Towards Distributed Architecture for Collaborative Cloud Services in Community Networks

Amin M. Khan, Mennan Selimi, Felix Freitag

Department of Computer Architecture
Universitat Politècnica de Catalunya
Barcelona, Spain
Email: {mkhan, mselimi, felix}@ac.upc.edu

***Abstract*—Internet and communication technologies have lowered the costs for communities to collaborate, leading to new services like user-generated content and social computing, and through collaboration, collectively built infrastructures like community networks have also emerged. Community networks get formed when individuals and local organisations from a geographic area team up to create and run a community-owned IP network to satisfy the community's demand for ICT, such as facilitating Internet access and providing services of local interest. The consolidation of today's cloud technologies offers now the possibility of collectively built community clouds, building upon user-generated content and user-provided networks towards an ecosystem of cloud services. To address the limitation and enhance utility of community networks, we propose a collaborative distributed architecture for building a community cloud system that employs resources contributed by the members of the community network for provisioning infrastructure and software services. Such architecture needs to be tailored to the specific social, economic and technical characteristics of the community networks for community clouds to be successful and sustainable. By real deployments of clouds in community networks and evaluation of application performance, we show that community clouds are feasible. Our result may encourage collaborative innovative cloud-based services made possible with the resources of a community.**

***Index Terms*—cloud computing; community cloud; community networks; collaborative resource sharing**

I. Introduction

The recent developments in information and communication technologies have significantly reduced the barriers for communication, coordination and collaboration for individuals and communities. This not only gave rise to widely adopted applications like social networking and user-generated content among many others, but infrastructures based on a cooperative model have also been built, for example community wireless mesh networks [1], which gained momentum in early 2000s in response to limited options for network connectivity in rural and urban communities. Using off-the-shelf network equipment and open unlicensed wireless spectrum, volunteers teamed up to invest, create and run wireless networks in their local communities as an open telecommunication infrastructure based on self-service and self-management by the users. These community networks have proved quite successful, for example Guifi.net[1] provides wireless and optical fibre based broadband access to more than 20,000 users. Current community networks use mainly wireless technology to interconnect nodes. With the commoditization of optical fibre, some community networks however have also started providing broadband services combining both technologies.

Community networks are a successful case of resource sharing among a collective, where resources shared are not only the networking hardware but also the time, effort and knowledge contributed by its members that are required for maintaining the network. Resource sharing in community networks from the equipment perspective refers in practice to the sharing of the nodes' bandwidth. This sharing enables the traffic from other nodes to be routed over the nodes of different node owners, allowing community networks to successfully operate as IP networks. Despite achieving sharing of bandwidth, community networks have not been able to extend this sharing to other computing resources like storage, which is now common practice in today's Internet through cloud computing. There are not many applications and services used by members of community networks that take advantage of resources available within community networks. When members of community network can share and trade resources based on a collaborative cloud computing model, they can provide their excess capacity to others as the demand fluctuates and in return can take advantage of services and applications that were not possible earlier due to the limited resources.

The concept of community clouds has been introduced in its generic form before, e.g. [2], [3], as a cloud deployment model in which a cloud infrastructure is built and provisioned for an exclusive use by a specific community of consumers with shared concerns and interests, owned and managed by the community or by a third party or a combination of both. We refer here to a specific kind of a community cloud in which sharing of computing resources is from within community networks, using the application models of cloud computing in general.

We centre the contribution of this paper on a collaborative distributed architecture for community clouds, which integrates into the cloud not only the computation and storage hardware contributed to the community network by its members, but also the socio-economic contribution they make to the collective effort in the form of knowledge, time and help. Such an archi-

[1] http://guifi.net

tecture tailored to the specific situation and social and economic context of the community networks allows the collaborative cloud services to better fit the demands of local communities, facilitating adoption and uptake of community cloud model. In our earlier work, we have explored how incentive-based resource regulation [4]–[6] and economic policies [7] can affect collaboration among the members of community networks, and how the scalability issues can affect the design of a community cloud system [8]. We are also building a prototype system to be deployed in Guifi.net community network [9], [10], and investigating the performance of cloud services in these real-world settings [11]–[13].

The rest of the paper is organised as follows. Section II presents related work, and section III discusses the requirements for a community cloud. Section IV presents the distributed architecture with support services taking into account socio-economic context of the community networks necessary for encouraging collaborative resource sharing. Section V details the prototype deployment in the community network testbed and presents results from experiments into collaborative cloud services. Section VI concludes and discusses future research directions.

