Geosci. Instrum. Method. Data Syst., 6, 285–292, 2017
https://doi.org/10.5194/gi-6-285-2017

Geoscientific Instrumentation Methods and Data Systems
Open Access
EGU

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

Stephan Bracke, Alexandre Gonsette, Jean Rasson, Antoine Poncelet, and Olivier Hendrickx

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

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

Received: 3 March 2017 – Discussion started: 27 March 2017
Revised: 13 June 2017 – Accepted: 14 June 2017 – Published: 9 August 2017

Abstract. In 2013 a project was started by the geophysical centre in Dourbes to install a fully automated magnetic observatory in Antarctica. This isolated place comes with specific requirements: unmanned station during 6 months, low temperatures with extreme values down to $-50\,^{\circ}\mathrm{C}$, minimum power consumption and satellite bandwidth limited to $56\,\mathrm{Kbit\,s^{-1}}$. The ultimate aim is to transfer real-time magnetic data every second: vector data from a LEMI-25 vector magnetometer, absolute F measurements from a GEM Systems scalar proton magnetometer and absolute magnetic inclination–declination (DI) measurements (five times a day) with an automated DI-fluxgate magnetometer. Traditional file transfer protocols (for instance File Transfer Protocol (FTP), email, rsync) show severe limitations when it comes to real-time capability. After evaluation of pro and cons of the available real-time Internet of things (IoT) protocols and seismic software solutions, we chose to use Message Queuing Telemetry Transport (MQTT) and receive the 1 s data with a negligible latency cost and no loss of data. Each individual instrument sends the magnetic data immediately after capturing, and the data arrive approximately 300 ms after being sent, which corresponds with the normal satellite latency.

1 Introduction

Princess Elisabeth Antarctica (PEA, Fig. 1b), 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 15 February 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 in Dourbes to install a fully automated magnetic observatory in Antarctica. Three instruments will be used (Fig. 2):

- LEMI-25 (Fig. 2: left panel) variometer, which registers 1 Hz magnetic variation data along three axes (X, Y, Z);
- GEM Systems GSM-90 (Fig. 2: middle panel) proton magnetometer, which registers 1 Hz absolute magnetic field measurements;
- AUTODIF (Fig. 2: right panel), automated declination–inclination (DI)-flux theodolite, which will execute a magnetic inclination and declination measurement five times a day (Rasson and Gonsette et al., 2011).

The data need to be sent to the observatory in Belgium. Recently the need for real-time data transfer has become 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 are used to monitor and correct directional drilling trajectories;
- INTERMAGNET is investigating the possibilities to achieve real-time data transfer.

The project in Antarctica gives us 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 us consider the following definition of real time as stated in the Cambridge dictionary: "Real

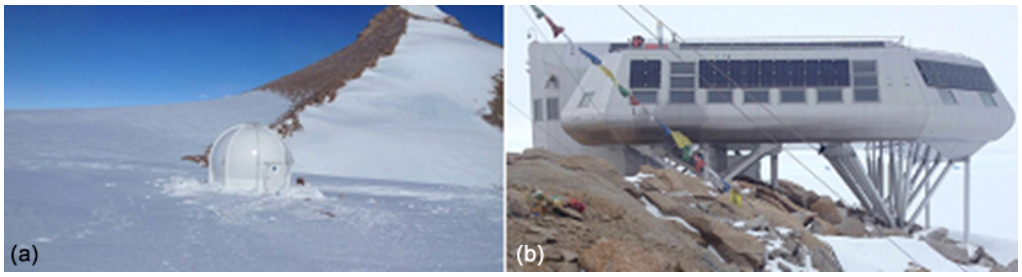

Figure 1. Magnetic radome and Princess Elisabeth station.

