International Journal of Future Generation Communication and Networking
Vol.7, No.1 (2014), pp.235-248
http://dx.doi.org/10.14257/ijfgcn.2014.7.1.24

Flexible Architecture for Internet of Things Utilizing an Local Manager

Patrik Huss, Niklas Wigertz, Jingcheng Zhang, Allan Huynh,
Qinzhong Ye and Shaofang Gong

Department of Science and Technology, Linköping University 601 74 Norrköping, Sweden Tel: +46 11 363020, Patrik.huss@liu.se

Abstract

This paper presents an flexible architecture for Internet of things utilizing a local manager with the goal to solve many issues. The solution is described from both the system architecture and example applications. The component design and the communication between these components are introduced. The Local Manager architecture is composed of a gateway, Message Broker, Message Relay Bridge and several small applications (Apps) with different purposes. The Local Manager can be used as a platform for future integration of things into cloud services via the Internet.

Keywords: *Gateway; Message Broker; Wireless Sensor Network; Internet of Things; AMQP*

1. Introduction

Many implementations of Internet of Things (IoT) exist but this paper focuses on an architecture having a central unit (cloud server) running web applications for dual-way communications with both remote sensors and actuators. The sensor networks will be remote, running on mobile Internet connections that may have irregular drops in the Internet connection. One goal with IoT is that low cost devices may be connected to and accessible from the Internet. The function can be sensor readings, sending control commands or receiving alarm messages. A Wireless Sensor Network (WSN) can be utilized for local applications so it may operate without a gateway. But for IoT usage there is a need of gateways [1]. For example, energy companies need a way to remotely read their meters [1] [2]. This type of usages demands high security in every part of the system. However, the Internet was not built for highly secure communications therefore methods like Secure Socket Layer/Transport Layer Security (SLL/TLS) are added over the ordinary TCP/IP protocol used in the Internet [3].

Today, most of the development for end users is done in the form of cloud services and mobile applications. For IoT applications it could be expected that the services for the end users will be done in a similar way, but the "things" may have many protocols. They can be located in rural areas where the Internet connection is unreliable. One way to solve these problems is given in this paper.

WSNs are often used for IoT purposes with the following features.

- Low data rate
- Low cost
- Limited resources

ISSN: 2233-7857 IJFGCN

- Often battery-operated
- Many different applications
- Many different network standards (ZigBee, 6lowPAN and Wireless HART, *etc*.)

If we study a typical IoT system with a WSN, it is in reality a flow of messages/data that needs to be transmitted to correct destinations. Destinations can be sensors, actuators, cloud services or mobile phones, etc. This leads to the fact that the system needs to have a reliable message structure. This paper suggests a system for IoT which solves many of the security, reliability and stability issues that follow with low cost wireless sensor networks connected to the Internet.

2. IOT Architecture and Use Cases

Many architectures for IoT look like that in Figure 1 [4] where WSN is deployed in different locations, running different applications. A gateway relays information from WSN into a cloud server where the user accesses the data from sensors or sends control commands to actuators. In this paper we introduce a so-called Local Manager instead of a gateway.

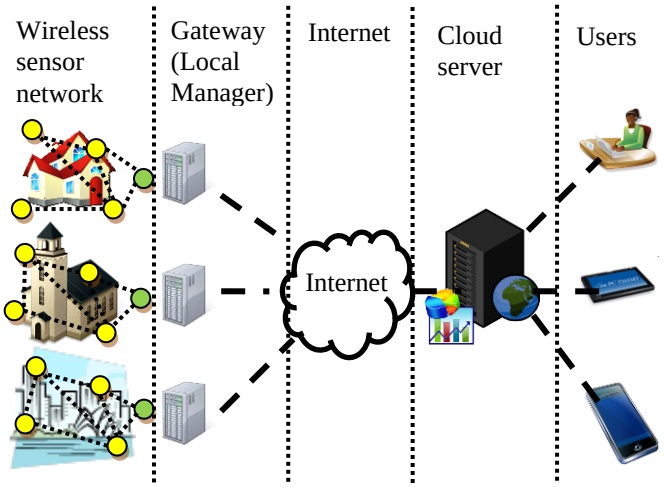

Figure 1. IoT Architecture

Case 1 shown in Figure 2 is a typical scenario used by energy companies to perform remote data collection from electric meters where every data log entry is triggered from a remote location. The energy company sends a Poll command and directly gets a response from the meter with the measurement. Due to the need of a remote trigger to initiate the data log reading, no data log is created when the Internet connection is unavailable. This leads to an unreliable solution for sensor data collection.