II. Related Work

The idea of collaboratively built community clouds follows on from earlier distributed voluntary computing platforms, like BOINC [14], Folding@home [15], PlanetLab [16] and Seattle [17], which mainly rely on altruistic contribution of resources from the users, though various mechanisms have been studied in the context of peer-to-peer systems [18] that address different problems of collaborative resource sharing. There are only a few research proposals for community cloud computing, for example Cloud@Home [19] project aims to harvest in resources from the community for meeting the peaks in demand, working with public, private and hybrid clouds to form cloud federations. Social cloud computing [20] takes advantage of the trust relationships between members of social networks to motivate contribution towards a cloud storage service, and such social clouds have also been deployed in CometCloud framework by federating resources from multiple cloud providers [21]. Gall et al. [22] have explored how an InterCloud architecture [23] can be adapted to community clouds, and federated cloud architectures [24] in general are being actively explored for combining services from multiple cloud providers.

From the review of related work, we find that none of the above cases have a prototype for the concrete situation of the community networks. In the cloud system that we present, we aim to take into account several of the important social and technical factors that characterise community networks, and therefore the cloud architecture we propose is tailored to the specific context of the community networks.

III. Requirements

A community cloud is a combination of a number of cloud systems being run and managed independently by the different community members. The community cloud bridges in different aspects the gap between the public cloud, the general purpose cloud available to everyone, and the private cloud, available to only a limited set of users with user-specific services. These requirements provide the foundation for the design of the community cloud system, and need to be satisfied for it to be deployed and adopted successfully by the community.

A. Autonomy

Community cloud systems may be formed based on individual cloud systems that are set up and managed independently by different owners. The main requirement for a cloud owner for participating in such a community cloud is that the local cloud setup should adhere to the common API provided by the community cloud, and contribute resources to the community.

B. Security

There are many security challenges that need to be addressed for ensuring users' trust in the system, and with multiple independent cloud providers from the community, security becomes even more important in a community cloud.

C. Self-Management

Community cloud should self-manage itself and continue providing services without disruption when nodes go offline. Self-management should also help in the coordination between different cloud owners that become part of a federated community cloud.

D. Utility

For the acceptance of the community cloud, it should provide applications that are valuable for the community, since usage strengthens the value of the community cloud, motivating its maintenance and update. These applications need to differentiate from the generic cloud services available over the Internet. For example, FreedomBox[2] and MeshNet[3] projects focus on ensuring privacy, and FI-WARE CoudEdge[4] and ownCloud[5] let cloud applications consume local resources.

E. Ease of Use

Most of the users of the community cloud will not be proficient in cloud technologies, so setting up nodes for deployment and managing cloud software should be simple and straightforward. The easier it is for users to join, participate and manage their resources in the community cloud, the more the community cloud model will be adopted. To this end, in terms of an institutional policy, we have developed a Linux-based distribution for deployment in the Guifi.net community cloud [9]. It will make the process of joining and consuming cloud services almost automated with little user intervention.

[2]http://freedomboxfoundation.org
[3]https://projectmeshnet.org/
[4]http://catalogue.fi-ware.eu/enablers/cloud-edge
[5]http://owncloud.org

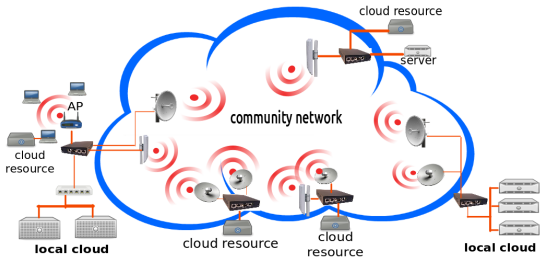

Figure 1. Nodes in a community network with cloud resources

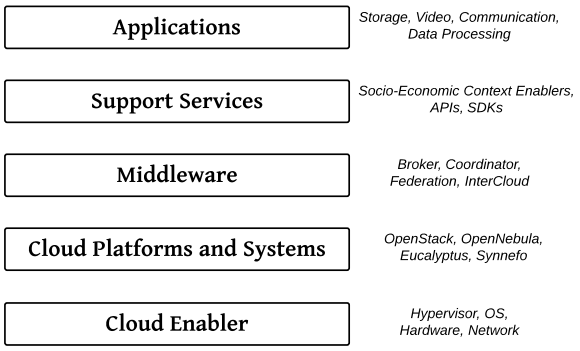

Figure 2. Different layers of the community cloud management system

F. Incentives for Contribution

Community cloud builds upon collective efforts of the members of the community networks, and requires the contribution of the volunteers in terms of their time, knowledge and effort as well as computing, storage and network resources. For the community clouds to be sustainable, incentive mechanisms are needed to encourage users to actively contribute towards the system.

G. Support for Heterogeneity

The hardware and software used by members in a community cloud can have quite varying characteristics, and the cloud system should handle this seamlessly.

H. Standard API

The cloud system should make it straightforward for the application programmers to design their applications in a transparent manner for the underlying heterogeneous cloud infrastructure. The API should provide the appearance of a middleware that obviates the need to customize the applications specific to each cloud architecture [25]. This is essential for community clouds when these result from the federation of many independently managed clouds. Providing a standard API for the community cloud ensures that applications written once for a particular community cloud system can be easily deployed on new cloud architectures.