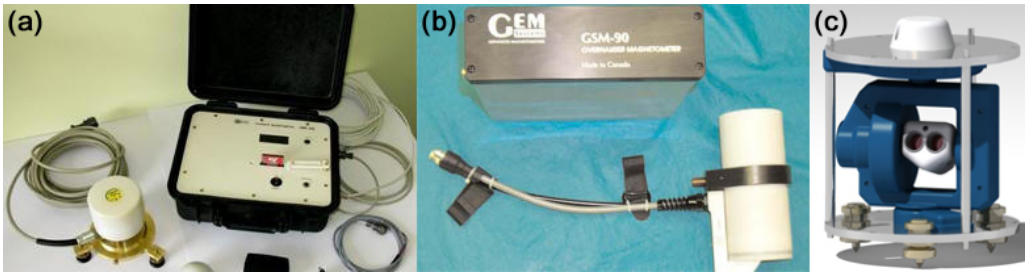

Figure 2. LEMI-25 **(a)**, GEM Systems GSM-90 **(b)**, AUTODIF **(c)**.

time is a term that applies to the communication, presentation or reporting of events at the same time that they actually happen" (Cambridge online dictionary, 2017). 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 are available and can be sent over the Internet. Because of these time delays, the correct term to use is near-real time (NRT).

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

- rsync: a Linux utility that synchronizes directories over the network. rsync typically uses Secure Shell (SSH) connections to encrypt and secure data transfer. It will only update the necessary differences between two files.
- FTP (File Transfer Protocol): classical network protocol that makes it possible to copy files to a remote server. When security and encryption are needed, FTP can be extended with certificates (FTPS).
- SFTP: same as FTP, but the 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.
- Email: traditional email can be used to send the file in attachment to a destination. As is always the case with email, there is an uncertainty of delivery and an unknown latency.

All these protocols have value in 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 fallback mechanisms, it is time to investigate other protocols that minimize the time it takes to read the data from the instrument and send them 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 with a limited-bandwidth satellite connection;
- the solution needs to have some level of guaranteed delivery.

Solutions with which to apply to these requirements can be found in message-oriented middleware. Message-oriented middleware is software or hardware infrastructure supporting the sending and receiving of messages between distributed

systems. In the next chapters possible message-oriented middleware solutions will be explored.

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, provided by Boulder Real Time Technologies). A third one that has gained popularity is called SeisComP3 and is provided by GFZ (GeoForschungsZentrum) in Germany. These three packages offer more than just data transfer and often come with seismological 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). The SeedLink protocol is a robust data transmission intended for use on the Internet or private circuits that support Transmission Control Protocol/Internet Protocol (TCP/IP) (IRIS website, 2017). 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 1 Hz magnetic data can be packaged in ± 30 bytes (a timestamp and three 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 match the continuous measurements of variometers and scalar instruments. For automated DIflux measurements, which are not evenly spaced in time, miniSEED records are not favourable. We can also observe that the user community of these solutions is not that big compared to solutions offered by alternative open-source projects (see Sect. 2.2), so good sample code and support are more difficult to find. Finally we can state that using these kinds of solutions to transfer magnetic data is a valuable option if on one's servers one already has Earthworm, Antelope or SeisComP3 for seismic data.

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: televisions, 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 from the evolution of data transfer protocols to apply to these emerging needs. The application often comes with similar requirements to the ones we have for instruments at remote locations:

- ability to run on small devices;
- minimum power usage (often devices or battery);
- running on unreliable networks (Wi-Fi, mobile networks);
- regularly sending 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 basis an Internet search 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 Queueing Telemetry Transport (MQTT).

First of all these three mentioned protocols are not implementations; they are just program language – agnostic descriptions of how clients and a server can communicate on an asynchronous level. Two of them, MQTT and AMQP, have even evolved into an Organization for the Advancement of Structured Information Standards (OASIS) open standard, with the advantage that different implementations of client libraries and servers have emerged in both open-source and commercial landscapes. To be able to choose among these protocols, it is important to look at their specifications and their target audience.

AMQP was designed as a replacement for existing proprietary messaging middleware (IBM, Microsoft). It is a full-blown business-to-business message protocol with a focus on reliability, interoperability and security. The protocol makes it possible to apply flexible routing of messages, with transaction support. The protocol is at its 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) 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 favourable 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, 2014).

