Association for Information Systems
AIS Electronic Library (AISeL)

CONF-IRM 2016 Proceedings

International Conference on Information Resources Management (CONF-IRM)

2016

Service Provisioning for Lightweight Community Cloud Infrastructures

Taariq Mullins
University of the Western Cape, tjma2001@gmail.com

Antoine Bagula
University of the Western Cape, bbagula@uwc.ac.za

Slim Rhekis
University of Carthage, slim.rhekis@gmail.com

Noureddine Boudriga
University of Carthage, noure.boudriga2@gmail.com

59. Service Provisioning for Lightweight Community Cloud Infrastructures

Taariq Mullins, Antoine Bagula
University of the Western Cape
{tjma2001@gmail.com,
bbagula@uwc.ac.za}

Slim Rhekis, Noureddine Boudriga
University of Carthage
{slim.rhekis,noure.boudriga2}@gmail.com

Abstract

Service provision in low power network environments remains a very difficult task due to the lightweight nature of the devices involved. This paper explores and demonstrates the efficacy of the light weight resource allocation protocol (LRAP) in allocating services in a heterogeneous network environment consisting of primarily low power nodes. We show that with the correct resource brokering approach, services can be provisioned on nodes in a way which does not cripple individual nodes in network, while ensuring that services are distributed to nodes in the network where the lowest impact on overall performance will be made. This strategy provides an effective way of providing services especially in low power environments

Keywords

Lightweight Cloud Computing; Lightweight Resources Allocation; Knapsack Algorithm; Service Provisioning

1. Introduction

Community clouds [1] are multi-tenant infrastructures which are shared among several organizations that have shared concerns. They are expected to form smart networked environments delivering critical services with high availability and optimal system performance in order for the applications to execute without delays and interruption. When deployed in Internet-of-Things (IoT) settings, community clouds can be used as backbone infrastructures for ubiquitous sensor networks [2] in diverse applications such as pollution monitoring, precision agriculture [3], water quality monitoring [4] and weather forecasting [5] for drought mitigation [6]. However, with the expected rise in the demand of community cloud computing services and systems, the amount of machines to monitor and manage might become too large a task for an administrator [6] to handle. The design and implementation of systems that can manage themselves autonomously is a solution to this problem. The autonomous service provisioning in community clouds require a) a lightweight cloud monitoring system that provide situation recognition in terms of availability of resources (processing, memory and communication) and b) a lightweight resource allocation process that may be built on optimal allocation techniques to match resources demands to their availability. The Lightweight Network Monitoring Protocol (LNMP) was presented in [7] as a first step towards community cloud management enabling resource monitoring by determining where resources are being consumed in the grid to enable making informed decisions on where to allocate services in the community cloud. While

addressing the first requirement, the LNMP protocol plays an important role in meeting the second requirement as it allows the cloud to allocate services to nodes in response to the performance statistics collected by the protocol.

1.1. Service provision in low power environments

The autonomous service provision in community cloud systems depends on how the cloud system deals with load. One way consists of having multiple servers to spread the load of service requests. The load can be spread by the help of DNS servers using DNS round-robin techniques [8] to spread the load to multiple servers. Using various load balancing techniques works well on networks of heterogeneous hardware to select from, but becomes more complex when the size of the network rises, and more importantly when the hardware profiles of the devices change. Furthermore, it is difficult to assess if each service request that has been made carries the same performance weight on the system since it can happen that every second request made for service carries a heavier or lighter weight than the preceding request. This will result in one of the servers receiving a massive performance drop and thereby "punishing" other users who are currently executing tasks on the machine. Service provision in lightweight network environments remains a very difficult task due to the lightweight nature of the devices involved in terms of processing power as well as the available RAM. When looking at service provisioning in lightweight networked environments, arbitrary allocation of services always becomes an issue because the devices used are much more subject to performance penalties when overburdened with tasks due to their lightweight nature. For these reasons, it makes sense to have a service allocation process that acts on input provided by monitoring systems on the network. This way services can be allocated to various nodes on the network based on their performance values instead of using arbitrary load sharing techniques.

