Automated observatory in Antarctica: real-time data transfer on constrained networks in practice.

Stephan Bracke[1], Alexandre Gonsette[1], Jean Rasson[1], Antoine Poncelet[1], Olivier Hendrickx[1]

[1] Institut Royal Météorologique (IRM), Centre de Physique du Globe, 5670 Viroinval (Dourbes), Belgium.

Correspondence to: Stephan Bracke (stephan.bracke@meteo.be)

Abstract. In 2013 a project was started by the geophysical centre at Dourbes to install a fully automated magnetic observatory in Antarctica. This isolated place comes with specific requirements: unmanned station during six months, low temperatures with extreme values up to - 50 °, minimize power consumption, satellite bandwidth limited to 56 Kbit/sec. The ultimate aim is to transfer real-time magnetic data every second: vector data from a LEMI-25, absolute F measurements from a GEMS proton magnetometer and absolute magnetic inclination and declination measurements (5 times a day) with an automated DI-flux. Traditional file transfer protocols (for instance FTP, mail, Rsync) show severe limitations when it comes to real-time. After evaluation of pro and cons of the available real-time internet of things (IoT) protocols and seismic software solutions known to UGCS, we chose to use message queuing telemetry transport (MQTT) and receive the one second data with a negligible latency cost and no loss of data. Each individual instrument sends the magnetic data immediately after capturing it and arrives at approximately 300 milliseconds after sending which corresponds with the normal satellite latency.

1 Introduction

Princess Elisabeth Antarctica (Figure 1 : right PEA), located on Utsteinen Nunatak in Queen Maud Land (71°57' S 23° 20' E), is a Belgian scientific polar research station, which went into service on February 15, 2009. It is approximately 220 km from the Antarctic coast, which makes it an ideal logistics hub for field exploration in the 20°- 30° E sector of Antarctica. The station is unmanned during the Antarctic winter period but each year from November until February there is a crew available. In 2013 a project was started by the geophysical centre at Dourbes to install a fully automated magnetic observatory in Antarctica. Three instruments will be used (Figure 2):

- LEMI-25 (Figure 2 : left) variometer registers 1 HZ magnetic variation data along three axes XYZ.
- GEMS GSM-90 (Figure 2: middle) proton magnetometer registers 1HZ absolute magnetic field measurements.
- AUTODIF (Figure 2: right) automated DIflux theodolite which will execute 5 times a day a magnetic inclination and declination measurement.(Rasson, J. et al, August 2011).

The data needs to be sent to the observatory in Belgium. Recently the need for real-time data transfer becomes an important topic in the operation of magnetic observatories:

- Space weather forecasts need real-time availability of magnetic data.
- In the oil industry real-time magnetic data is used to monitor and correct directional drilling trajectories.
- Intermagnet is investigating the possibilities to achieve real-time data transfer

The project in Antarctica gives the opportunity to achieve real-time data transfer. To be able to talk about real-time it is necessary that we come to a mutual agreement of what "real-time" exactly means. Let's consider following definition of real-time as stated in the Cambridge dictionary:

Real-time is a term that applies to the communication, presentation or reporting of events on the same time that they actually happen. However computers that collect and transmit data over a network have a time delay due to automated data processing and network transmission. For example, the LEMI-25 introduces a known delay of 0.3 s (LVIV, October 2014))

before the data is available and can be send over the internet. Because of this time delay the correct term to use is near real-time (NRT).

Usually, magnetic data collection projects use file transfer protocols .Most of the time data is written to a file and this one is then transmitted over the internet. Popular solutions are:

- Rsync: is a linux utility that synchronizes directories over the network. Rsync typically uses SSH connections to encrypt and secure data transfer. It will only update the necessary differences.
- FTP: Classical file transfer protocol that makes it able to copy files to a remote server. When security and encryption is needed FTP can be extended with certificates (FTPS)
- SFTP: same as FTP but underlying connection is different. The security is done over SSH as in Rsync which comes with a latency impact compared to the pure FTP(S) but is simpler to use behind firewalls and highly secured.
- Mail: traditional mail can be used to send the file in attachment to a destination. As it is mail there is always an uncertainty of delivery and an unknown latency

All these protocols have their value as being standard and easy to use but in terms of near real-time they add a serious overhead. First of all the files need to be created and secondly the protocols are focussed on sending a file correctly but not really on fast transfer with limited overhead. Using these protocols to attain near real-time data transfer is against their intended purpose and all of them are still bandwidth intensive. While these protocols can always serve as fall-back mechanism, it is time to investigate other protocols that minimize the time it takes to read the data from the instrument and send it to the data centre.

