Participatory Cloud Computing: A Smart Middleware for the Internet-of-Things

T. Mullins, J. Martin & A. Bagula

ISAT Laboratory, Department of Computer Science,
University of Cape Town, South Africa.
MLLTA001003@myuct.ac.za, MRTJAR001@myuct.ac.za,
bagula@cs.uct.ac.za

Abstract. Low power devices are crucial for use in wireless sensor networks in that they are compact and have very low power requirements. This paper reports on the design and implementation of a participatory cloud computing middleware that uses a mesh of low power devices as gateways for a wireless sensor network. We consider a middleware model where the sensor data collected in the field is delivered to the participatory cloud via various delivery mechanisms and protocols. While the participatory cloud computing middleware is assumed to be made up of a collection of low power nodes, we conducted testbed experiments using a heterogeneous mesh infrastructure consisting of both low power and standard desktop equipment to demonstrate the effectiveness of the solution in both standard and low power computing environments. The experimental results reveal that data is reliably sensed, transmitted and securely stored in databases in our participatory cloud.

Keywords
Internet of things, Smart Middleware, Participatory Cloud Computing, Intelligent Platforms.

1 Introduction

Cloud Computing has advanced to a stage where it is being adopted by enterprises and individuals at a rapid pace. However, this rapid adoption has not been able to adequately mitigate its own shortcomings. These include issues related to the ever rising carbon footprint resulting from the immense cooling requirements for these large data centres, the lack of privacy controls and vendor lock-in. These are some of the traits these systems exhibit that may raise security concerns. These issues make moving towards a cloud environment an unfavourable solution for users that have low power constraints, stringent privacy requirements or bandwidth issues. We take as an example rural communities in developing countries, where telecommunication facilities are not present and an intermittent electrical power supply, the difficulty in providing services in these locations require high setup and maintenance costs. This is usually because not

only does dedicated hardware usually need to be set up to host the services, infrastructure needs to be put in place thereby making the entire solution unfeasible for most case studies.

Wireless Sensor Networks (WSNs) are a family of wireless networks which have been designed to provide a bridge between physical and virtual worlds of information. They are currently deployed in our daily life to sense what is happening in our environment and report to a processing gateway where appropriate decisions and actions are taken about the environment which is being monitored. They provide a great opportunity to bridge the digital divide [1] as they are built around least cost and cheap-to-maintain devices to construct unattended networks. They are deployed in installations of thousands of nodes in the harsh environments of the developing world with either intermittent power supply or harvested energy through solar, wind or other energy scavenging methods. Many applications are emerging from the industrial, environment, health-care and energy fields where objects manipulated in these fields are being outfitted with sensor and RFID devices and endowed with an Internet address to become smart objects of an *"Internet-of-Things (IoT)"*. This allows these devices to become globally discoverable and queried; and similarly, discover and interact with external entities by querying humans, computers and other smart objects. This paper demonstrates the relevance of using a cloud computing networked infrastructure deployed in the context of a delay tolerant remote IoT transmitting data to a cloud environment which consists of low power devices used as gateways [2] which are networked to form a participatory cloud computing middleware.

2 Middleware for the internet-of-things

In this project we wished to frame the use of the middleware in a rural IOT context. This experiment consisted of us emulating 2 different kinds of networks. First there was the participatory cloud which was connected in our lab environment through various networked computers. In the field this would typically be Alix boards connected to each other wirelessly. Secondly we have our IOT network which would use various opportunistic methods of transmitting data to our participatory cloud. The rationale behind this framework is the need to provide services in rural/remote areas in the developing world that can both enhance our understandings of our environments as a whole as well as providing services that would otherwise be too expensive to otherwise deploy. The opportunistic method of deploying data that we wish to demonstrate in this project and the low cost/power community cloud that we develop allows us, in this context, to deploy an IOT network with a relatively low cost.

3 The Participatory Cloud Computing Middleware

In this project we use these low power devices because they are compact and have a minimal power footprint when compared to traditional computing hardware, and can be easily deployed in remote environments for use in wireless mesh and sensor networks. The problem with these devices is that they have very limited processing power and very limited storage capacity.

This means that in order for us to create or provision value added services from a network of these devices one would need to connect them in a way so that the overall computing resources can be combined.

We decided due to the limited time constraints of the project to approach this in two ways. First we would implement a file storage system that would allow us to aggregate all the storage capacity in the network. Secondly we would gather information on all the system performance values that were relevant to us and allocate services to nodes in the network depending which nodes gave us the best performance.