1.2. Contribution and outline

This paper revisits the issue of service provision in cloud infrastructures by presenting, exploring and demonstrating the efficiency of the lightweight resource allocation protocol (LRAP) in allocating services in a community cloud infrastructure made of a heterogeneous network environment consisting of primarily low power nodes. As illustrated by the community cloud architecture in Figure 1, the LRAP protocol is a resource allocation protocol that builds upon a) the LNMP performance monitoring to achieve situation recognition in terms of resources availability, b) a distributed storage and c) the evldns DNS [9] for service provisioning. The LRAP agents use an approximation of the Knapsack algorithm [10] to enable optimal resource allocation. This differs from many resource allocation systems in cloud infrastructures which are based on a myopic or best-effort static approach discounting system optimality.

The rest of the paper is organized as follows: Section 2 presents the community cloud management model by describing its main components and expanding on the LRAP protocol. Section 3 follows with discussions on the experimental evaluation while section 4 covers the experimental results. Our conclusion and a discussion on directions for future research are presented in section 5.

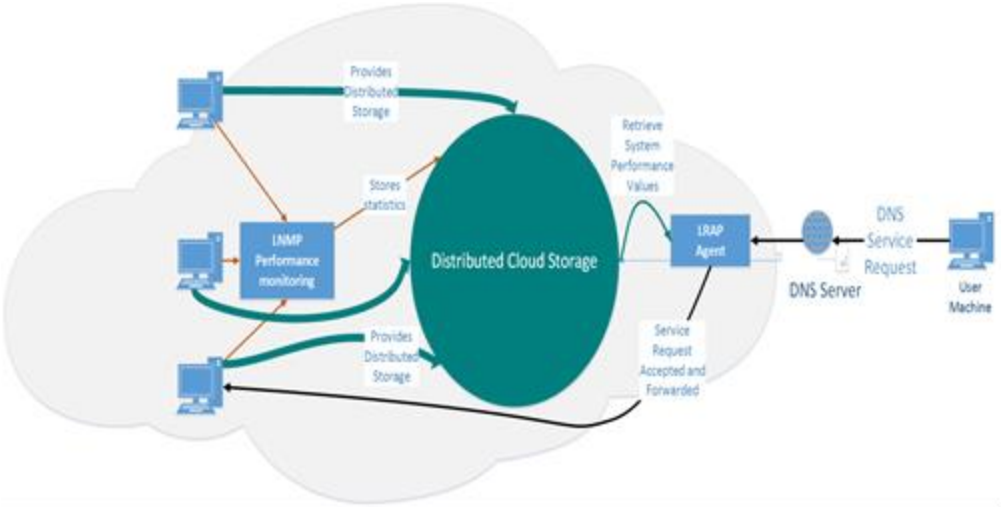


Figure 1: Community Cloud Architecture

2. Community Cloud Management Model

Figure 1 depicts a community cloud architecture which reveals how its two main components are interconnected through distributed cloud storage. The Lightweight Resource Allocation Protocol (LRAP) works by using the system performance to find where services should be executed in a networked environment. The emphasis is that it should, in its implementation, be suitable for low powered environments as well conventional. The performance value can be collected using LNMP or any other system monitoring service that is applicable to the prospect network environment. Allocation of services would come as a direct result of services being queried in the network. Incoming service requests would access a networked or local global API which would either return the location of the host where the request could be executed or forward the request to the particular node/service that the user application has requested in case that the requests made are actually successful.

2.1 Lightweight Resource Allocation Protocol

The idea behind the knapsack problem is assign a number of items of finite weight and value to a knapsack of finite size in order to achieve the most optimized ratio of value and weight in the bag/knapsack. The allocation process behind the LRAP protocol takes inspiration from the greedy approximation of the knapsack algorithm. In our case, each host on the network has a finite amount of resources available and each node on the networked environment advertises to the server its maximum penalty value. This is a predefined value that determines the maximum load allowed on the node in question, i.e. the maximum size of the bag. This is typically set to a value which expresses the performance threshold for the current machine. The “bag” is also not empty to begin with. These nodes are running their operating systems with whatever processes they need to be part of the network. Nodes in an LRAP network connect to a central point called a grid coordinator which is responsible for the allocation of tasks to nodes. It is at this node where the LRAP agent resides. The grid coordinator usually performs additional tasks in addition to being responsible for the allocation of services. This however is not part of the scope of this

paper. Each service running on the network typically consumes a certain amount of resources, such as CPU and memory. For example, loading a web page via Apache2 would place certain constraints on the disk, memory and CPU.