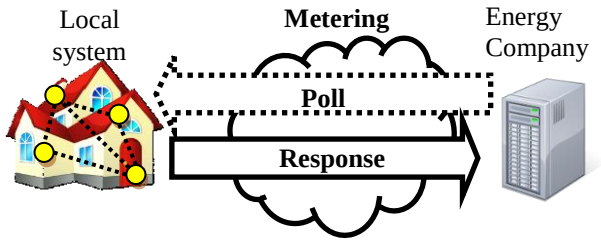

Figure 2. Metering

Case 2 shown in Figure 3 is the scenario where the sensor is preconfigured to measure, create a data log and report with a specific interval. The local system pushes the message to a remote database and directly gets a response of success. The local system can also poll for commands and get response. If the Internet connection is disconnected each sensor must buffer data to prevent data loss.

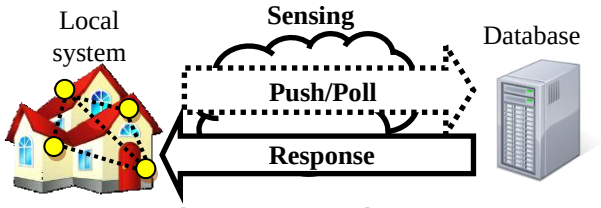

Figure 3. Logging Data

Case 3 shown in Figure 4 is the scenario where a heating system set points or control commands are remotely configured by a user. A heating system can also send status messages to the user in real-time. Dual-way push messaging is used for sending messages.

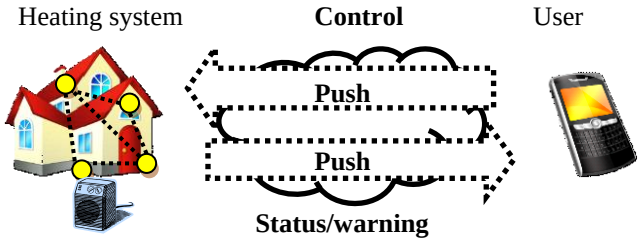

Figure 4. Remote Control

Utilizing push messaging leads to lower latency in the transfer of commands and other data compared to poll messaging because each message is sent directly to the destination. The latency in the poll technique is directly related to the poll interval and dense poll intervals leads to more data traffic to be consumed.

A WSN connected to the Internet has high complexity. Many developers strive for interoperability between all devices utilizing a common network layer. A typical example is 6lowPAN [5] where the integration seamlessly works with IPv6 [6]. It has many possibilities to solve problems when everything utilizes the same network layer, but this could cause problems when having devices exposed directly to the Internet. Especially, those resource-constrained devices driven by small batteries have difficulties to build in security measures.

Many people are looking into 6lowPAN as a solution for IoT but it mainly utilizes an adaptive network layer to remove the need of complex gateways and make it easy to connect

the device to the Internet. The goal in this study is to suggest an architecture that helps developers to easily add new functionality to the IoT network. Also the architecture should be both flexible and reliable. The Message Relay Bridge proposed in this paper solves many problems with the need to build a secure dual-way connection between one cloud service and one Local Manager suitable for unreliable mobile Internet connections. The Message Relay Bridge can relay all messages to or from the cloud service with a dual-way push technology. It also filters information from WSN to reduce the amount of data traffic over the Internet. It also opens the possibility to share information and receive data from a third party system by adding a second customized Message Relay Bridge.

3. Function of Local Manager

The goal for the Local Manager (see Figure 1) is to solve reliability problems associated with the mobile Internet and be flexible for many applications. But first the functions needed for high reliability and flexibility should be identified, as listed below.

- Middleware for reliable message transport
- Gateway function
- Secure-transport over the Internet
- Multi-language support
- Modular design
- Traffic reduction