I. QoS and SLA Guarantees

The community cloud system needs mechanisms for ensuring quality of service (QoS) and enforcing service level agreements (SLA).

IV. Collaborative Distributed Architecture for Community Cloud

A community network is managed and owned by the community, where nodes are managed independently by their owners. The computer machines or nodes in a community network vary widely in their capacity, function and capability, as illustrated in Figure 1. Some hardware is used as super nodes (SNs) that have multiple wireless links and connect with other SNs to form the backbone of the community network, and are usually intended to be stable with permanent connectivity. Others act just as ordinary nodes (ON) and are only connected to the access point of a SN. Topological analysis of the Guifi.net community network [26] indicates that from approximately 17,000 analysed nodes of Guifi.net, 7% are SNs while the others are ONs.

From the node types shown in Figure 1, it can be seen that principally the hardware for computation and storage is already available in community networks, consisting of some servers attached to the networking nodes. No cloud services, however, are yet deployed in community networks to use this hardware as a cloud, leaving the community network services significantly behind the current standard of the Internet. Our vision is that some community wireless routers will have cloud resources attached, building the infrastructure for a community cloud formed by several cloud resources attached to the nodes. We note that ONs could principally also contribute cloud resources.

An architecture for the community cloud system that manages such infrastructure needs to be robust, self-managing and efficient at handling the heterogeneity among the nodes. The option for enabling a community cloud on which we focus here is to deploy a cloud management platform tailored to community networks on the nodes attached to the network. There are a few cloud management systems available to manage public and private clouds, notably OpenStack[6], OpenNebula[7], CloudStack[8], Eucalyptus[9] and Synnefo[10] among others. Such cloud management systems can be tailored for community networks by extending the existing functionality to address the particular conditions of community networks. For example, incentive mechanisms inspired by the social nature of community networks can be built into resource regulation component to encourage users to contribute resources [4]–[6].

The conceptual overview for the cloud management system that we propose for community networks consists of multiple layers, as shown in Figure 2, with different components at each layer, as highlighted in Figure 3. The nodes along with

[6]http://www.openstack.org
[7]http://www.opennebula.org
[8]http://cloudstack.apache.org
[9]http://www.eucalyptus.com
[10]http://www.synnefo.org

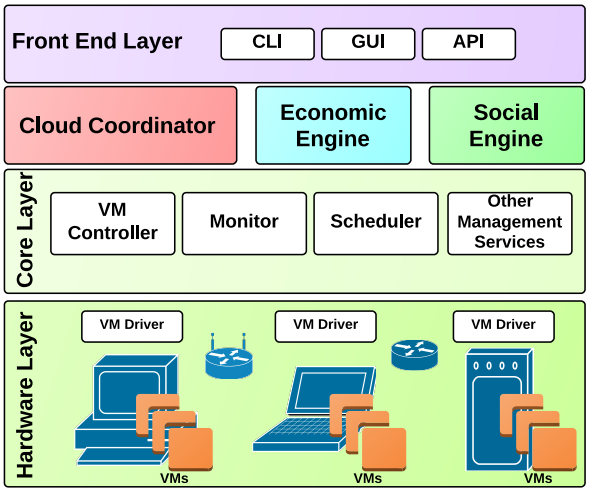

Figure 3. Architecture of the community cloud management system

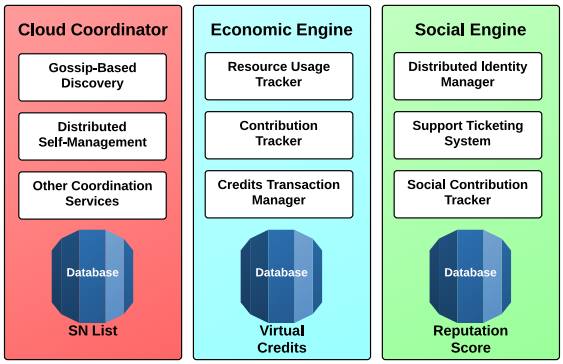

Figure 4. Distributed components of the community cloud management system

the communication infrastructure of the community network form the the hardware layer of the cloud architecture. The core layer residing in the SN contains the software for managing and monitoring the virtual machines (VMs) on ONs. The front end layer provides the interface of the infrastructure service (Infrastructure-as-a-Service, IaaS). The components cloud coordinator, economic engine and social engine provide additional services for customising cloud infrastructure to the community networks, see Figure 4.

The core of community cloud management system is the virtual machine manager (VMM) that is responsible for instantiating, scheduling and monitoring virtual machines on the nodes. The virtual machine manager consists of the following layers, which are common to most cloud computing architectures.