2 Near real-time data transfer protocols

Previous protocols are all based on file transfer. Instead of working with intermediate files, we need protocols that can send data packages over the network, immediately after they are read from the instrument. Although we focus on the near real-time aspect, other aspects are equally important for this project:

- The solution needs to work on a limited bandwidth satellite connection.
- The solution needs to have some level of guarantee delivery.

To apply to all this requirements the considered solutions in the next chapters are message oriented solutions, which make it possible for software components to communicate with each other by using predefined messages.

2.1 Protocols used in seismology

In seismology near real-time data transfer was always an important requirement. At USGS two software packages are mentioned that promise near real-time data transfer: earthworm and antelope (proprietary). A third one that has gained popularity is called SeisComP3. These three packages are offering more than just data transfer and often come with seismologic data visualization and analysis tools. They are often full blown server implementations (with tools and database to configure) and although they have different possibilities to communicate data, each of them comes with the possibility to send data to it in the form of SeedLink packages. SeedLink is a data transfer protocol defined by the incorporated research institutions for seismology (IRIS). As stated on their website: The SeedLink protocol is a robust data transmission intended for use on the internet or private circuits that support TCP/IP. The protocol is robust in that clients may disconnect and reconnect without losing data, in other words transmissions may be resumed as long as the data still exist in the servers buffer. Requested data streams may be limited to specific networks, stations, locations and/or channels. All data packets are 512-byte miniSEED records. Normally 1HZ magnetic data can be packaged in +/- 30 bytes (a timestamp and 3 real values). When it comes to limited bandwidth connections on satellite links, sending packages that are only filled with 10 % of useful data becomes a waste of resources. MiniSEED records are defined with equally spaced time series in mind, which matches

to the continuous measurements of variometers and scalar instruments. For Automated DI-flux measurements, which are not evenly spaced in time, miniSEED records aren't favourable. We can also observe that the user community of these solutions aren't that big compared to solutions offered by alternative open source projects (see S2.2.1), so good sample code and support is not easy to find. Finally we can state that using these kinds of solutions to transfer magnetic data is a valuable option if on your servers you have already earthworm, antelope or SeisComP3. If not it will come with a big learning curve and sometimes poorly documented support.

2.2 The internet of things.

Today we live in a world where everything is connected and all devices become "smart". All these devices register data and send data over the internet: tv's, smart phones, smart thermostats, smart lights, surveillance systems, robotic lawn mowers, etc. This inter-networking of physical devices is called the internet of things (IoT). Thanks to this increase in commercial applications we can profit of the evolution of data transfer protocols to apply to these emerging needs. The application often comes with similar requirements as the ones we have for instruments on remote locations:

- Need to run on small devices
- Minimize power usage (often devices or on battery)
- Run on unreliable networks (Wi-Fi, mobile networks)
- Send regularly small messages.

Another advantage of this evolution is that there is a large open source community driven by sharing knowledge and solving problems together. On this base a search on the internet on data transfer resulted in three protocols that were suitable to establish the needed near real-time data transfer:

- Advanced message queueing protocol (AMQP)
- Streaming text oriented protocol (STOMP)
- Message queue telemetry transport (MQTT)

First of all these three mentioned protocols are not implementations, they are just program language-agnostic descriptions how clients and server can communicate on an asynchronous level. Two of them: MQTT and AMQP are even evolved into an OASIS open standard, with the advantage that different implementations of client libraries and servers have emerged in both open source and commercial landscape. To be able to choose among these protocols it is important to look at their specifications and their targeted audience. While taken the definition from there respectively websites:

AMQP was designed as a replacement for existing proprietary messaging middleware (IBM, Microsoft). It is a full blown business to business message protocol with focus on reliability, interoperability and security. The protocol makes it possible to apply flexible routing of messages, with transaction support. The protocol is at his lowest level an efficient, binary, peer-to-peer protocol for transporting messages between two processes over a TCP/IP connection.

MQTT is a machine-to-machine (M2M)/"Internet of Things" connectivity protocol. It was designed as an extremely lightweight publish/subscribe messaging transport. It is useful for connections with remote locations where a small code footprint is required and/or network bandwidth is at a premium. It is a binary protocol over TCP/IP.