3.1 Hardware and software configuration

We firstly set up all our nodes in our network. For our project we wished to demonstrate the efficacy of the solution on a myriad of hardware. We thus chose the following system profiles:

- Windows 7 x64 Native
- Windows XP Virtualised
- Ubuntu 12.04 Native
- Ubuntu 10.04 Virtualised
- Gentoo Linux Native
- Debian4Alix

Each system was installed with the base operating system with no additional software except that which was required for our middleware to run. This included:

- Python
- Tahoe-LAFS
- Net-SNMP
- Apache and PHP
- SQLite

On each of these devices we allocate drives or folders that we wish to use as free space to contribute to the greater storage pool of our networked filesystem. To achieve this we used a software solution called Tahoe-LAFS (Tahoe Least Authority File System). The solution allowed us to allocate drives or folders on disk that we could use as well as how much space was allocated to the filesystem. Systems were allocating free storage space anywhere from 512 MB to 4 GB of data. Where this was to be achieved on the Alix boards (these devices come with no readily accessible write storage), we installed an additional USB flash disks.

3.2 File Storage and Services

The Tahoe client node would then connect to a server node, which also acts as a client, and then made this information available to all the other nodes on the network. The server node was only used for introducing client nodes to each other and played no role in storing information. The storage procedure was negotiated between the clients connected in the network.

Files to be stored were disseminated to various nodes in the network while ensuring redundancy should one of the nodes go down. Files are also stored in a way that ensures that the strictest confidentiality and integrity is maintained when files are distributed throughout the network. Next we used SNMP to monitor and collect various system resources from the required hosts. SNMP was running as an agent on each machine and net-SNMP was used to poll and collect the system data from all the hosts in the network.

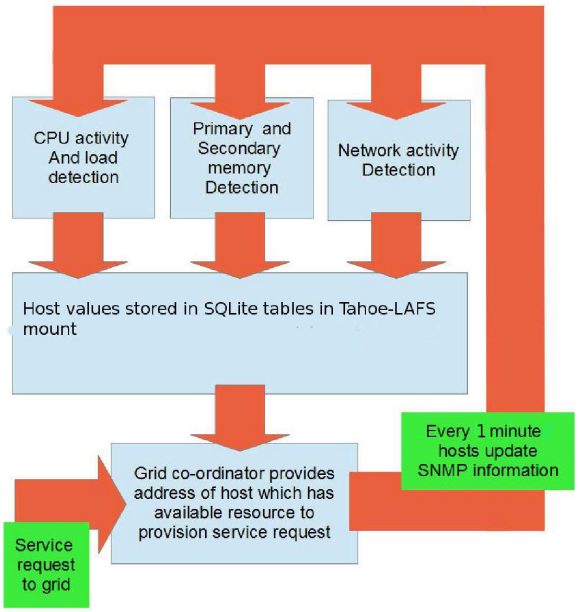

Fig. 1. SNMP system data collection

Data was polled on the all the systems at 60 seconds intervals via a CRON script. This system data was then collected and stored within various SQLite tables. SQLite was chosen as the database provider due to incompatibilities between Tahoe and MySQL. An initial test was done where the MySQL data directory would be stored in the Tahoe-LAFS mount, but due to the fact MySQL is continuously reading and writing to the database caused the entire network to crash within hours of running due to the continuous cryptographic operations that were being performed on nodes in the network. (Every operation performed by Tahoe-LAFS was encrypted) Sample apparent write speeds on an Alix board and a desktop can be viewed in Fig.2 and reasons for why this might possibly be happening.

To demonstrate to users of the system how service requests to the cloud would be handled we created a webserver on all of the participating nodes and pointed its webroot to a folder on we mounted from our cloud filesystem (We exposed the filesystem to the users by userspace kernel modules, namely Fuse[Linux] and WinFuse[Windows]). So

```
[Alix] Write test:
dd count=100 bs=1M if=/dev/urandom of=/ro/root/.tahoe/fs/test
100+0 records in
100+0 records out
104857600 bytes (105 MB) copied, 328.903 s, 319 kB/s
```

```
[desktop PC] Write test:
sudo dd count=100 bs=1M if=/dev/urandom of=/media/disk/test
100+0 records in
100+0 records out
104857600 bytes (105 MB) copied, 6.86004 s, 15.3 MB/s
```

Fig. 2. Comparison of write values on an Alix board (right) and a Desktop (right)

in the end the users systems are configured to select one directory which is used for storage of files in the network. This contains encrypted file fragments which are of no use to the user in its current form. Then a secure connection is made to the network filesystem to an authorized directory which is then mounted on the users filesystem. Only after this has been completed can the user and other userspace applications access the data stored in the network.