1) Hardware Layer: This consists of the physical infrastructure that is needed to run a cloud system. The hardware in the community networks mostly consists of ONs and SNs and the wireless links provided by the mesh network, along with any attached computation, storage and other resources.

2) Core Layer: The core layer consists of components that are responsible for creation, allocation, scheduling, monitoring and management of VMs on the nodes. This can include has following main components, some of which are shown in Figure 3.

- Virtual Machines Controller
- Virtual Machines Scheduler
- Virtual Machines Monitor
- Hosts Manager
- Virtual Network Manager
- Virtual Machines Image Data Store

The functionality of the core layer is already provided by tools like OpenStack and others. Community cloud manager can, therefore, make use of these existing tools and extend their functionality to suit the needs of the community network.

3) Cloud Coordinator: The cloud coordinator is responsible for the federation of the cloud resources which are independently managed by different SNs. It provides the interface for other components like economic engine and social engine to request information from other SNs. The cloud coordinator components in different SNs connect among themselves in a decentralised manner to exchange relevant information about managing the available resources. By default applications running at a local cloud can only consume resources from the ONs directly managed by the local SN. With the cloud coordinator, the infrastructure service can provide a unified view of the resources contributed by multiple local clouds. When federating multiple local clouds, the cloud coordinator applies a peering regulation mechanism [4], [5] fed by the economic engine and social engine to perform resource allocation. The cloud coordinator can consist of multiple components, some of which are indicated in Figure 4.

- Gossip-Based Discovery: The design of a community cloud manager follows a decentralised approach, so the cloud coordinator relies on gossip-based discovery mechanisms to manage overlay network of the SNs in community cloud.
- Distributed Self-Management: The community cloud has to efficiently manage the distributed resources in an autonomous way as ONs and SNs join and leave the network. Distributed self-management forms a core component of the cloud coordinator to ensure successful operation of the community cloud.

4) Economic Engine: The role of economic engine is to manage the accounting and auditing for the infrastructure service so that the access can be regulated to the users of the community cloud. In contrast to public clouds, the main incentive for the providers in community clouds is the utility that they will get from the system by consuming its services and applications. The economic engine manages a system of virtual credits that encourages the users to contribute resources to the cloud. This component will consist of many modules, some of which are highlighted in Figure 4.

- Resource Usage Tracker: This module connects with the VM Monitor in the core layer to get details about the resource usage. It links this information to the user who requested the VMs and keeps record of it for accounting and auditing purposes. This information forms the basis

for regulating access to the resources.

- Contribution Tracker: The information from the VM Monitor is also fed to the contribution tracker module which uses it to register the resources contributed by the owner of the nodes. This information is used to reward the provider of the resources with virtual credits.
- Credits Transaction Manager: This module manages the virtual credits database and credits or debits the account of different users with virtual credits. The database gets updated whenever someone requests resources or contributes them to the community cloud. The challenge for this module is to handle these transactions in a secure manner within a distributed system.

5) Social Engine: The community cloud is as much a social construct as it is a technical construct. The existence of the community cloud is not possible if there is a lack of participation from the community. Running a community cloud not only requires supply of technical resources like storage and network bandwidth, but also the time and effort of the users who setup and manage the network equipment. Whereas the economic engine takes care of the incentives in the virtual world, the social engine is the component that encourages contribution in the physical world. We discuss here some of the modules that help to achieve this goal. These modules may not be integral to the cloud management platform from a technical point of view, but nevertheless provide functionality necessary for the smooth running of the community cloud.

- Distributed Identity Manager: This module manages the global identity of the users in the system in a decentralized manner. This unique system-wide user ID is needed to track the usage and contribution by each user.
- Support Ticketing System: This module provides a system for the users to help each other in resolving the problems encountered while using the community cloud. The volunteers who provide the support to others are encouraged by rewarding them with better reputation in the system. This reputation can then translate in to an increase in virtual credits which the user can spend for consuming services in the community cloud.
- Social Contribution Tracker: This module provides incentives to the volunteers who help with the smooth running of the community cloud. The volunteers contribute with their time and effort to setup and maintain the hardware and network. This module tracks this contribution of the volunteers in the reputation score database. The social contribution tracker interacts with the credits transaction manager module in the economic engine and the users can exchange the reputation score with virtual credits. The virtual credits allow the volunteers to consume the applications and services provided by the community cloud.

6) Frontend Layer: The frontend layer provides the interface to interact with the infrastructure service of the community cloud. This includes modules like command line interface (CLI), graphical user interface (GUI), application programming interface (API), and any other tools that assist with developing cloud application using the infrastructure service.

A. Interaction between Different Components

We discuss here some usage scenarios and explain how different components of the community cloud management system can interact with each other.