STOMP is a protocol that is completely text based making it more in favour to be used over HTTP. It stays very simple and has no knowledge of transactional context meaning that there is no direct support for any guarantees of delivery (Mesnil J et al; August 2014).

protocol	lightweight	binary	Quality of Service	Designed for low-bandwidth networks
AMQP	No	Yes	Yes	No
MQTT	Yes	Yes	Yes	Yes
STOMP	Yes	No	Not out of the box	No

Table 1: comparison of message protocols

The table shows that among these three protocols MQTT is the most appropriate to solve the data transport for the installation in Antarctica

3 MQTT explained

Message Queue Telemetry transport was designed and documented by Andy Stanford-Clark of IBM and Arlen Nipper of Cirrus Link Solutions (a company specialized in real-time telemetry solutions) in 1999. The design principles were to minimize network bandwidth and device resource requirements whilst also attempting to ensure reliability and some degree of assurance of delivery (Lampkin V et al. 2012). In 2011 it was used by Facebook to reduce their phone-to-phone delivery of the messenger from several seconds to hundredths of seconds. In 2013, IBM submitted MQTTv3.1 to the organization for the advancement of structured information standards (OASIS) making it available for everybody to use and implement it into their applications (Bank A.et al. October 2014). From that moment on MQTT has definitely evolved to one of the most used protocols in the internet of things applications.

3.1 Publisher and Subscribers

The MQTT protocol is a publish/subscribe mechanism which needs a broker (Figure 3) to communicate between those who send messages (Publishers) and those who receive messages (Subscribers). Publishers and subscribers are completely decoupled. This means that publisher and subscriber don't have to know about each other and are completely independent. A subscriber doesn't need to be up and running to be guaranteed to receive the messages from the publisher. For each message published there can be multiple subscribers to receive the messages. We could compare it with a subscription to a newsletter. Messages are published on topics. In MQTT topics are UTF-8 strings, which are defined on the broker (see S3.2) to filter messages for each client. A topic consists of one or more topic levels. The forward slash is used to separate each level within the topic tree e.g.: **myhome/floor/room/temperature**. The publisher on this topic will be a small device with a temperature sensor which reads on regular intervals a temperature and publishes this on the topic. The messages are byte messages which form the contract between publisher and subscriber. There is no formal way to document a message structure but it comes down to describe how the received byte array needs to be interpreted. From now on anyone who needs the temperature can subscribe to the topic and will receive the byte message. Using topic levels to define a topic tree becomes interesting when a subscriber is interested in messages at a certain level in the tree. For example a subscriber that wants to get all messages related to one particular room can use multilevel wildcard **#** in his subscription. A subscription to myhome/floor/room/**#** will result in receiving all messages to all sub levels of myhome/floor/room. If however a subscriber is interested in the temperature in all rooms on a certain floor, he can use a single level wildcards + and subscribe to myhome/floor/+/temperature. Publishers and subscribers are software components. To make MQTT work we can use client libraries which are available in multiple programming languages (C++, java, C#, python, etc).

3.2 The MQTT broker

The client libraries for publisher and subscribers are lightweight, but before they can communicate with each other you need another piece of software called the broker. You can find many different broker implementations (some open-sourced, some proprietary).

The main responsibilities of the broker are:

- Decouple the publishers from subscribers
- Make topics available for publishers and subscribers
- Ensure receiving and correct delivery of the MQTT messages
- Configure security

The choice of the correct broker will depend on lots of factors (ease of use, needed scalability, own preferences etc.). In our particular case the load is relatively low and the choice was made on ease of use and lightweight deployment. That's why we used mosquitto which did fulfil our requirements.

There are different ways of deploying a broker:

- Deploy and use one broker on the sensor site (remote site)
- Deploy and use one broker on the data centre site
- Use of two brokers one on each site and configure a bridge between the two brokers.(Figure 4)

The bridge solution is the most elaborated one where decoupling is optimized between publisher and subscriber. The usage of a bridge puts the responsibility of guarantee delivery, buffering and redelivery on the configuration of the bridge. To bridge the brokers you will link topics from one broker to the other one by means of configuration. As a consequence continuous monitoring of bridges is necessary. In the setup at Antarctica, we opted for one broker on the data centre site at Dourbes for the simple reason that we had full control on the firewall settings and servers on this site and no upfront knowledge of the possibilities at Antarctica.