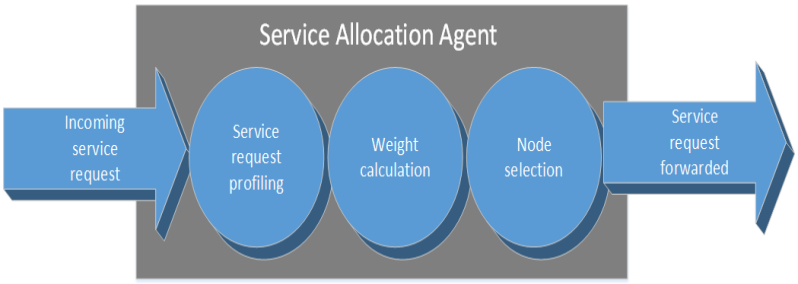


Figure 2: Service request flow

These can all be measured through LNMP and with it an average usage can be calculated to determine a resource consumption profile for the application in use. This profile can be translated into a weight that the process would be associated with. This value though should typically consist of multiple variables as each process could have different sort of impacts on the system. For example, some process might consume more RAM whereas another would be more CPU intensive and some might even consume a lot of everything available on system. Working with this sort of multidimensionality was too complex for this work. The optimization and classification of the weight values was also beyond the scope of this paper. Instead, a broader single valued weight was decided upon for each process in the system. In the examples used in this paper, we simply applied integer values (which had no upper bound); each value being an approximation of the service impact on the system when it runs per user request.

2.2. LRAP Implementation

The resource allocation model described above results in each node of the cloud infrastructure offering its own bag or 'execution bin' where services can run. Compared to the greedy approximation of the knapsack algorithm, we are trying to fully optimize one bin and then move on to the next, filling all of the bins by selecting the most available bin (read 'most empty') at the time of service request. That is, if the bin is provisioned to run the service of course. If not the next most available bin is selected. This procedure is outlined in the following steps.

- All the hosts on the network are connected to the grid coordinator. The coordinator ensures that it has the latest performance values for all of the connected hosts. Each host on the network is identified by a unique universal identifier (UUID).
- The node connects with the list of all its available services to the grid coordinator and shares this list. The services are then added to the connection pool and the list of hosts associated with particular services is updated. The nodes that connect have their penalty values adjusted using the following formula.

  penalty = (RAM + SWAP + CPU) / 3 (1)

  Where RAM is the ratio of used RAM, SWAP is the ratio of used SWAP space, and CPU is the ratio of occupied CPU time.
- The grid coordinator then arranges these hosts in ascending order by how much absolute availability they have. This means that if a host is connected and it has a much higher

maximum penalty value than another node on the network, the chances are much more likely that the node with the higher maximum penalty will be selected to be on top of the list.

- When a service is to be selected, the following checks are made
  - **New_penalty = penalty + service_weight (2)**
  - If New_ penalty is more than the maximum penalty defined, then the service request is rejected.
  - Otherwise, the service requests an IP address from the network manager and returns it on success.
- Thereafter the currently penalty is readjusted by adding the weight until the next evaluation cycle (every 60 seconds) completes and the new penalties are adjusted based on any new performance values retrieved from the client nodes.

  penalty = penalty + service_weight (3)