1) Workflow for a resource request: Consider the case when a user requests a new VM from the community cloud. The user connects to the GUI in the frontend layer and submits a request for a VM instance. The request is forwarded to the cloud coordinator that checks for availability at ONs, and if not available locally, forwards the request to neighbouring SNs. The cloud coordinator then checks with the identity manager component of the social engine, which authenticates the user to confirm whether the user has access to the resources. Cloud coordinator then checks the virtual credits database of the economic engine to see if the user has sufficient credits available to fulfil the request. After confirming that the user can consume resources, the request is forwarded to the scheduler in the core layer which selects the ON where the VM will run. The monitor in the core layer provides the details of the consumed resources to the resource usage tracker component in the economic engine. The credits transaction manager in the economic engine updates the credits of the requester and provider of the VM in virtual credits database. The contribution tracker in the economic engine updates the details for the provider. The user who requested the VM can check the GUI in the frontend layer for the status of the VM.

2) Workflow for social contribution: Consider the case when a user contributes to the community cloud by providing services like setting up routers or performing network maintenance. The user connects to a GUI application in the frontend layer and submits the details of her contribution. The request is forwarded to the cloud coordinator which contacts the identity manager and social contribution tracker components to confirm whether the user is registered, and also forwards the user's details to neighbouring SNs. The social contribution tracker updates the values for the user in the reputation score database. It also contacts the credits transaction manager component in the economic engine which updates the virtual credits database for the user. The social contribution tracker and the credits transaction manager provide the details to the frontend layer and the GUI informs the user of the outcome of the operation.

3) Workflow for support provision: Consider the case when a user contributes to the community cloud by providing support to others in resolving issues with the system. The support ticketing system in the social engine provides a mechanism for users to request and provide support on self-help basis. The main role of the support ticketing system in the community cloud is to provide incentives to the volunteers by rewarding them with virtual credits for their effort. When a user helps others with fixing their problems, the support ticketing system keeps track of the feedback. It checks with the identity manager component of the social engine to authenticate the user. The social contribution tracker then updates the reputation score

database for the user, and also asks the credits transaction manager to update the virtual credits database for the user. The details are provided to the frontend layer where the user can see her virtual credits via the GUI.

B. Socio-Economic Mechanisms for Supporting Collaboration

The purpose of economic mechanisms and social and psychological incentives is to let the community cloud transition from inception through early adoption to finally ubiquitous usage [7]. In the nascent stage, the community cloud may not be able to provide a lot of value until a critical mass of users are using the system. After that threshold, still the relative cost to achieve a little utility will be significant, which means that the early adopters of the system remain highly motivated and committed to the success of community cloud and continue to contribute resources even though they receive little value from the system in return. But once a significant proportion of community network members have joined the community cloud, the relative cost to obtain value from the system tumbles and in the longer run the system is able to sustain itself with contributions that may be small in size but are made by a large number of users.

The mechanisms must take into account the costs and benefits involved in participating in community cloud. For instance, the initial costs for setting up nodes in the community cloud involves hardware and installation costs. The continuous operation of the cloud node requires additional costs including network costs given by donating network bandwidth and any other subscription fees, energy costs to pay for electricity bills to run the computer equipment as well as cooling apparatus, maintenance cost to fund any technical support and replacements of parts, and hosting costs to provide storage space for the equipment. Besides these costs at the individual level, there are also the transaction costs and management overheads necessary for the collective operation of community cloud.

The individuals in community cloud act as private enterprises where they offer services to generate revenue. The revenue for the community cloud users include tangible benefits like the services and applications that they will be able to consume, and intangible benefits like the sense of belonging to the community and personal satisfaction because of their contributions. The services can range from infrastructure to platform to software services meeting a spectrum of different needs of the users.

Different policies addressing relevant issues of the technical, social, economic and legal aspects of the community cloud are designed to encourage collaboration, for example commons license and peering agreements can be implemented that extend the idea of reciprocal sharing from Wireless Commons License[11] and Pico Peering Agreement[12] in community networks. The social context of community networks provides opportunity to harness social capital and the different roles of social relationships. Similarly, lowering transaction costs and entry barriers, facilitating participation of developers, exploring different service models to provide value addition and differentiation,

[11] http://guifi.net/es/ProcomunXOLN
[12] http://www.picopeer.net

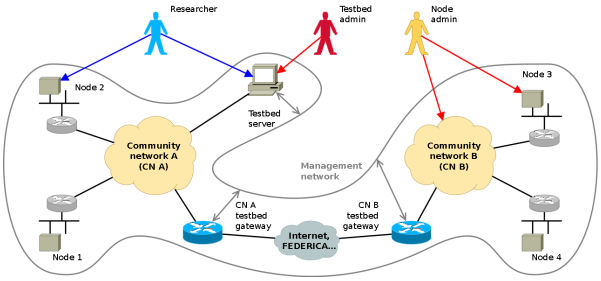

Figure 5. Experiment setup using Community-Lab testbed, with Guifi.net and AWMN connected through FEDERICA

and taking advantage of locality and overlay topology of the network can prove useful. Such mechanisms help adapt the ecosystem of community cloud infrastructure and services to the aspirations of community network members.