3.3 Quality of service

The foot print of MQTT is tiny and the overhead small what makes it a fast protocol. Beside speed and overhead, we also need some control on the guarantee of delivering a message in MQTT.

MQTT introduces three levels of quality of service (QOS).

- QOS 0: lowest level of assurance. The message will be send at most once, but it will not survive failures. It will never introduce duplicates. Often it is referred to as "fire and forget".
- QOS 1: The message will be send at least once. This quality of service introduces the possibility of duplicate messages (it is the subscriber that can receive messages more than once). It will survive connection loss. The QOS requires an acknowledgement back from the server before the client can discard the message.
- QOS 2: The message is send exactly once. The subscriber will be guaranteed to receive the message exactly one time. This QOS survives connection loss but introduces extra layer of communication messages between publisher and broker (two extra messages to assure that message was received)

QOS 0 is by all means the fastest and the least band-width consuming. However acknowledge messages are 1 byte so the overhead compared with the advantage of being sure the message arrives, is negligible and often the preferred way of sending messages. QOS can be set on each individual message, so it is simple to change QOS between messages.
In our case we use QOS 1 to publish messages.

4. Application to an automated observatory in Antarctica

In February 2014 a first mission was planned to locate a place where it is possible to install a magnetic observatory. This resulted in finding the best spot to place a magnetic observatory at approximately 500 meters of the station. The location is near to the Utsteinen mountain to make it possible to fix the station on solid rocks such that the pillars will not move. A radome (Figure 1: left) was chosen because of the fact that it is not magnetic, it doesn't block the gps signals and its form reduces greatly wind loads. The radome has some limitations to be taken into account: it is not heated, power supply will be delivered from the station but consumption should be minimized as much as possible.

During the season of 2014-2015 a second mission took place. The main goals were to install the first two instruments for continuous monitoring of the magnetic field and establish a way to communicate the data back to Belgium. For measurements of the magnetic field vector a LEMI-25 (Figure 2: left) was selected for its known temperature stability. Because the radome is not heated the LEMI-25 got adapted by the supplier so that it can withstand the cold temperatures. The second instrument is a GEMS GSM-90 magnetometer (Figure 2: middle) that measures continuously the magnetic field strength. Both instruments have a sampling rate of 1Hz and come with a windows program that constantly logs the measurements in a file. They are equipped with their own GPS and samples are delivered with a timestamp. Because of the limitations of the radome, we don't use windows pc's but looked into the possibilities of small ARM processors. Thanks to popularity of Rasberry PI, these small cheap ARM processors gain in popularity and opened up a whole new market. We decided to use the Beaglebone Black (Figure 5): a small Texas Instruments single board computer. It is comparable with the Raspberry but it is rather designed for robotics having the following advantages:

- The hardware is completely open-sourced so adaptations or own productions are fairly easy.
- A rugged version for extended temperature range (-40°C – 80 °C) is available .
- It comes with two 46 pin GPIO headers
- It has on board 4 Gigabyte of internal storage making the boot from SD card not necessary.

As these boards are linux based computers the needed software has been rewritten and both instruments are controllable via a web-interface. The radome itself has a diameter of 4,380 meter and the instrument emplacement was optimized to limit interference which resulted into the layout shown in Figure 6. The electronics of both instruments, network switch and Beaglebones are placed on a shelf (+/- 2,30 m above the ground) as far as possible from the instruments (Figure 7). All installed instruments and electronics are guaranteed to work at -40°C, except the electronics of the LEMI which is has a minimum operating temperature of -20°C. Therefore it is placed in a box that is heated when temperature drops lower than -20°C. A pillar is foreseen to place an AUTODIF (Gonsette et al. 2013) in the next season to establish a completely automated magnetic observatory. As shown on the schematics depicted in Figure 8 the radome is connected to the Princess Elisabeth station with fiber-optics. Princess Elisabeth station has a permanent internet link and limits each scientific project to 56 Kbit/s. To transfer the data MQTT is used. A mosquitto broker is installed at Dourbes (Figure 8). This MQTT broker is only accessible from the public ip address of the Princess Elisabeth station. The broker is configured to allow access by username and password. Each username has access to predefined topics and access can be limited to read or write. The security is defined on the broker by means of access control lists. At the broker site two subscribers listen permanently to the incoming messages and store the data in a database.