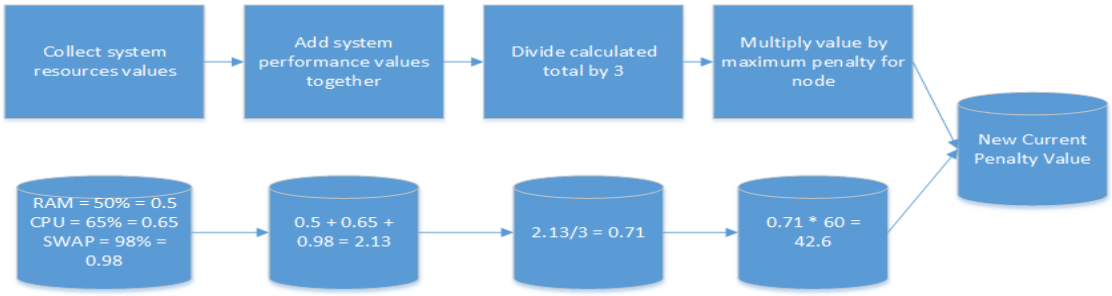


Figure 3: Current penalty calculation flow

Hostname	System Type	CPU	RAM	IP Address
alix1.local	Alix system board	500MHz AMD Geode LX800	256 MB	192.168.1.7
alix2.local	Alix system board	500MHz AMD Geode LX801	256 MB	192.168.1.6
alix3.local	Alix system board	500MHz AMD Geode LX802	256 MB	192.168.1.8
alix4.local	Alix system board	500MHz AMD Geode LX803	256 MB	192.168.1.9
beige.local	PC	2.2GHz Intel Pentium Dual	2048 MB	192.168.1.3
cheddarcheese.local	PC	1.8GHz Intel Core 2	1024 MB	192.168.1.2

Table 1: Hosts used in DNS performance tests

3. Experimental Evaluation

In order to display the efficacy of the solution in its ability to disseminate services in a networked environment, a DNS server was considered the perfect application to test the system. The DNS server was built using ‘evldns’; a self-described lightweight library to be used in constructing light, fast DNS servers. The package uses ‘libevent’ for high speed event handling and ‘ldns’ for the DNS packet manipulation. We configured each service to respond to queries made to the DNS server by their own custom subdomain names. The DNS server would be tasked to interpret these subdomain names and then interpret them as keys for selected services on the network. The networked system being tested in this particular paper is a heterogeneous network made up primarily of low powered network devices. The emphasis on the allocation protocol is to provision services in networks while providing as minimal impact as possible on

one particular node in the network. The devices used are Alix system boards and PC machines. A complete list of devices used in the network can be referenced in Table 1.

3.1. Stress Evaluation

These tests will be used to detect the state at which the system collapses. The results will be compared to that of the results achieved by the creator of 'evldns'. The stress test would consist of a network of two nodes. One being a high performance machine, whilst the other an Alix board configured with the LRAP agent running as the grid coordinator and a DNS server connected to it. The performance machine will then proceed to start slowly spamming the DNS server with queries which it has to resolve by first finding available nodes in the network. In this case the response is not the primary concern. The number of requests are slowly ramped by a value of 20% up every 20 seconds. This is con- ducted for a period of 15 minutes and 20 seconds. The system statistics will be recorded as before by using 'vmstat' to collect the system information every second. The solution from Frank Denis, 'DNS blast', was taken and modified to perform the DNS spamming in our specific network context.

3.2. Performance Evaluation

These tests will just be used to determine the overall operating values of the system within the network while the software solution is running and DNS requests are being made. The requests will be made at constant time intervals as opposed to the previous test. This is so we can better detect changes in the network and operating performance on the host nodes.

Name	Weight	Port	Service-URL
asterisk	3	5060	
stats	4	80	<host>/stats
storage	2	29999	
email	4	587	
maps	11	80	<host>/map
calendar	23	80	<host>/calendar
store	12	2342	
printing	6	8763	
webservice	4	8080	<host>/API/
webserver	8	80	<host>/

Table 2: Network service and weighting list

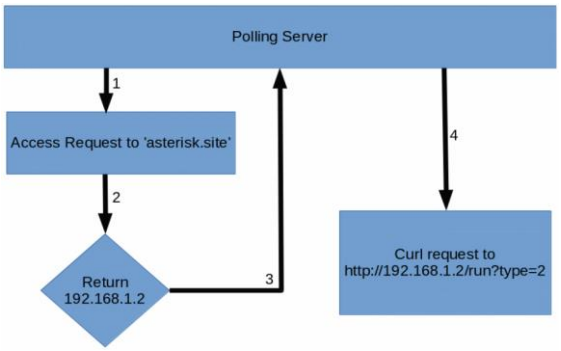


Figure 4: Hosts used in DNS performance tests

All performance values will be stored in the network using the existing BigCouch database infrastructure that was setup in the previous experiment. The idea here is to make requests in a

similar fashion to the stress tests. This however is not a test for the systems' limits but more to demonstrate the effects in a networked environment.
The experiment can be described as follows:

- The entire network is connected in its deployed environment with all the nodes are connected to each other wirelessly.
- The BigCouch database instances are running on all the connected nodes in the networks. So each instance will store its system data within the BigCouch cloud to allow for the allocation agent to monitor and calculate new performance values based on the load information stored over the previous minute.
- The DNS server is started on the dedicated DNS node (alix1.local). This is simply configured in the configuration file and the DNS server is started on boot. The correct permissions are required as port 53 (DNS listen port) requires privileged access.
- A polling server is setup with python installed to run the testing script. As before, the sub-process module is used again as the Popen call that it provides is non-blocking. This will allow us to easily send multiple packets to each hosts in the network in quick succession.
- The polling server is then selected to send payload packets of service requests to each node in the network. This will be done with all the nodes being connected in the network. The polling server will make a request to resolve the chosen hostname by randomly selecting from a list of available services.
- Load had to be simulated on the servers. To achieve this, Apache web service was installed on each machine in the network along with PHP5. The process of sending a payload was achieved with running an actual process on each machine. The idea here is to get the DNS server respond with the appropriate IP address when the polling server requests a service on the network. The polling server will then take the IP address returned by the DNS server and in turn query a PHP script via Apache2. Take a look at the example service request displayed in Figure 6.
- The polling server will then access the PHP script. The type value supplied in the URL is in reference to the service being accessed on the machine. This is done by creating a bash/python script which uses curl to access the resource at the specified IP. We are able to retrieve this value with relative ease by using the 'host' command. This even allows us to specify the DNS server that we wish to query, without having to change the default nameserver on the system specified in '/etc/resolv.conf'.
- The PHP script will then proceed to execute a loading process while consuming a portion of the CPU time and memory representative of the weight that the process is associated with. The aim is to capture the overall system load averages across all systems.
- The loading process executed is a benchmark stress testing tool called 'stress-ng'. This allows one to specify anything from IO, RAM, SWAP and CPU usage for a specified amount of time and at a certain level and simulate the various process weights and impacts on the systems.