V. Community Cloud Deployment

We explain in this section our current work in setting up a prototype cloud infrastructure in Guifi.net and Athens Wireless Metropolitan Network (AWMN)[13] community networks, and present results from our experiments with distributed storage and data sharing service running in this testbed.

A. Experiment Environment: Community-Lab Testbed

For having a realistic community network setting for the collaborative cloud services, we have used Community-Lab[14] testbed for setting up our community cloud infrastructure. Community-Lab is a distributed infrastructure developed by the CONFINE project [1], where researchers can deploy experimental services on several nodes deployed within federated community networks. Community-Lab provides IaaS for community clouds by providing the researchers with a set of VMs, implemented as Linux containers (LXC), from the nodes which are distributed within the community network. Within these VMs we deploy Cloudy[15] [9], a Debian based distribution, which comes pre-installed with some of the collaborative distributed applications, like Tahoe-LAFS[16], ownCloud, etc.

The primary configuration for our application deployments consists of nodes from the two community networks, Guifi.net in Spain and AWMN in Greece, which are connected on the IP layer though Federated E-infrastructure Dedicated to European Researchers (FEDERICA)[17], enabling network federation, as illustrated in Figure 5. This implies that some part of the distributed applications are in fact spread over nodes in Guifi.net, while the other components are hosted on the nodes belonging to AMWN. The nodes of our experiments are the real nodes from both the community networks, and they are connected to other actively used nodes within the community network through wireless IEEE 802.11 a/b/n connections.

[13] http://www.awmn.net
[14] http://community-lab.net
[15] http://repo.clommunity-project.eu
[16] https://tahoe-lafs.org
[17] http://www.fp7-federica.eu

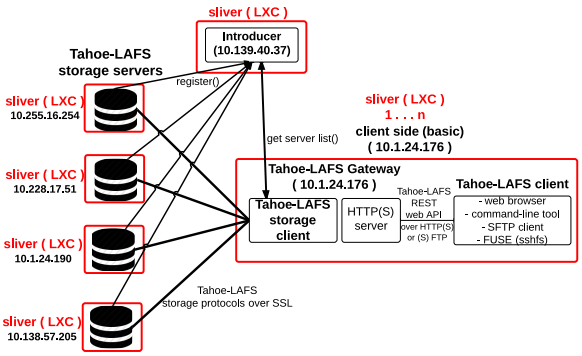

Figure 6. Tahoe-LAFS deployed in the Community-Lab testbed

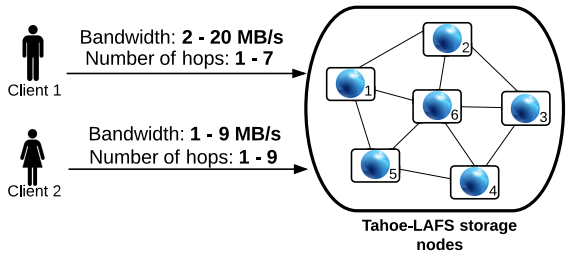

Figure 7. Two Tahoe-LAFS clients at different locations

B. Collaborative Cloud Services

We present here a brief overview of the different collaborative cloud services that we are exploring and planning to deploy in our community cloud testbed based on Community-Lab.

1) Infrastructure-as-a-Service (IaaS): We have set up different machines with OpenStack, Eucalyptus, Proxmox[18], Docker.io[19] and OpenWRT/LXC installations, which provide either a virtual machine or Linux container based environment. All the nodes share the same Guifi.net IP-address space and are network reachable. This means that they can support applications and services deployed on the federated infrastructure from multiple cloud setups.

2) Platform-as-a-Service (PaaS): We set up a storage service (based on Tahoe-LAFS and ownCloud) and a database service (based on CATS project's Caracal database[20]) at platform level to support the development of cloud applications for the end users. The Debian-based Cloudy distribution [9] also integrates support services, like Avahi[21] which provides services discovery and management.

3) Software-as-a-Service (SaaS): We are also looking into providing useful collaborative services for the end users because these application are critical for the uptake of community cloud model among the existing users of community networks. For instance, we are setting up collaborative distributed storage service using a combination of ownCloud, XtreemFS[22] and Tahoe-LAFS, and video streaming service using PeerStreamer [27] and PeerTV [28].

C. Experiment Setup

For our experiments, we have used nodes from Guifi.net located in our research lab, UPC campus[23], and elsewhere in Barcelona city, and from AWMN located in Athens. The hardware of most of these Community-Lab nodes consists of Jetway devices that are equipped with an Intel Atom N2600 CPU, 4GB of RAM and 120GB SSD. We also include a few nodes in our lab from a Proxmox cluster consisting of machines with 4x Intel Core i7-3770 3.40GHz CPU with 16 GB RAM and 1 TB hard disk, and this provides heterogeneity in terms of storage space and processing power.