Upon a node receiving a request for a particular web service, the system would check in the various database tables stored in the cloud to check the various system performance values to determine which node on the network had the best performance score. The user would then be redirected and the request would then be handled by the selected host. Our score was worked out by adding our raw usage percentages together. Thereby implying that the host with the lowest overall score would have achieved the best score and then be selected to service the user request.

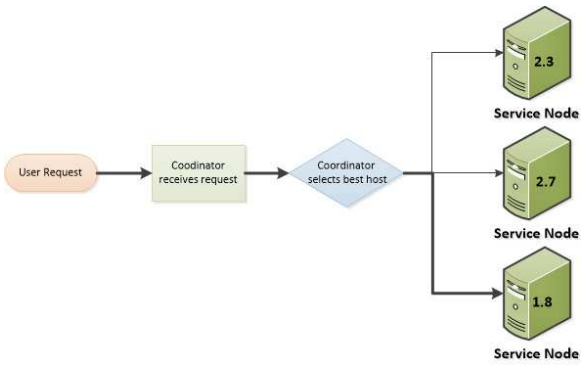

Fig. 3. Network resource request flow

3.3 Wireless Sensor Network

A network of wireless sensor motes were deployed around the campus of the University of Cape Town.

The function of the sensor motes was to provide the participatory cloud with data but most importantly to monitor the environment where they were installed. They took measurements of air contaminants such as Carbon monoxide, Benzene, Toluene and

Hydrogen Sulphide. An experimental network was setup with sensor motes equipped with air contaminant sensors as well as wireless communication hardware. Two sensor motes within the network were setup to use short-range IEEE 802.15.4 compatible hardware for communication and one was setup to operate using a cellular phone network for communication.

In both configurations the sensor data would be transmitted to a gateway for storage and computation in the participatory cloud.

The sensor motes were also be mounted on vehicles so that they become mobile air contaminant sensors. By being mobile the system as a whole can measure levels of air contaminants over a much larger region of a city. Thereby providing a mechanism where sensor data down to individual streets within the city could be measured. Subscribers of the system would then be able to access the data on their own devices, which are part of the Participatory Cloud. This design was chosen to demonstrate the efficacy of data mules in a remote context where perhaps a vehicle that regularly travels through some remote region could be used as a mule in this context to carry around our sensors to detect changes in the environment.

The design of the wireless sensor network was driven by the low energy constraints, low processing capabilities and low memory capacity of the sensor motes. Additionally the sensor motes were deployed in places which may not be easily accessed should there be software or hardware failure or even simply battery depletion.

The two hardware configurations of sensor motes mentioned were chosen to determine the most energy efficient method of communication by comparing power consumption; since energy within the network is extremely limited.

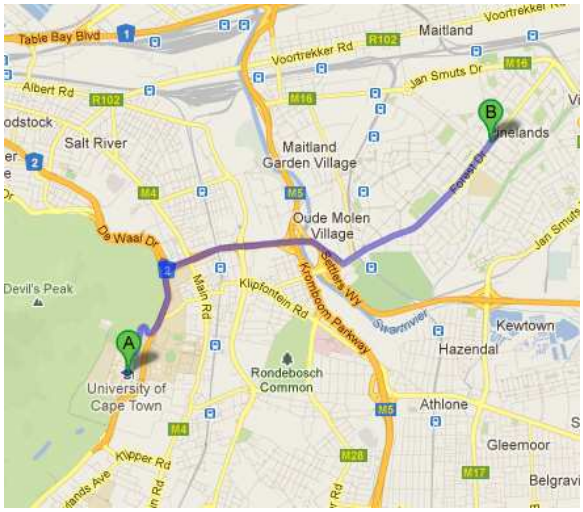

Fig. 4. Route taken by data mule

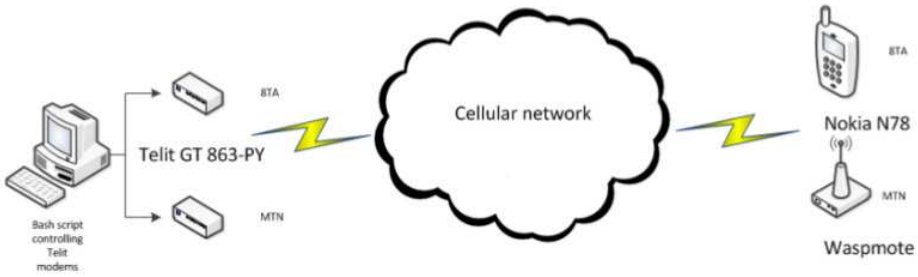

Fig. 5. SMS communication between the TELIT modems, N78 and Waspmote

In the experimental network, communication between sensor mote and the gateway happens as follows; where the short range IEEE 802.15.4 protocol is used, communication is done opportunistically. This means that when two sensor motes come within communications range they exchange sensor data with the hope that either mote is on its way to the gateway. Where SMS communication is used the sensor mote communicates directly with the gateway which is a computer with a GPRS modem configured to receive all information via SMS.