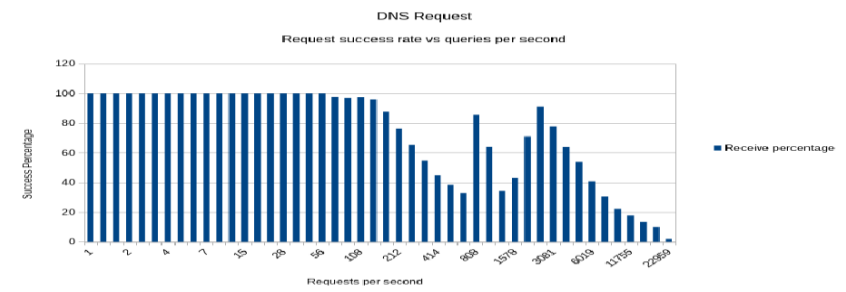


Figure 5: DNS request success rate vs queries per second

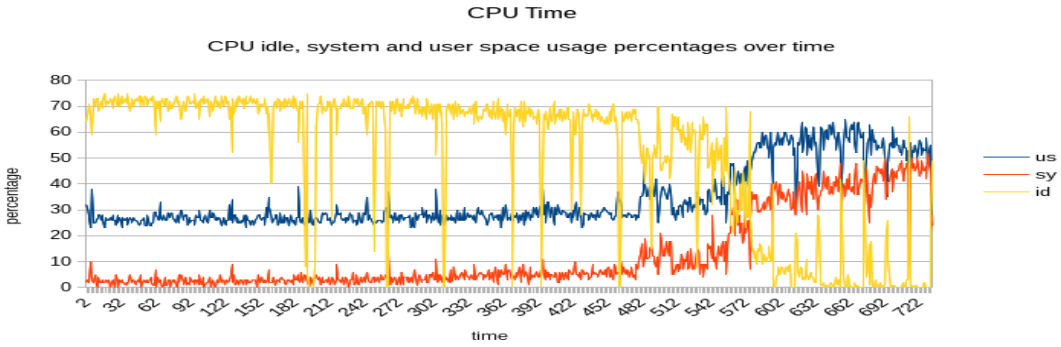


Figure 6: CPU usage vs DNS requests per second

4. Experimental Results

4.1 Stress Tests

It seemed (from observing the output from 'top' and 'vmstat' and the DNS server host), that a maximum threshold is reached where raising the amount of requests per second on an already loaded server seemed to make no difference. This was probably an issue relating to the network I/O of the Alix system board as they do not possess separate network and I/O controllers and everything is managed by the CPU. It should also be noted that these tests were not being conducted over a wireless connection. The results received when spamming the DNS server over a wireless connection were far off from even the first attempts before we started using DNS blast. This is probably due to greater latency when connecting via wireless. This is confirmed by the average result returned from the ping tests (17ms) to various nodes in the network as this network latency would be a predefined value in determining our maximum rate at which we could expect results to be sent and received. As this was more a test of the maximum load on DNS server and not a test involving the DNS server running in the wild, the fact that we tested this over a cabled connection is irrelevant. The system performs well initially but as it nears 482 seconds (136 169 requests per second), it starts to keel over and the reliability falls. There seems to be a trend downwards but the spikes only confirm the assertion that the system becomes unstable/unreliable as the number of requests increase. The system therefore seems to only be able to reliably respond to requests that are coming in every 5 - 7 milliseconds. Everything else after that seems to indicate a sharp drop in reliability of the system. These results are confirmed when looking at the CPU performance in Figure 6. We can see that the CPU idle (yellow) drops around the same time that the accuracy is dropping while the usr (blue) and sys (red) increases.

The figure shows an increased amount of time spent in user and kernel space the CPU was spending executing code. The increased usage just seems to snowball from here making the system unusable.

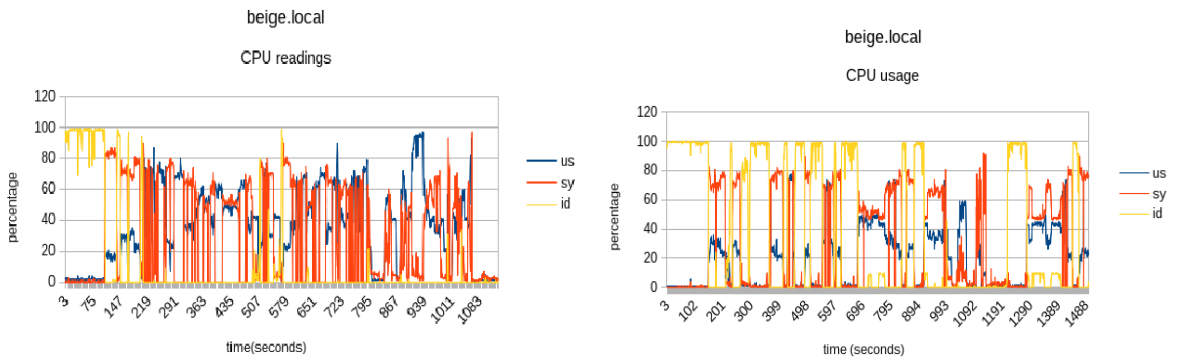
Number of requests	Host
10	192.168.1.7
71	192.168.1.2
103	192.168.1.3
3	192.168.1.6
18	192.168.1.8
16	192.168.1.9

Number of requests	Host
27	192.168.1.7
77	192.168.1.2
36	192.168.1.3
12	192.168.1.6
30	192.168.1.8
29	192.168.1.9

