any other kind of information send by the sensor (e.g. security/cryptography information, device identification, etc.). That means that with a typical internet rate of 1 GB/month with a cost of $10, around 350000 packets can be sent, and hence the cost per packet is $0.0000286. Therefore, keeping in mind the previously introduced IoT application where each sensor provides its position once an hour, the annual cost of the telecommunication (i.e. 3G/4G in this test setup) between the cloud GNSS sensor and the cloud server would be of $0.25.

Then, combining the costs of hiring the EC2 instances and sending the packets from the sensor to the cloud, the annual cost per sensor would be of $0.86, $1.71 or $3.9 if it is a spot, reserved or on-demand instance, respectively. Thus, the cost implied by the use of the Cloud GNSS receiver is not a significant burden, considering that the expected cost of an IoT sensor is usually around tens of dollar.

CONCLUSIONS

This work has discussed the energy consumption of the different components of a conventional positioning sensor, being the GNSS module the most consuming one if the size of the packet to be forwarded from the sensor to a remote server or control center is not large. To tackle the energetic dilemma of conventional approaches, the use of a cloud processing scheme is proposed, namely Cloud GNSS receiver, where the purpose of sensors is just to capture the GNSS signal and send it to a cloud server where then will be processed. The energy consumption of the proposed cloud GNSS sensor has been analyzed and compared to conventional approaches. The obtained results show that under the constraint of working with a signal length of 25-50 ms, the use of the Cloud GNSS receiver is up to more than an order of magnitude more energy-efficient than the conventional approach (considering a GNSS receiver assisted start), and thus the battery lifetime of the sensor is improved. Then, the economic cost implied by the use of the Cloud GNSS receiver has been addressed. It is shown that for an application that provides one position fix per hour, the annual cost per sensor of hiring cloud computing resources is of $0.61. Finally, the cost of sending one packet from the cloud GNSS sensor to the cloud server has been estimated to be $0.0000286, with an annual per sensor of $0.25.

ACKNOWLEDGMENTS

The views presented in this paper represent solely the opinion of the authors and not necessarily the view of ESA. This work was partly supported by the European Space Agency (ESA) under contract No. 4000119070/16/NL/GLC and by the Spanish Government under grant TEC2014-53656-R.

REFERENCES

[1] GNSS GSA, *Market report, issue 5*, no. 5. 2017.
[2] V. Lucas-Sabola, G. Seco-Granados, J. A. López-Salcedo, J. A. García-Molina, and M. Crisci, "Cloud GNSS receivers: New advanced applications made possible," in *Proc. International Conference on Localization and GNSS (ICL-GNSS)*, 2016.
[3] A. Carroll and G. Heiser, "An Analysis of Power Consumption in a Smartphone," 2010.
[4] P. Juang, H. Oki, Y. Wang, M. Martonosi, L. Peh, and D. Rubenstein, "Energy-Efficient Computing for Wildlife Tracking : Design Tradeoffs and Early Experiences with ZebraNet," *ACM SIGARCH Comput. Archit. News*, pp. 96–107, 2002.
[5] J. Liu, B. Priyantha, T. Hart, H. S. Ramos, A. A. F. Loureiro, and Q. Wang, "Energy Efficient GPS Sensing with Cloud Offloading," *Proc. 10th ACM Conf. Embed. Netw. Sens. Syst.*, pp. 85–98.
[6] J. López-Salcedo, Y. Capelle, M. Toledo, G. Seco, J. López Vicario, D. Kubrak, M. Monnerat, A. Mark, and D. Jiménez, "DINGPOS: a hybrid indoor navigation platform for GPS and GALILEO," *21st Int. Tech. Meet. Satell. Div. Inst. Navig. (ION GNSS 2008)*, vol. 2, pp. 1780–1791, 2008.
[7] u-blox, "MAX-M8 u-blox GNSS chip datasheet." datasheet MAX-M8, https://www.u-blox.com/sites/default/files/MAX-M8-FW3_DataSheet_%28UBX-15031506%29.pdf, 2016.
[8] Texas Instruments, "CC1310 SimpleLink™Ultra-Low-Power Sub-1 GHz Wireless MCU." datasheet CC1310, http://www.ti.com/lit/ds/symlink/cc1310.pdf, 2016.
[9] Atmel, "SPI Serial EEPROM." http://www.atmel.com/images/Atmel-8832-SEEPROM-AT25M02-Datasheet.pdf, datasheet AT25M02, 2017.
[10] Infineon, "SLM 76CF3601P Secure µSlim Flash / EEPROM." datasheet SLM 76CF3601P, https://www.infineon.com/dgdl/SPO_SLM+76CF3601P_2013-03.pdf?fileId=db3a30433fcce646013fdef25e6a7b51, 2011.
[11] u-blox, "u-blox 8 / u-blox M8 Receiver Desription." receiver description M8, https://www.u-blox.com/sites/default/files/products/documents/u-blox8-M8_ReceiverDescrProtSpec_(UBX-13003221)_Public.pdf, 2016.
[12] P. Misra and P. Enge, *Global Positioning System: signals, measurements and performance*, 2nd ed. Ganga-Jamuna Press, 2006.
[13] E. D. Kaplan and C. J. Hegarty, *Understanding GPS: Principles and Applications*, 2nd ed. Artech House, 2005.
[14] K. M. Pesyna, R. W. Heath, and T. E. Humphreys, "Precision Limits of Low-Energy GNSS Receivers," *Proc. ION GNSS+ Meet. Tennessee), Inst. Navig.*, 2013.