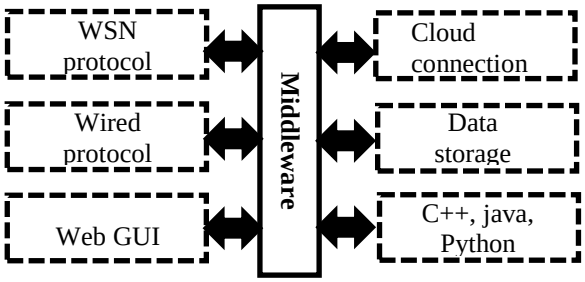

Figure 5. Functions of Local Manager

Any system, no matter it is a system for IoT, WSN or M2M communications can be considered as a box with inputs, outputs and internal messages/logic. This leads to comparison of the operations of a computer operating system or mobile operating systems where each application shares resources from CPU, memory and sensors. When abstracting an IoT system to be a block, with inputs, outputs, internal message flow and logic, the coding of a program can get large, complex and it easily goes out of control. It is better to utilize a modular concept where each function is divided into small programs so that each program can be tested individually and reach high stability. But when many different systems and programs need to be interconnected problems often appear. For instance in computer operating systems a lot of different programs for various applications run at the same time, sharing the same resources. If we look at sensor data and Internet services in a similar fashion

we see that there is a need to share information from sensors and systems to applications and web services at the same time.

To solve these problems with sharing resources with different programs a good middleware shown in Figure 5 must be used. Here we propose a design around an Open Source Message Broker (MB) as middleware [7]. MB has been used in large scale applications to route messages to one or many destinations. Many mobile phone applications use them to keep connection with a central cloud server [8]. But destinations do not need to be limited to remote destinations (clients). A destination can be a program in the same physical device which makes the MB a good candidate for exchanging messages between multiple applications programmed in different programming languages and/or different protocols. Therefore it is best to build the structure around a stable architecture. The goal is to share information between sensors and applications, or between applications.

Because of many different network protocols and IoT solutions, there will be a need for gateways between hardware, network protocols and application protocols. Therefore the Local Manager needs to be flexible so that it can add more gateway solutions in the future. Some sensor data from a sensor network is more important (depending on applications) and should never be lost; it should always be saved in nonvolatile memory (secure storage). However, other unimportant data need not always be saved. This demands that the system has possibility to filter out important data. The local manager also needs a flexible way to connect it to cloud services to either receive or send messages.

Because of many different use cases for IoT, applications, protocols and programming languages, the Local Manager needs to be flexible in managing these issues.

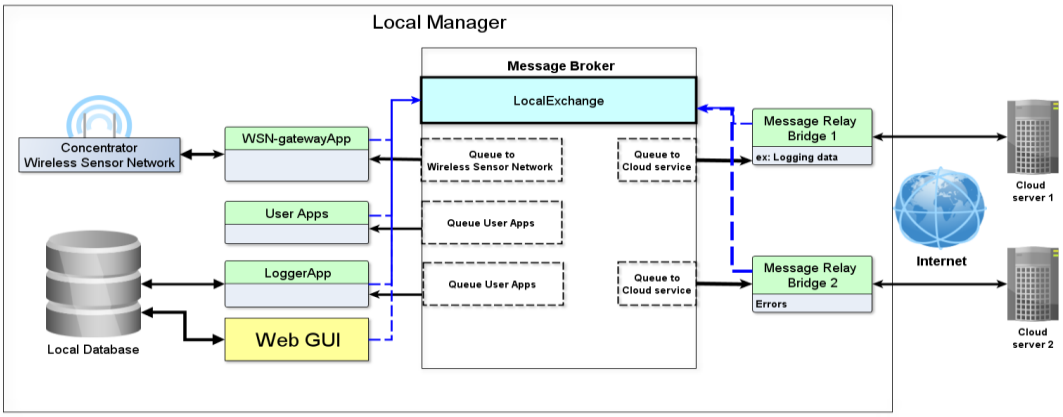

Figure 6. Architecture of Local Manager

4. Design and Implementation

Figure 6 illustrates the Local Manager architecture that was implemented. The Message Relay Bridge was developed to relay data from a cloud service with the goal to have secure dual-way real-time communications.

The usual way of connecting to MB is that the developer designs what is called a client application that either consumes messages from the MB or publishes them to the MB as illustrated in Figure 7. This still gives a problem in buffering messages in the client when the Internet is unavailable. Our solution is to put an extra MB at the client side as illustrated in Figure 8 to be used as middleware and data buffer.