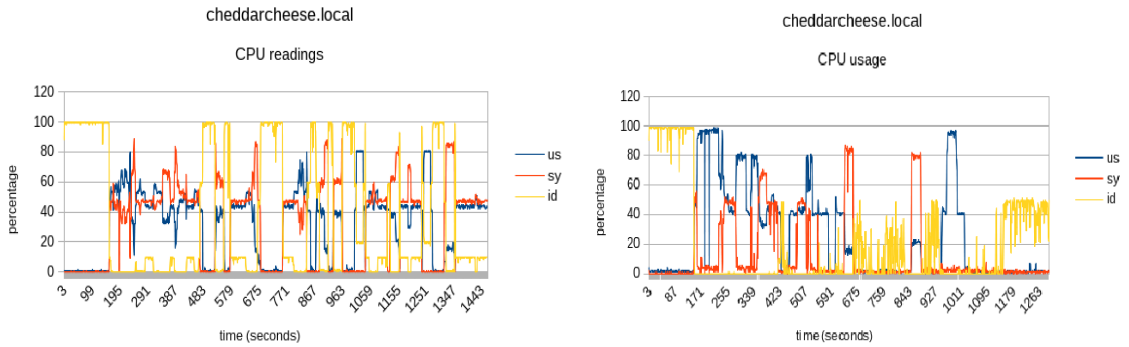
Table 3: Service requests per host

4.2 Performance Tests

The performance tests are performed twice. The first experiment is conducted with a maximum penalty that is way too large for one node to handle. A maximum penalty of 160 is given to the desktop machines with the rest of the nodes been given a very conservative 60. During the second experiment, this value is readjusted much lower (60). On the first run, we can see from Table 3 that the desktop machines are absorbing most of the service requests. This is because of the really high maximum penalty values which in turn make the hosts more 'available' for service requests. It should be noted that the higher the maximum penalty is defined, the harder it is to reach maximum since it is based on a ratio of the current performance values. When we compare the service requests in the network we can see that the load now appears to be much more distributed than before. The desktop machines still absorb considerably more tasks than the Alix system boards. It is expected that the desktop machines will be allocated more tasks since they have a much lower penalty than the Alix boards. Also, having a much lower maximum penalty value means that the device won't be allocated too many heavy tasks. This assumption is confirmed when we look at the CPU performance graphs for the desktop machines displayed in Figure 7. The second graph clearly shows that the system spends more time in idle than it does in the user and system states. This indicates that the CPU is doing less work for the duration of the experiment. This allows it to both complete tasks with relative ease and free itself up for other tasks. The Alix system boards in all likelihood were never even hit with a request to load one of the more weighty tasks (map, calendar, store).



(a) Desktop machine: beige.local



(b) Desktop machine: cheddarcheese.local

Figure 7: Two performance graphs showing the CPU performance on the first and second runs of the solution respectively

4.3 Cloud System Benchmarking

Ray Bellis' implementation of the 'evldns' [9] was able to achieve a query rate of 60 000 queries per second. However, Ray was testing on an HP DL385 server which is somewhat considerably faster than our Alix board system. Also, it is not entirely apparent how Ray tested this but we can consider that a similar technique to ours was applied. Our solution was only able to reach 136 queries (with 95% accuracy) per second before the great decline in reliability started kicking in. Considerably speaking though, the system performs reasonably well. Especially since we will most likely not be able to receive data even close to the max value (136-169 DNS queries per second) tested in a wireless networked environment. The various controllers on the device are simply far too slow to keep up with the amount of network traffic coming and the packets just get dropped or lost. The performance tests indicate the system's ability to distribute the load among various nodes in a wirelessly networked environment. Also having high 'penalty max' values should really be discouraged unless the machine has really high performance. This is because the current penalty calculation is done with a relative comparison of performance to the 'penalty max' value. However, the services, when being allocated to hosts in the network are adding a fixed not a scaled penalty value. Which is the correct intended behavior? But this means

that when the maximum penalty value is decided upon, it must be done in a way which most accurately describes the performance state of the machine, especially in relation to the processes it must run.

5. Conclusions and future work

The issue of service provisioning in low power environments was addressed in this paper by building upon the lightweight resource allocation protocol (LRAP) designed for lightweight community cloud infrastructures. The main contribution of this work was to show that with the correct resource brokering approach, services can be provisioned on the nodes of a lightweight cloud infrastructure in a way which does not cripple individual nodes of the underlying network, while ensuring that services are distributed to the nodes where the lowest impact on overall performance will be made. This strategy provides an effective way of providing services especially in low power environments. However, it shall be noted that in order to better allocate services to nodes, the multidimensionality of the service resource requirements and systems involved has to be brought to the table as this will only make the allocation algorithm more robust. Another dimension which has not been alluded or inferred upon so far in this work, is the particular network topology involved with the calculations. All the service operations involved in the network require the transmission of data via the network. This data transmission applies its own load on the various systems involved in the network and when topology is not considered, highly congested nodes could be selected by the LRAP protocol while the least burdened are left idle. In a performance sensitive context such as VoIP, this can become a very crucial determination step. The weights of processes and the maximum penalty attributed to nodes should also be something that is dynamically calculated based on historical data on the nodes. Addressing these issues is an avenue for future extensions of the work presented in this paper. The management of the communication network underlying the lightweight cloud infrastructure is another key parameter that may require redesigning existent network management techniques for the efficient engineering of the infrastructure. Multipath routing techniques such as presented in [11,12] will be redesigned to support QoS by having different forms of healthcare data propagated over different paths form a source to a destination. The cost-based traffic engineering techniques proposed in [13,14] will also be redesigned to balance traffic over the communication platform to increase throughput and reduce communication delays. Deploying a long distance sensor network [15] using flexible gateways as defined in [16] can expand the deployment in the rural settings of the developing world. This is another key issue that needs to be addressed as future research work.