The test bed resulted in a complex array of hardware devices being used. This included

- Arduino Uno
- Waspmotes
- SD card(temporary storage)
- GPS module
- IEEE 802.15.4 radio
- GSM/GPRS radio
- Carbon monoxide, volatile organic compounds and temperature sensors

A mobile service provider had to be chosen for the SMS mote, two mobile providers; MTN and 8ta were selected for an experiment to determine the most reliable operator. The experiment consisted of a GPRS modem connected to a PC and a mobile phone, SMSes were sent from the GPRS modem to the mobile phone for one week. This was done twice, once with MTN and once with 8ta, the ratio of $\frac{received}{sent}$ SMSs was used in determining the best operator for the mobile sensor mote.

The sensor data gathered from both the 802.15.4 sensor mote and the SMS sensor mote were transmitted to a gateway device which then inserted our data into a database residing in our participatory cloud.

4 Performance evaluation

The experimental evaluation of a sensor network as done in [3] is an important process upon which the field readiness of an IoT setting depends. We conducted a number of

experiments using GSM and 802.15.4 to compare the efficiency of both communication models in delay-tolerant IoT settings. During our GSM tests we noticed that the MTN network had 8 times the amount of unreceived SMSes than what we got on the 8ta network. This could be due to the high demand on the MTN network in the areas being tested and the low 8ta subscriber uptake at the time of the tests. Duplicate messages were also sent on the MTN network at times.

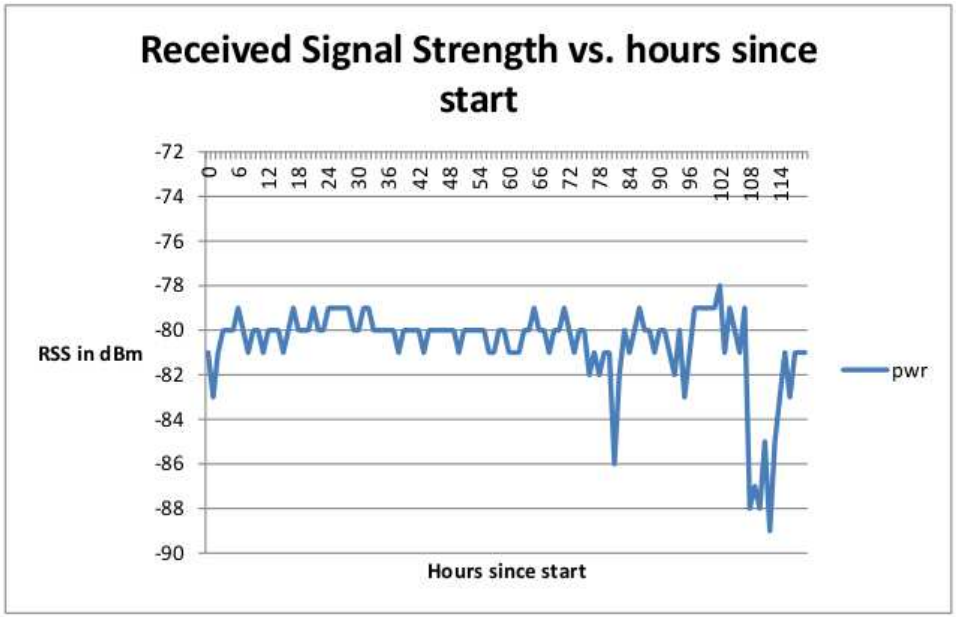

Fig. 6. Graph of RSS vs time of Telit GT 863 using 8TA, measured at UCT upper campus

We were able to send 278 SMSes after which the battery level dropped too low (17% and voltage of 3.1V) to send any more SMS messages. This was contrary to previous research which which reported battery levels around 60% as a cut off for sending messages.

We found that when comparing this result to the results from the IEEE 802.15.4 method of communication, the latter allows for much more communication of messages while consuming less battery life than GSM communication. The XBee 802.15.4 motes were able to send 1500 sensor messages while only consuming 33% of the battery charge, compared to the 278 messages sent via the GSM network.

The experiment concluded with a setup with two motes in two different locations with motor vehicle, with a mote on-board being used to courier the data from the remote sensor network location to the participatory cloud environment.