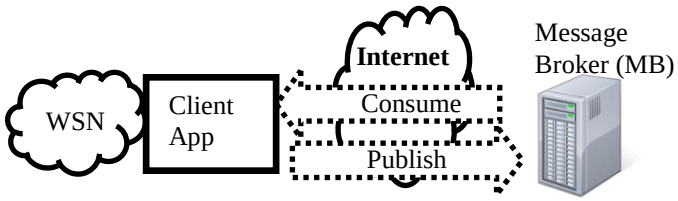

Figure 7. Client -MB Connection

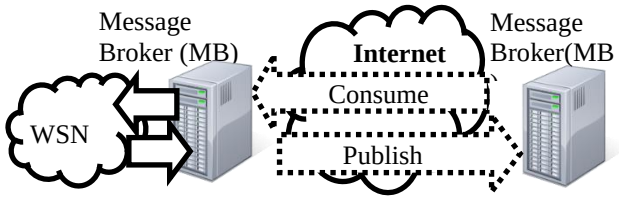

Figure 8. MB Client -MB Connection

4.1. WSN-gateway Application

A WSN-gateway acts as a gateway between the local MB and the concentrator of the WSN. In our solution (see Figure 6), the application searches for an available WSN concentrator, verifies the connection and creates a dual-way connection between the WSN and the local MB. The MB exchange will route the data originating from the WSN to the correct applications on the Local Manager, depending on the type of messages.

4.2. Web GUI

Instead of designing a specific Graphical User Interface (GUI) to the Local Manager, a Web GUI was developed for configuration and interaction with the system, see Figure 6. It reads data from the database and publishes commands to sensors through the MB middleware. The Web GUI was developed in a modular fashion for future reuse of codes and ease of development. It was developed utilizing a Python framework for security [9] named Django [10].

4.3. Message Broker as Middleware

Many types of MBs exist in the market, but for our purpose RabbitMQ was the candidate because of its maturity and open source availability. It has built-in support for many protocols and is designed for managing large number of connections, high data throughput and versatile support of different operating systems.

RabbitMQ utilizes a message protocol named Advanced Message Queuing Protocol (AMQP) [11] which is a standard for message exchange between different systems. It has a well-defined structure focusing on the following issues.

- Security
- Reliability
- Interoperability
- Open standard

The MB follows the concept of publish or subscription according to the AMQP protocol, but it has many different configurations. For our solution the topic configuration was the most suitable for flexibility. An application connects the MB, registers to an exchange with the help of topics (types of messages). The MB will create a queue for the program where all messages tagged with a correct topic are buffered. The program can now consume the messages directly or perform it periodically depending on the application. For example, warning messages may need real-time processing and therefore consume the messages direct. Historical messages as stored data need not to be processed directly and can be buffered in a queue for later processing.

When a program publishes a message to the MB exchange it tags the message with a topic about what type the message is and the MB will send it to the correct queues. Configuring a message with durability mode tells the MB to save the message into non-volatile memory so that intermittent power failure will not lead to data loss.

4.4. Message Relay Bridge

Message Relay Bridge shown in Figure 9 connects to both the local MB, used as middleware and the cloud MB. Each Message Relay Bridge is connected with both an uplink and downlink connection. Figure 9 illustrates how the message is transferred from the Local Manager to a remote server and also how messages are sent from the remote server to the Local Manager. For example, the Message Relay Bridge consumes one message from the CloudSendQ and publishes it to the remote cloud exchange. If the publication fails because of Internet failure, the message is still saved in the Local Manager CloudSendQ because the message has not yet been acknowledged from the Message Relay Bridge. The detailed Message flow between Message Relay Bridge and remote cloud server can be seen in Figure 10.

Because of the usability of the AMQP with topic configuration, The Message Relay Bridge can be configured to only relay specific types of message and can therefore easily reduce the amount of data sending over the Internet for applications utilizing the mobile Internet. The Message Relay Bridge can also be used to share selected data with third party systems in a controlled manner. For example, it can be used to share temperature data from a WSN, with an external analytics company.