References

[1] Gerard Briscoe and Alexandros Marinos, “Community Cloud Computing”, First International Conference on Cloud Computing, 1-4 December, Beijing, China, 2009.

[2] Antoine Bagula, Marco Zennaro, Gordon Inggs, Simon Scott and David Gascon, “Ubiqui tous Sensor Networking for Development (USN4D): An Application to Pollution Monitoring”, Sensors ISSN 1424-8220, Vol 12, Pages 391-414; doi:10.3390/s12010039, 2012.

[3] Million Mafuta, Marco Zennaro, Antoine B. Bagula, Graham W. Ault, Harry Sam Harrison Gombachika and Timothy Chadza, “ Successful Deployment of a Wireless Sensor Network for Precision Agriculture in Malawi ”, in International Journal of Distributed Sensor Networks 04/2013 2013:1-13, 2013.

[4] M. Zennaro, A. Floros, G. Dogan, T. Sun, Z. Cao, C. Huang, M. Bahader, H. Ntareme and A.B. Bagula. “On the design of a Water Quality Wireless Sensor Network (WQWSN): an Application to Water Quality Monitoring in Malawi”, in the proceedings of the IEEE NGWMN conference, Sept. 2009.

[5] Antoine B. Bagula, Nzioka J. Muthama, “The role of ICTs in downscaling and up-scaling integrated weather forecasts for farmers in sub-Saharan Africa”, In proceedings of ICTD'12, Pages 122-129, Atlanta, GA, USA, March 12–15 2012.

[6] M. Masinde and A. Bagula, “Framework for Predicting Droughts in Developing Countries Using Sensor Networks and Mobile Phones”, in Proceedings of SAICSIT 2010, October 2010.

[7] T Mullins and B.A. Bagula, “Monitoring Community Clouds: The Lightweight Network Management Protocol”, in the IEEE 2013 International Workshop on Intelligent Techniques for Ubiquitous Systems (ITUS-2013), Italy, December 2013.

[8] J. Shiou Leu et. Al, “Design and Implementation of a Low Cost DNS-based Load Balancing Solution for the SIP-based VoIP Service”, in proceedings of the IEEE Asia-Pacific Services Computing Conference, 2008.

[9] R Bellis, “evldns, An event-driven DNS server framework”, https://github.com/raybellis/evldns, last accessed 01/04/20116.

[10] Ritika Mahajan and Sarvesh Chopra , “Analysis of 0/1 Knapsack Problem using Deterministic And Probabilistic Techniques”, in proceedings of the second International Conference on Advanced Computing & Communication Technologies, 2012.

[11] A. B. Bagula. “Modelling and implementation of QoS in wireless sensor networks: a multi-constrained traffic engineering model”, in EURASIP Journal on Wireless Communications and Networking, 1, 2010.

[12] A.B. Bagula and AE Krzesinski, "Traffic engineering label switched paths in IP networks using a pre-planned flow optimization model", in Proceedings of the 9th MASCOTS symposium, Cincinnati, USA, ISBN 0-7695-1315-8, Pages 70—77, August 2001.

[13] A.B. Bagula, “Hybrid routing in next generation Ip networks”, in Computer Communications 29, number 7, pages 879-892, 2006.

[14] A. B. Bagula, “On achieving bandwidth-aware LSP/LambdaSP multiplexing/separation in multi-layer networks”, in IEEE Journal on Selected Areas in Communications, 25, (5): pages 987- 1000, 2007.

[15] Marco Zennaro, Antoine Bagula, David Gascon, and Alberto Bielsa Noveleta ,“Planning and Deploying Long Distance Wireless Sensor Networks: The Integration of Simulation and Experimentation“, in Lecture Notes in Computer Science (LNCS) Volume 6288/2010, Pages 191–204, 2010.

[16] 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.