Table 1 shows that among these three protocols MQTT is the most appropriate to solve the data transport for the installation in Antarctica.

Table 1. Comparison of message protocols.

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

3 MQTT explained

Message Queueing 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 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 milliseconds. In 2013, IBM submitted MQTTv3.1 to OASIS, making it available for everybody to use and implement it into their applications (Banks and Gupta, 2015). The MQTT standard was subsequently submitted by OASIS to the ISO/IEC JTC 1 information technology body and resulted in the ISO/IEC 20922 standard in 2016. MQTT has definitely evolved into one of the most used protocols in Internet of things applications.

3.1 Publisher and subscribers

The MQTT protocol is a publish–subscribe mechanism which needs a broker (Fig. 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 do not have to know about each other and are completely independent. A subscriber does not 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 at 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, so we need to describe how the received byte array needs to be interpreted. From now on anyone who needs the temperature information 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 who 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.).

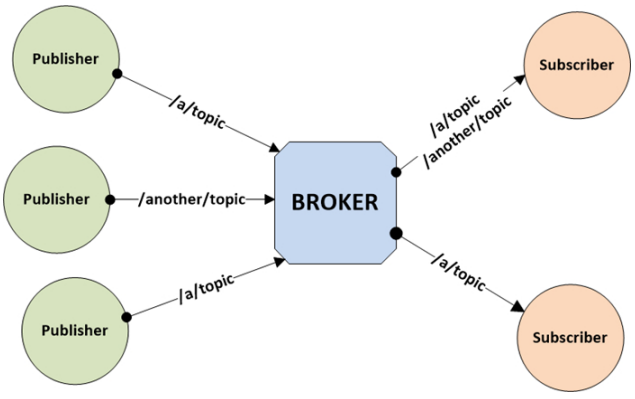

Figure 3. MQTT broker.

3.2 The MQTT broker

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

The main responsibilities of the broker are to

- decouple the publishers from subscribers;
- make topics available for publishers and subscribers;
- ensure correct delivery of the MQTT messages;
- configure security.

The choice of the correct broker will depend on a lot of factors (ease of use, needed scalability, own preferences etc.). In our particular case the load is relatively low, and the choice was made based on ease of use and lightweight deployment. That is why we used the Mosquitto broker, which fulfilled our requirements.

There are different ways of deploying a broker:

- deploy and use one broker at the sensor site (remote site);

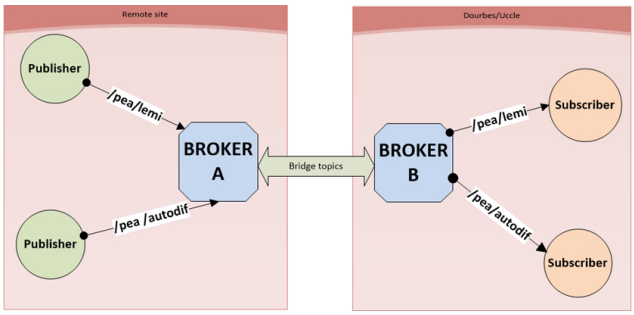

Figure 4. Bridging brokers.

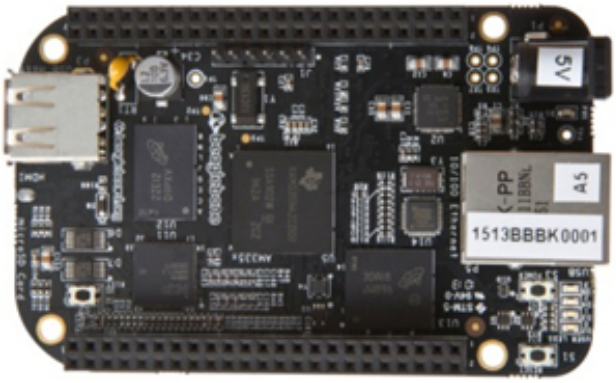

Figure 5. BeagleBone Black ARM processor.