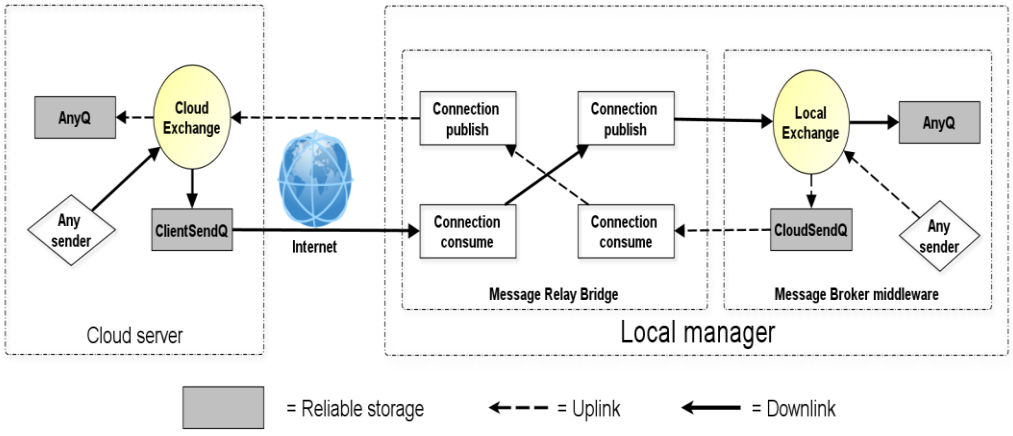

Figure 9. Message Relay Bridge Dual-way Connection

4.5. High Reliability of Sensor Data

Reliable message flow is important when connecting different hardware and sensors to different websites or cloud services. When many small systems are connected, it is important to identify every step in the message transport where the message could be lost, due to Internet failure, power failure or program crash, *etc*. The goal should be that every secure storage (non-volatile memory) is responsible for the delivery to the next secure storage.

For example, Figure 10 (a-c) illustrates a design of the whole message flow from a sensor with the goal not to lose collected data. The sensor wakes up, takes a measurement and sends it to the concentrator. The concentrator relays the data to the gateway application which publishes the data into the local MB and is saved in a queue for secure storage. An acknowledgment is sent to the sensor that the measurement is now safe. As seen in Figure 10 (b), if the Internet connection is available the Message Relay Bridge will consume the message from the local MB and publish it to the cloud MB through an encrypted SSL/TLS connection. When the message is successfully published into the cloud MB it is removed from the local queue. When the message reaches the cloud MB queue, the DataloggerApp will now consume the message from the cloud MB and save it into the selected database, see Figure 10c when the message is successfully delivered to the database it is removed from the queue.

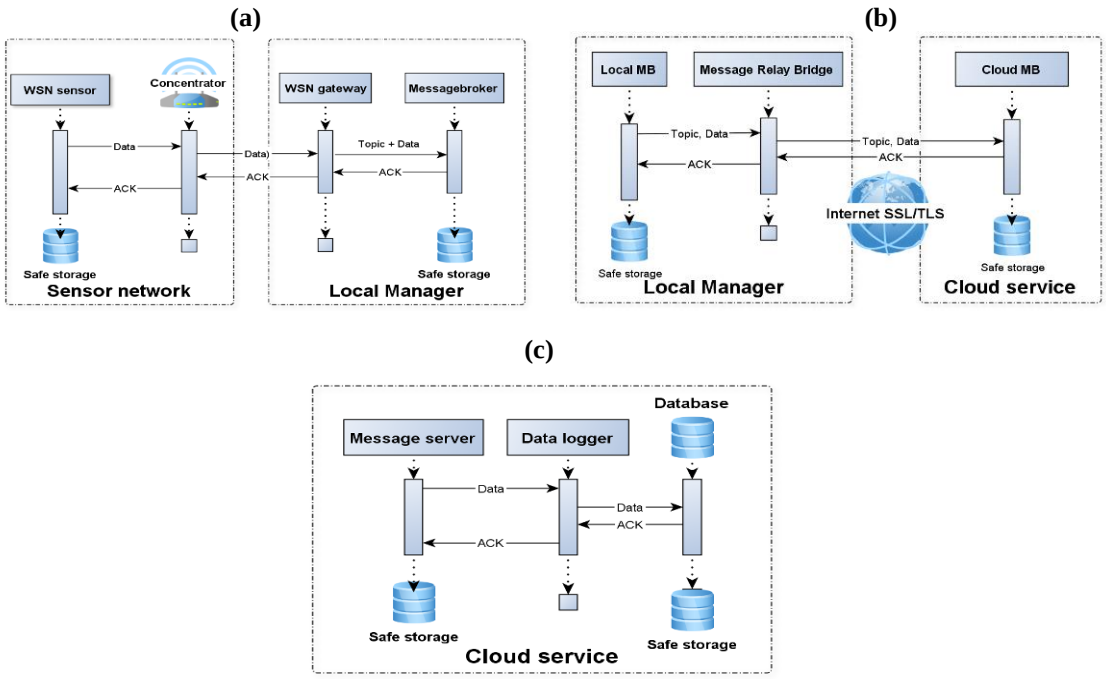

Figure 10. Reliable Message Flow

4.6. Cloud Server with Web Application

A webpage is a source of information and is in most cases static information. However a web application which is constantly changing can be more advanced. Information is automatically updated and users get information personally directed to them. A web application can also perform background events. A cloud server with a web application was developed for testing the Local Manager, see Figure 11. An MB is set-up to allow connection from the Local Manager. The sensor data from the WSN is published into the local exchange

and relayed into the SensorData queue. An application named LoggerApp is active on the server; it consumes each message from the queue and writes it to the MySQL database. A Web GUI built on a Python-based web framework manages the visualization of sensor data with a graph module. Also a control feature to switch an actuator in the WSN was developed.