In the radome in Antarctica two publishers send second data:

- Vector instrument LEMI-25 sends each second a byte message of 16 bytes.
- Scalar instrument GEMS-GSM 90 sends each second a message of 8 bytes.
- Both use a QOS 1 which guarantees that messages arrive at least once

The programs on the beaglebones (Figure 5) are written to publish the magnetic data immediately after it is read from the serial port of the device. This makes it possible to publish data captured in the radome on Antarctica directly to a broker installed on a server in Belgium. The topic structure used is: iagacode/instrumentid/samplerate: pea/lemi0001/sec. This means that this topic receives 1 second vector data of the LEMI-25 located at Princess Elisabeth station Antarctica (PEA). The topic tree contains the key that identifies the metadata of the instrument which implies that the message doesn't need to contain the metadata. The topic corresponds with one up front defined byte message structure, which can be used to receive and read near real-time data coming from Antarctica.

Every MQTT connection is made once and never closed. The library used (in our case Node.js library: MQTT.js) will automatically reconnect if connection gets lost. To be able to detect that the connection is stale, there is an important

parameter called keep alive. The keep alive interval is the longest possible period of time in seconds, which broker and publisher can endure without sending a message. If the broker doesn't receive any messages after the keep alive interval the broker will disconnect. The publisher can detect a connection broken by sending a ping message after the keep-alive interval. If the broker doesn't respond the publisher will try to re-establish the connection. In our project the keep alive is set to 10 seconds. On protocol level this means that as long as we assume there is a connection we can send messages and the connection can be broken without noticing it for maximum 10 seconds. On this unnoticed stale connection, we could continue to send messages of QOS1 that will never be acknowledged by the broker (maximum 10 because of the keep alive which assures to close the stale connection after 10 seconds). After the connection is re-established these messages will be send automatically again by MQTT because of the assurance of QOS 1. If we however try to send a message during the time we have no connection (for example a physical problem on the satellite itself or the server in Belgium) the MQTT library will respond with an error and it is up to the software to deal with this not send messages. To cope with this problem we just used a memory queue which stores 4 hours of second data. This means that from now on we can live with connection failures up to 4 hours (keep in mind that we lost our real time promises here). Once a connection is re-established the four hours of not transmitted data is send to the broker.

So during one year data transmission we evaluated that the mean delivery time is approximately 300 ms which corresponds with standard satellite delays (no real impact due to the MQTT protocol). Connections got lost +/- 10 times a month (small failures of a couple of seconds), which are re-established and data is resend without any problem. During one year of experience we had only one big connection lost mainly because of the fact that a fibre optic cable at Belgium got broken. It took two days to recover the connection. To recover these two days of data we felt back to standard FTP of the recorded files at Antarctica.

5. Conclusions

Realizing near real-time data transfer today is feasible with standard open protocols and open source tools. It doesn't come for free because it introduces the need of managing and monitoring a message broker. All research was done in 2014, when we re-evaluate the taken decisions we can see that today MQTT has evolved to a mature near real-time data transfer protocol widely adopted. Although we can't neglect that other new promising alternatives need to be investigated:

- MQTT-SN: MQTT for Sensor Networks is aimed at embedded devices on non-TCP/IP networks, such as Zigbee. MQTT-SN is a publish/subscribe messaging protocol for wireless sensor networks (WSN), with the aim of extending the MQTT protocol beyond the reach of TCP/IP infrastructure for Sensor and Actuator solution
- CoAP: The Constrained Application Protocol (CoAP) is a specialized web transfer protocol for use with constrained nodes and constrained networks in the Internet of Things. The protocol is designed for machine-to-machine (M2M) applications such as smart energy and building automation.

A third mission will be executed in the next season to install an AUTODIF. This will eventually result in the realisation in the first fully automated magnetic observatory with near real time data transfer.

References

Mesnil J., Mobile and web messaging: ISBN-13: 978-1491944806, ISBN-10: 1491944803, August 2014

Banks A. and Gupta R.: MQTT version 3.1.1, . OASIS Standard . 29 October 2014

Rasson, J. L. and Gonsette, A.: The Mark II Automatic Diflux, Data Sci. J., 10(August), IAGA169-IAGA173, doi:10.2481/dsj.IAGA-24, 2011

Gonsette, A., and Rasson, J. AUTODIF: Automatic Absolute DI Measurements, pp. 16-19, proceedings of the XVth IAGA Workshop on Geomagnetic Observatory Instruments, Data Acquisition and Processing. Boletin ROA No. 03/13, 2013.