In the first experiment, we deploy ownCloud and Tahoe-LAFS instances for a collaborative distributed cloud storage in Guifi.net. We measure the read and write throughput as observed from two different Tahoe-LAFS clients and evaluate the overall functionality of the application. In the second experiment, we deploy BitTorrent application in nodes from both Guifi.net and AWMN to demonstrate the option of collaborative data sharing within different community networks. In these experiments, Community-Lab nodes from AWMN in Athens and from Guifi.net in UPC Campus and Barcelona city provide 1 GB of storage space, wile the nodes in our lab share 5 GB of storage space.

1) ownCloud Setup: We deploy an instance of ownCloud on a relatively stable node in Community-Lab testbed, since the instance of ownCloud is not replicated. Using ownCloud server URL, authenticated users can remotely upload and download files through the ownCloud web interface. To store the files uploaded to ownCloud, we have replaced the ownCloud backend with Tahoe-LAFS, as explained next.

2) Tahoe-LAFS Setup: For Tahoe-LAFS deployment in the testbed [12], we use the Cloudy distribution [9] to set up the gateway and the introducer, storage and client nodes required for Tahoe-LAFS in the VM instances provided by the testbed. There are total 12 nodes from Guifi.net, four nodes each located in the lab, UPC campus and Barcelona. Figure 6 shows the resulting Tahoe-LAFS architecture used in our experiments in the Community-Lab testbed. While the Tahoe-LAFS introducer service runs on a separate node, the Tahoe-LAFS clients are on the same nodes where ownCloud is installed. This way the performance of Tahoe-LAFS translates directly to the performance observed at the ownCloud server itself.

Figure 7 shows the bandwidth and number of hops observed from the two different clients to the Tahoe-LAFS storage nodes. The first client is located in one of the nodes at our lab, while the second client is at a different location in Barcelona. The six storage nodes where the clients write to and read from are also shown. We see that the network characteristics observed by these two clients are not the same. The links from the first client to the storage nodes have better connectivity as compared to the links for the second client. We have used the default

[18] http://proxmox.com/
[19] http://docker.com
[20] http://cats.sics.se
[21] http://avahi.org
[22] http://xtreemfs.org
[23] http://dsg.ac.upc.edu/qmpsu

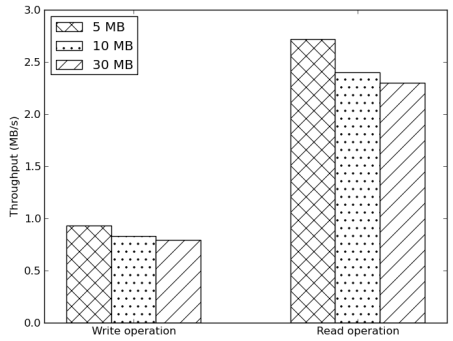

Figure 8. Read and write performance for Tahoe-LAFS client in the lab

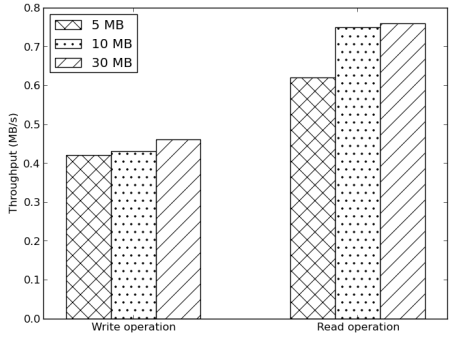

Figure 9. Read and write performance for Tahoe-LAFS client in Barcelona

Tahoe-LAFS erasure coding parameter of 3-of-10 which means that for every file uploaded, Tahoe-LAFS uses 10 of the 12 storage servers. These storage servers can differ significantly in bandwidth and latency.

D. Experiment Results

1) Tahoe-LAFS Evaluation: We evaluate the storage performance of Tahoe-LAFS in the community network in order to assess the impact of network latency, connectivity and bandwidth. We focus on the read and write performance and ignore other Tahoe-LAFS features such as data recovery, repair, maintainability, etc. We collect measurements from the two Tahoe-LAFS clients at different locations in the community network for reading and writing fixed-size files. In these tests, we ignore concurrent reads and write operations.

We use workloads consisting of files of different types, such as jpg, pdf, zip, mp4, etc. We use 15 consecutive read and write operations of files of size 5 MB, 10 MB and 30 MB. We present the average read and write throughput in MB/s. In every write operation, we use file with different type and content, since uploading the same file results in Tahoe-LAFS returning the same capability string for the file. The capability string in Tahoe-LAFS is derived from the content of the file and the convergence secret which is randomly generated by the node when it first starts up.