– deploy and use one broker at the data centre site;

– use two brokers, with one on each site, and configure a bridge between the two brokers (Fig. 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 guaranteed delivery, buffering and redelivery on the configuration of the bridge. To bridge the brokers, one will link topics from one broker to the other by means of configuration. As a consequence continuous monitoring of bridges is necessary. In the setup in Antarctica, we opted for one broker at the data centre site in Dourbes for the simple reason that we had full control over the firewall settings and servers on this site and no upfront knowledge of the possibilities in Antarctica.

3.3 Quality of service

The footprint of MQTT is tiny, and the overhead small, which 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 sent 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 sent at least once. This quality of service introduces the possibility of duplicate messages (it is the subscriber who can receive messages more than once). It will survive connection loss. The QOS requires acknowledgement back from the server before the client can discard the message.

– QOS 2: the message is sent exactly once. The subscriber is guaranteed to receive the message exactly one time. This QOS survives connection loss but introduces an 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 bandwidth-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: approximately 500 m from the station. The location is near the Utsteinen mountain to make it possible to fix the station on solid rocks such that the pillars for the geomagnetic instruments will not move. A radome (Fig. 1: left panel) was chosen because of the fact that it is not magnetic, it does not block the GPS signals and its form greatly reduces wind loads. The radome has some limitations to be taken into account: it is not heated, and 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 (Fig. 2: left panel) 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 GEM Systems GSM-90 magnetometer (Fig. 2: middle panel) that measures continuously the magnetic field strength. Both instruments have a sampling rate of 1 Hz 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

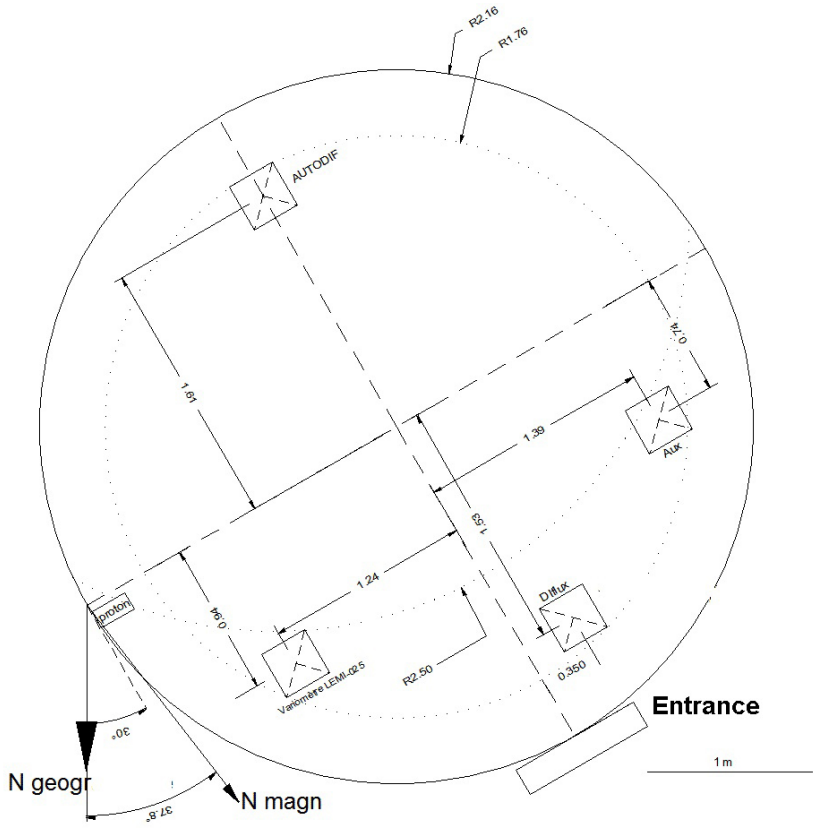

Figure 6. Positions of instruments inside the radome.