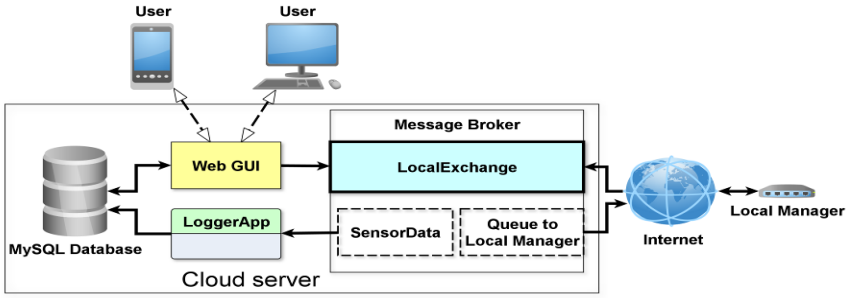

Figure 11. Design of Cloud Server

4.7. Flexible Solution

Because of many different applications for IoT solutions, some will be provided with cloud services. Even more general a Web GUI is developed for the user to monitor its system or utilize the function an IoT system provides. Other interactions will be done by buttons, e-mail, or smartphone. With the help of the Local Manager the integration to different cloud services can be simplified. The Local Manager also works for IoT solutions which have demands on local access only to the WSN through a smartphone App or Web GUI by utilizing a Local Area Network not connected to the Internet. Many users of applications for IoT could be reluctant to host their services at a third party cloud service. Therefore the Local Manager can run locally with local web services without any access from or to the Internet. To be as flexible as possible the suggested Local Manager architecture can add many applications on run-time (without disrupting already running applications). Mobile device can connect directly to the Local Manager if needed and the Local Manager with the help of Message Relay Bridges can also connect many cloud services or destinations utilizing many protocols.

5. Results

The developed Local Manager can be utilized as a development platform for web applications with integration of WSNs.

It has the following features:

- Reliable message relay
- Data storage
- Gateway between different protocols
- System monitor
- WebGUI
- Data buffer for failures

5.1. Hardware

Figure **12** shows the Local Manager running on Beaglebone Black ARM based hardware [12]. On top is a developed adapter to our own developed Generic WSN module [13]. The Local Manager is connected to the Internet through a Mobile Internet dongle.

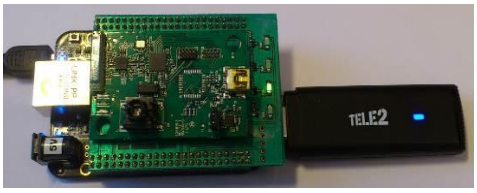

Figure 12. Local Manager Implementation

Figure 13: Remote Control Test, Heater

Implementation Figure 13 shows the test case where a portable heater is controlled from the cloud service. The control box in the figure is connected to the WSN and is self-registered with the cloud service.

5.2. Software

The software for the Local Manager was developed on a Linux Operating System (OS) utilizing the script language Python. Python has advantages in its ability to run on multiple OSs and therefore the architecture should easily be modified to run on other OS like Windows. Each application is developed to run as a background service in the OS and utilize as low resources as possible.

5.3. Web GUI and Cloud

Figure 14 shows the Local Managers Web GUI. It has three choices, Analyze, Settings and Controllers and has the same functionality as the Cloud service.