Upon accessing web pages from our participatory cloud we achieved access times to our data at times that were comparable to standalone webserver access times. We did however discover that the Alix board had extremely long write times when compared

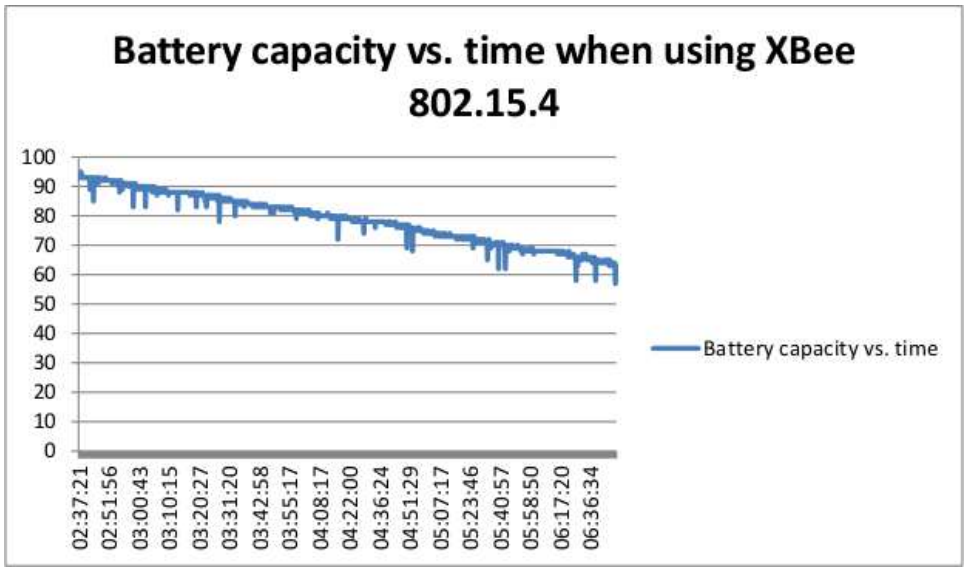

Fig. 7. Remaining battery capacity during experiment

to the performance one could expect from a desktop machine. This did slow down our access times on the device which caused Tahoe at times to crash on the device. For the duration of the experiment we only performed shallow writes to the Alix board, which only wrote to the filesystem running in memory and nothing was committed to storage. So the device was able to participate in being part of the middleware but was unable to reliably commit to bring a provider of secondary memory.

Another reason for the slow writes on the Alix board was that even though the boards themselves had on-board hardware cryptographic accelerators, it was only for AES 256 CBC mode. Which unfortunately Tahoe, at the time of working on this project was unable to support.

5 Conclusion, Summary and future Work

Overall we were fairly satisfied that as a proof of concept that the project achieved its goals and future work will go into reworking the filesystem so that it is able to function in an manner that does not penalise the participating devices like the Tahoe system does. A more dynamic coordinator of resources will also be put in play allowing for more efficient balancing of network resources.

It should also be noted in our findings that the alix boards took extremely long, in terms of comparing raw write times, this could also be to a skewed testing procedure we committed to. If the data being generated from /dev/urandom also places a burden on the CPU we are looking at heavily skewed results in favour of desktop machines in terms of raw performance values. This behaviour too will have to be further examined in future work.

The expansion of the network also needs to be further investigated where we include more Alix boards in the network and also examine the results of constructing the network entirely from these low powers devices.

Solar cells will be introduced to extend the uptime of the motes almost indefinitely, while taking care to further calibrate the gas sensors would result in us getting more accurate readings from the sensors.

References

1. M. Zennaro, B. Pehrson and A. Bagula. *Wireless Sensor Networks: a great opportunity for researchers in Developing Countries*. In the IFIP Proceedings of WCITD2008 Conference, Pretoria, South Africa, October 2008.
2. Marco Zennaro & Antoine B. Bagula, "Design of a flexible and robust gateway to collect sensor data in intermittent power environments", in International Journal of Sensor Networks, Vol. 8, Nos. 3/4, 2010.
3. M. Zennaro, H. Ntareme and A.B. Bagula. Experimental Evaluation of Temporal and Energy Characteristics of an Outdoor Sensor Network". In Proceedings of PAWN-08 Conference, Ilan, Taiwan, September 2008.
4. The NIST Definition of Cloud Computing. "National Institute of Science and Technology, U.S. department of commerce". [Online]. Available: http://csrc.nist.gov/publications/nistpubs/800-145/ SP800-145.pdf
5. A. Marinos and G. Briscoe. "Community Cloud Computing," Computing, vol. 5931, no. December, p. 11, Jul. 2009.