a timestamp. Because of the limitations of the radome, we do not use Windows PCs but looked into the possibilities of small ARM processors. Thanks to popularity of Raspberry Pi, these small, cheap ARM processors have gained popularity and opened up a whole new market. We decided to use the BeagleBone Black (Fig. 5): a small Texas Instruments single-board computer. It is comparable to 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 (−40–80 °C) is available;
- it comes with two 46-pin GPIO headers;
- it has on board 4 GB 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 m, and the instrument emplacement was optimized to limit interference, which resulted in the layout

Figure 7. Electronics on a shelf in the radome.

shown in Fig. 6. The electronics of both instruments, the network switch and BeagleBones are placed on a shelf (±2.30 m above the ground) as far as possible from the instruments (Fig. 7). All installed instruments and electronics are guaranteed to work at −40 °C except the electronics of the LEMI-25, which 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 and Rasson, 2013) in the next season to establish a completely automated magnetic observatory. As shown in the schematics depicted in Fig. 8, the radome is connected to the Princess Elisabeth station with fibre optics. Princess Elisabeth station has a permanent Internet link

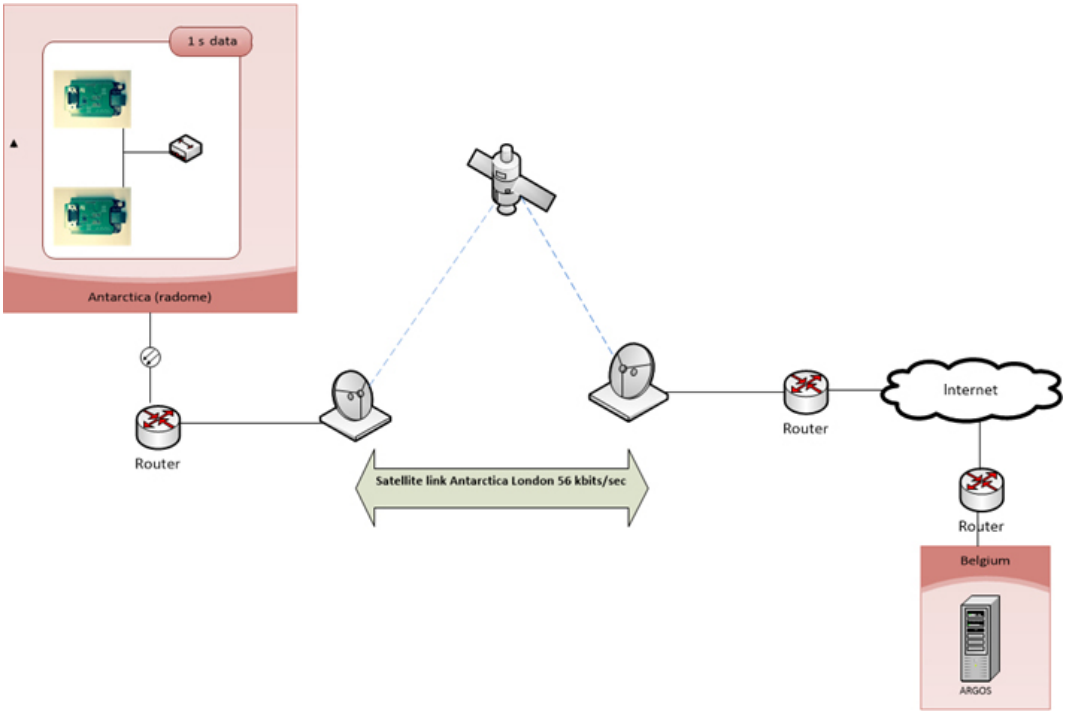

Figure 8. Data transfer.

and limits each scientific project to 56 Kbit s^{-1}. To transfer the data, MQTT is used. A Mosquitto broker is installed in Dourbes (Fig. 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 is 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 1 s data:

- vector instrument LEMI-25 sends a byte message of 16 bytes each second;
- scalar instrument GEM Systems GSM-90 sends a message of 8 bytes each second;
- both use a QOS 1, which guarantees that messages arrive at least once.