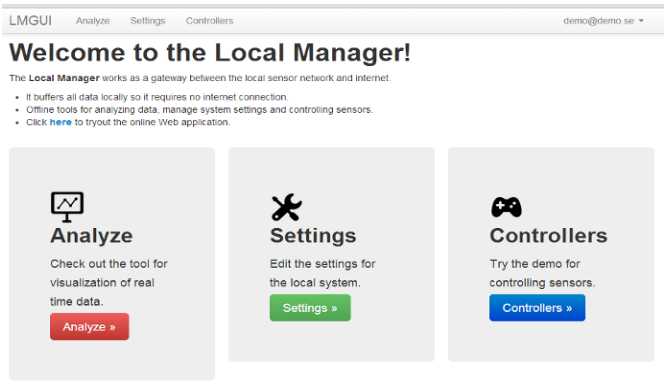

Figure 14. Web GUI Local Manager

With the Analyze function shown in Figure 15 (showing Cloud service), one can monitor the measured data flowing in from the WSN in real-time. It is possible to change the time

interval and do basic analysis on the measured data. When a new sensor device for collecting measurements is registered with the WSN, it will automatically be marked as a new device in the list and all database registrations will be created. Also the status of the connection to the Local Manager is visualized in the top right corner.

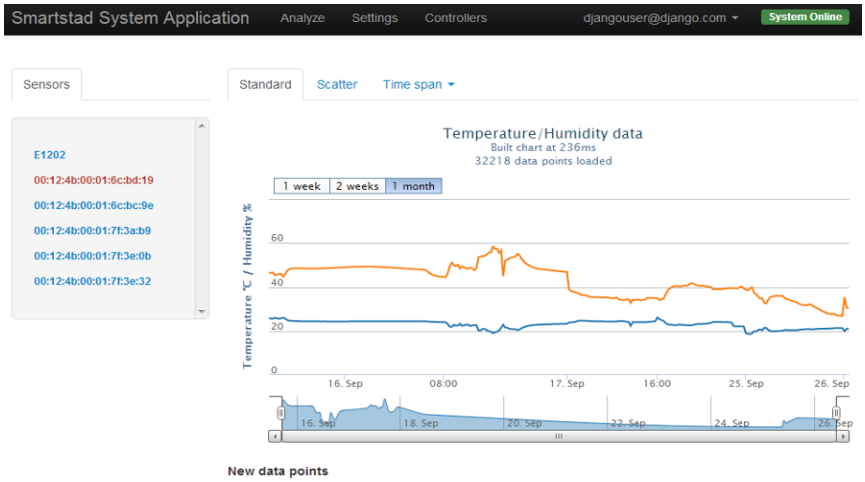

Figure 15. Cloud Service with Data Analyzes

With the Controllers function shown in Figure 16 one can initiate control commands to any registered control devices in the WSN. When pressing the buttons the message is sent to the control box in the WSN and when the response from the control box is sent back, the user is notified of execution.

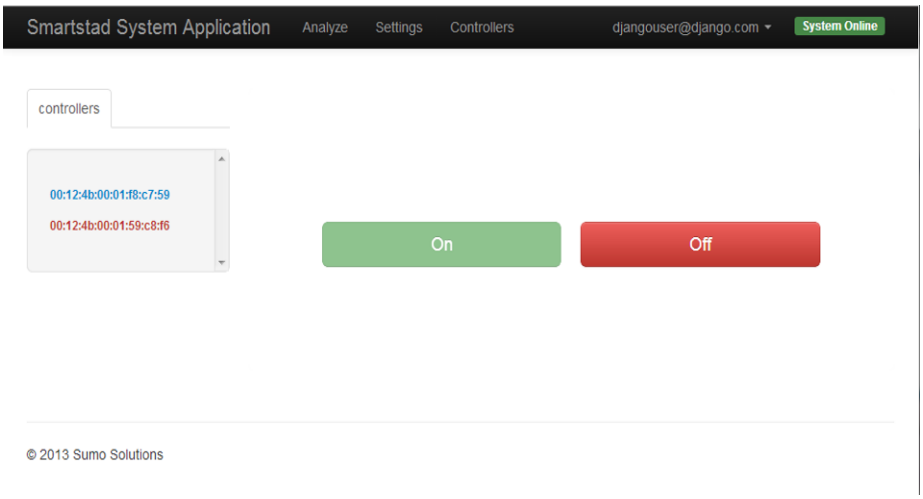

Figure 16. Cloud Service with Remote Control

5.4. Traffic Data Reduction

Comparing the Message Relay Bridge with the real-time dual-way push communication implemented as Figure 4 to the pull communication illustrated in Figure 3 leads to the results listed in Table 1.