Lampkin V. Tat Leong W., Olivera L., Rawat S, Subrahmanyam N , Xiang R.: Building Smarter Planet Solutions with MQTT and IBM WebSphere MQ Telemetry, ISBN: 0738437085, September 2012

Intermagnet 1-second standard flux-gate magnetometer user manual rev 1.6, LVIV October 2014

Figure 1: Magnetic radome and Princess Elisabeth Station

Figure 2: LEMI-25 (left) GEMS GSM-90 (middle) AUTODIF (right)

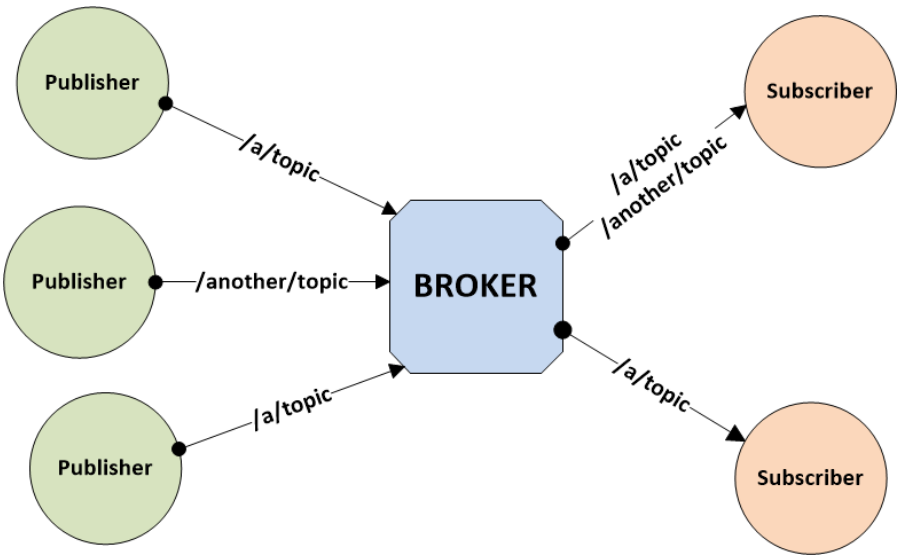

Figure 3: MQTT broker

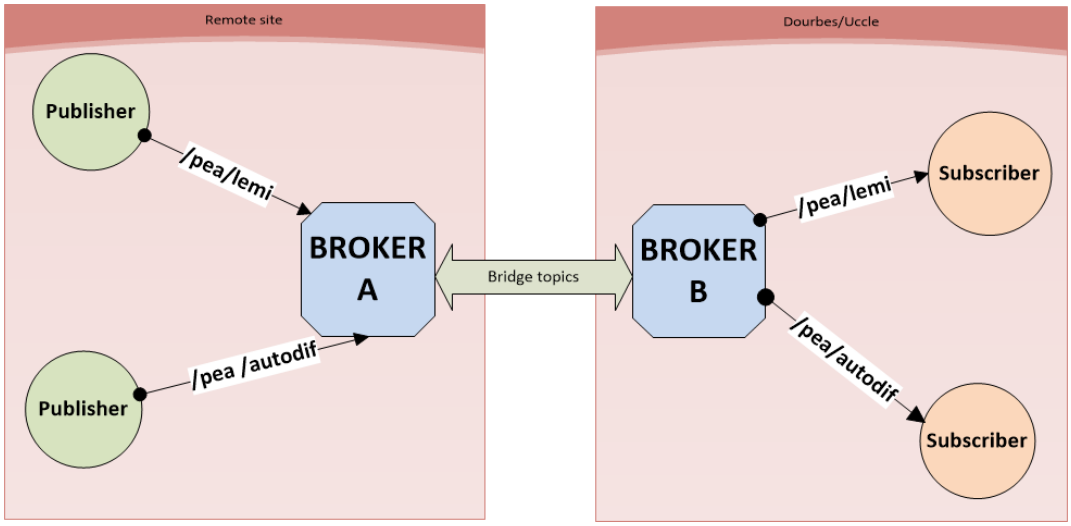

Figure 4: bridging brokers

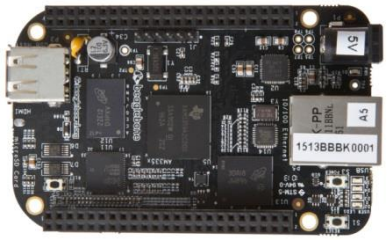

Figure 5: Beaglebone black ARM processor

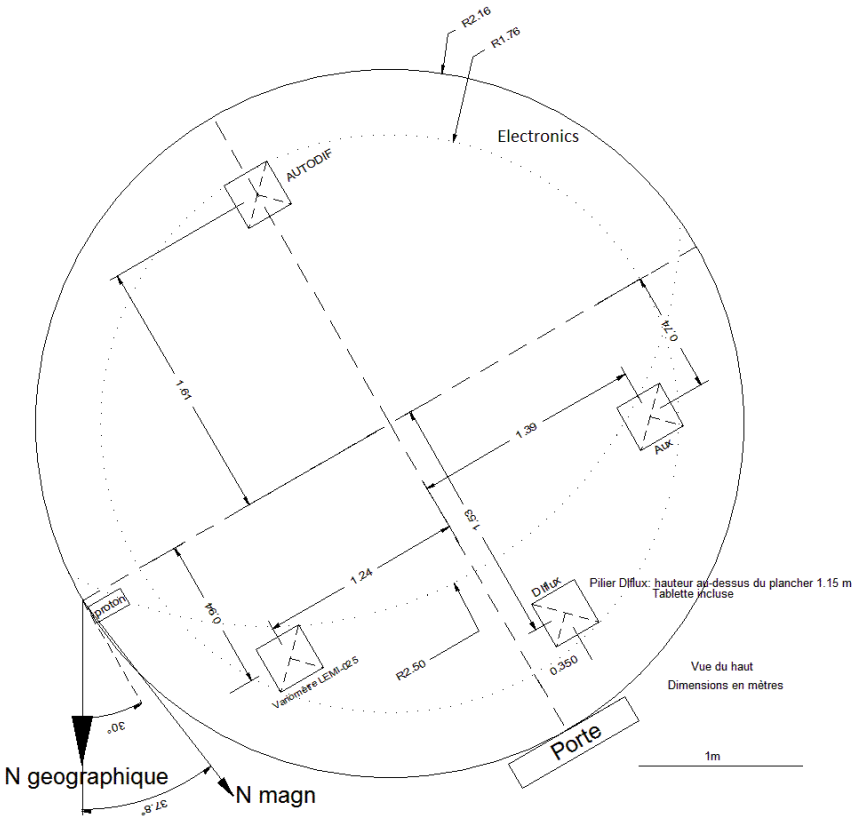

Figure 6: positions of instruments inside the radome

Figure 7: Electronics on a shelf in the radome

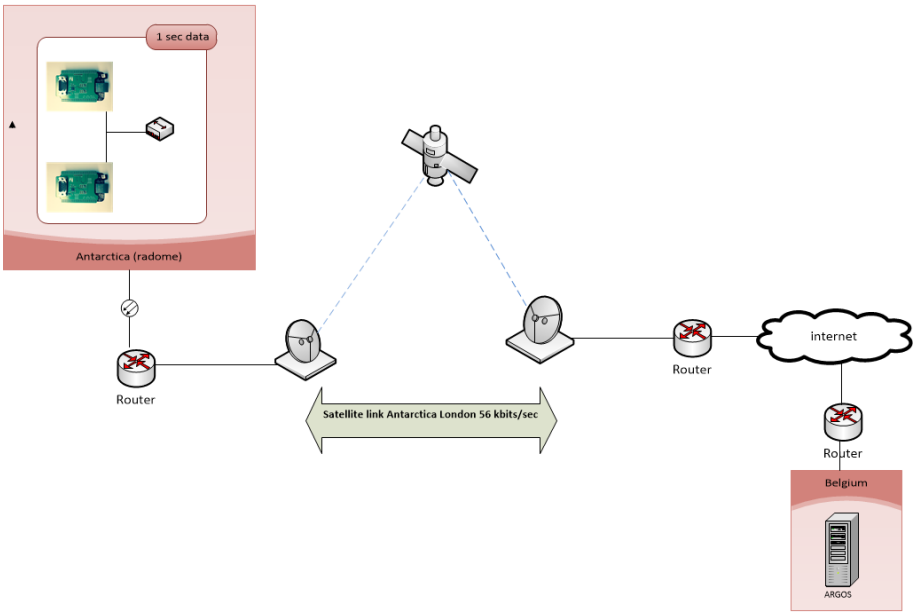

Figure 8: data transfer