The programs on the BeagleBones (Fig. 5) are written to publish the magnetic data remotely with MQTT and log them locally in a file immediately after it is read from the serial port of the device. This results in publishing the data captured in the radome in Antarctica directly to a broker installed on a server in Belgium, while preserving a local file copy. The topic structure used is iagacode/instrumentid/samplerate: pea/lemi0001/sec. This means that this topic receives 1 s vector data of the LEMI-25 located at PEA. The topic tree contains the key that identifies the metadata of the instrument, which implies that the message does not need to contain the metadata. The topic corresponds with an upfront-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 is 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 does not 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 does not respond, the publisher will try to re-establish the connection. In our project the keep alive is set to 10 s. 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 a maximum of 10 s. With this unnoticed stale connection, we could continue to send messages of QOS1 that will never be acknowledged by the broker (maximum of 10 because of the keep alive, which assures the stale connection is closed after 10 s). After the connection is re-established, these messages will be sent 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 these not-sent messages. To cope with this problem, we just used a memory queue which stores 1 h of 1 s data. This means that from now on we can live with connection failures up to 1 h (keep in mind that we lost our real-time promises here). Once a connection is re-established, the 1 h of not-transmitted data is sent to the broker.

Every day completeness of data is checked; if data are missing, an automated procedure to transfer the missing data file will be implemented in the future. For the moment this FTP fallback procedure is a manual procedure.

So during 1 year of 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 were lost ± 10 times a month (small failures of a couple of seconds), which are re-established and data are resent without any problem. During 1 year of experience we had only one big connection lost mainly because of the fact that a fibre-optic cable in Belgium got broken. It took 2 days to recover the connection. To recover these 2 days of data, we fell back on the standard FTP of the recorded files in Antarctica.

5 Conclusions

Realizing near-real-time data transfer today is feasible with standard open protocols and open-source tools. It does not 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 decisions taken, we can see that today MQTT has evolved to a mature near-real-time data transfer protocol that is widely adopted. Nevertheless we can not 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 (WSNs), with the aim of extending the MQTT protocol beyond the reach of TCP/IP infrastructure for sensor and actuator solution.

- CoAP (Constrained Application Protocol) is a specialized Web transfer protocol for use with constrained nodes and constrained networks in the Internet of Things. The protocol is designed for 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 realization of the first fully automated magnetic observatory with near-real-time data transfer.

Data availability. No data sets were used in this article.

Competing interests. The authors declare that they have no conflict of interest.

Special issue statement. This article is part of the special issue "The Earth's magnetic field: measurements, data, and applications from ground observations (ANGEO/GI inter-journal SI)". It is a result of the XVIIth IAGA Workshop on Geomagnetic Observatory Instruments, Data Acquisition and Processing, Dourbes, Belgium, 4–10 September 2016.

Edited by: Christopher Turbitt
Reviewed by: Simon Flower, Charles Blais, Roman Leonhardt, and one anonymous referee

References

Banks, A. and Gupta, R.: MQTT version 3.1.1, Organization for the Advancement of Structured Information Standard (OASIS), http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html (last access: August 2017), 2015.

Cambridge online dictionary: http://dictionary.cambridge.org/, last access: May 2017.

Gonsette, A. and Rasson, J.: AUTODIF: Automatic Absolute DI Measurements, in: Proceedings of the XVth Iaga Workshop on Geomagnetic Observatory Instruments, Data Acquisition and Processing, Extended Abstract Volume, Real Instituto y Observatorio de la Armada en San Fernando, Boletín Roa, no. 3/2013, 186–188, 2013.

Lviv Centre Of Institute for Space Research Intermagnet 1-second standard flux-gate magnetometer user manual rev 1.6, National Academy of Ukraine at Lviv, Lviv, October 2014.

IRIS – Incorporated Research Institutions for Seismology: http://ds.iris.edu/ds/nodes/dmc/services/seedlink/, last access: May 2017.

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

Mesnil, J.: Mobile and Web Messaging, O'Reilly Media, Inc., 2014.

Rasson, J. L. and Gonsette, A.: The Mark II automatic diflux, Data Sci. J., 1, IAGA169–IAGA173, https://doi.org/10.2481/dsj.IAGA-24, 2011.