Table 1. Data Traffic Usage

	Test case Overhead	Overhead traffic/h	Overhead traffic/month
1	Pull technic	1MB	800MB
2	Keep connection (push)	0,6 kB	720 kB
	Real test usage	**Traffic/h**	**Traffic/month**
3	10 sensors (15 min interval) (push)	164 kB	11MB

Pull communication with an interval of 5s to fetch a control command from the remote cloud server utilizes approximately 800 MB/month in overhead traffic. The Message Relay Bridge with real-time dual-way push technology utilizing only about 0,7MB in overhead traffic

A test case of 10 sensors collecting temperature, humidity and timestamp at a 15 minute interval can be seen in Table 1. Each data point is pushed up to the cloud service with the Message Relay Bridge in real-time. The data traffic consumption in one month reaches only 11MB.

6. Discussion

This verified solution of a Local Manager has reached the goal being suitable for a mobile Internet connection which may have irregular connection drops. However the implementation of AMQP solution still adds considerable overhead to each message. Nevertheless data can now be analyzed in the Local Manager instead of sending everything over the mobile Internet connection. The functionality that specific data can be filtered out gives it a considerable advantage in flexibility. The data traffic comparison of the new solution with the previous solution where poll techniques were implemented shows great improvement. It is now easy for a web developer to publish a message into the MB and the Message Relay Bridge will then receive the message over an SSL encrypted link over the Internet. However, future work should be done in the message protocol to make it even more flexible and also more work should be spent in testing MQTT as a candidate for the Message Relay Bridge for reduced overhead traffic.

Because the developed Local Manager has all components needed for most web design to interact with hardware, it can be configured such that a mobile phone App can directly connect to the Local Manager; this means that a cloud service is not needed at all in such a configuration.

7. Conclusion

The Local Manager concept with its reliable message architecture, modular design and many options for local applications works well as a flexible solution for Internet of Things. The developed Message Relay Bridge also solves many problems concerning the reliability and flexibility of remote deployment of Wireless Sensor Networks connected to cloud services, utilizing mobile Internet connection that may have irregular connection drops.

Acknowledgements

The municipality of Norrköping in Sweden and the Swedish Energy Agency are acknowledged for financial support of the study which leads to this paper. Pär Eriksson and Oscar Ivarsson, two students at Linköpings University are acknowledged for their assistance in the web development.

References

[1] Q. Zhu, R. Wang, Q. Chen, Y. Liu and W. Qin, "IOT Gateway: Bridging Wireless Sensor Networks into the Internet of Things", 2010 IEEE/IFIP International Conference on Embedded and Ubiquitous Computing, **(2010)**.
[2] P. McDaniel and S. W. Smith, "Security and Privacy Challenges in the Smart Grid", IEEE Security and privacy, **(2009)** May-June, pp. 75-77.
[3] P. K. Lee and L. L. Lai, "A practical approach of smart metering in remote monitoring of renewable energy applications", Power & Energy Society Meeting 2009. PES '09, IEEE, **(2009)** July.
[4] R. Oppliger, "SSL and TLS Theory and Practice", Artech House, **(2009)**.
[5] J Zhang, A. Huynh, Q. Ye and S. Gong, "Design of the Remote Climate Control System for Cultural Buildings Utilizing Zigbee Technology", Sensors & Transducers Journal, vol. 118, no. 7, **(2010)** July, pp. 13-27.
[6] 6LowPAN (IPv6 over Low power Wireless Personal Area Networks), Specification 6LoWPAN IETF group, RFC 6282, http://www.ietf.org.
[7] IPv6 (Internet Protocol version 6), Specification IPv6 IETF group: http://www.ietf.org/.
[8] RabbitMQ,V3.1.4 07, available from http://www.rabbitmq.com (Open source software), **(2013)** August.
[9] Google Cloud Messaging, "Google Cloud Messaging for Android", available from http://developer.android.com/google/gcm/index.html, **(2013)** October.
[10] Django security features, available from http://www.djangobook.com/, **(2013)** July.
[11] Django (The Web framwork for perfectionists with deadlines), available from https://www.djangoproject.com/, **(2013)** July.
[12] AMQP, (Advanced Message Queuing Protocol), V 0-9-1, released 13 Nov 2008, available from www.amqp.com, **(2013)** July.
[13] BeagleBone Black, "ARM based embedded computer platform". http://beagleboard.org/Products/ BeagleBone+Black/, **(2013)** October.
[14] A. Huynh, J. Zhang, Q. Ye and S. Gong, "Zigbee Radio with External Power Amplifier and Low-Noise Amplifier", Sensor & Transducers Journal, vol. 118, no. 7, **(2010)** July, pp. 110-121.