Figure 8 shows the results for the client located in the lab. Moderate write performance of Tahoe-LAFS can be attributed to the fact that Tahoe-LAFS performs expensive cryptographic operations and the default stripe size, which determines the granularity at which data is being encrypted and erasure coded, is optimized for writing small files. This results in 0.9 MB/s when writing a 5 MB file using the default 3-of-10 configuration. The topological placement of the nodes also contributes to this latency, since during write operation, Tahoe-LAFS tries to distribute the files as widely as possible, using a different pseudo-random permutation for each file, and Tahoe-LAFS does not take into account other node properties like its location in the network. The read performance of 2.7 MB/s for a 5 MB file is better then write operation, which is expected in erasure coded systems, since read operations transfer less data than write operations. Figure 9 shows the performance for the client located in Barcelona. The throughput for writing a 5 MB file is 0.42 MB/s, which is twice as slower as the write throughput for the client in the lab. This is because the performance is affected by the heterogeneous and dynamic network conditions of the community network.

2) BitTorrent Evaluation: The goal of this experiment is to show the applicability of BitTorrent for data sharing between different community networks. We evaluate the performance of BitTorrent for sharing small files between Guifi.net and AWMN. The nodes are the same as the ones used in the Tahoe-LAFS experiment, with 10 nodes from Guifi.net in Barcelona and 10 nodes from AWMN in Athens. We install Opentracker software[24] as BitTorrent tracker on a node located in Guifi.net. The BitTorrent Transmission client[25] is installed on the other nodes, and the seeder node, which serves the file, is located in AWMN. The initial seeder provides the complete file of 30 MB and the other nodes from both Guifi.net and AWMN download this file. The download performance depends on the location of the nodes and the mechanisms of the BitTorrent protocol itself. For nodes located in Guifi.net, the average download rate achieved is 5.6 Mbps resulting in download latency of 42 seconds for 30 MB file. For the nodes located in AWMN, download rate achieved is 9.2 Mbps resulting in download latency of 26 seconds for 30 MB file. All the file sharing operations we experimented with completed successfully.

VI. Conclusion and Outlook

Community networks would greatly benefit from the additional value provided by the applications and services deployed in the community clouds. Such clouds for community networks, however, have not been specified yet by the related work to enable further developments. We proposed a collaborative distributed service architecture for providing cloud services that is tailored to the unique nature and conditions of community networks. Our architecture proposes on top of existing cloud management platforms a set of support services for regulated

[24]http://erdgeist.org/arts/software/opentracker
[25]http://www.transmissionbt.com

resource sharing to encourage active participation of the community members which is required to form and maintain the cloud infrastructure, and for supporting the federation of cloud resources. Since community networks are volunteer organisations, we consider such support services an essential step for assuring a sustainable community cloud within community networks. We have deployed attractive applications on community cloud infrastructures in the Guifi.net community network to assess the applications' performance. We observed the feasibility of such applications in the community cloud and their correct functioning, which is crucial to attract real users for the next step of our research.

Based on the proposed architecture, our next step is to further develop the identified components and test them in the community cloud by engaging end users from community networks with these applications. The deployed prototype will allow running experiments in the real setting of a community network to investigate the performance of such a collaborative distributed community cloud. The resulting empirical studies will feed back to validate and improve the design of different components in the architecture. This proposal of the distributed community network architecture is a first step to exploit the potential of community clouds to complement existing public cloud services, opening the way to build collaborative user-shaped innovative applications for local communities.

ACKNOWLEDGEMENT

This work was supported by the European Framework Programme 7 FIRE Initiative projects CONFINE, FP7-288535, and CLOMMUNITY, FP7-317879, and by the Universitat Politècnica de Catalunya BarcelonaTech and the Spanish Government through the Delfin project, TIN2010-20140-C03-01.

REFERENCES

[1] B. Braem *et al.*, "A case for research with and on community networks," *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 3, pp. 68–73, Jul. 2013.

[2] P. Mell and T. Grance, "The NIST Definition of Cloud Computing," *NIST Special Publication*, vol. 800, no. 145, 2011.

[3] A. Marinos and G. Briscoe, "Community Cloud Computing," in *Cloud Computing, First International Conference, CloudCom 2009*, ser. Lecture Notes in Computer Science, M. Jaatun, G. Zhao, and C. Rong, Eds. Beijing, China: Springer Berlin Heidelberg, Dec. 2009, vol. 5931, pp. 472–484.

[4] A. M. Khan, U. C. Buyuksahin, and F. Freitag, "Prototyping Incentive-based Resource Assignment for Clouds in Community Networks," in *28th IEEE International Conference on Advanced Information Networking and Applications (AINA'14)*. Victoria, Canada: IEEE, May 2014.

[5] ——, "Towards Incentive-based Resource Assignment and Regulation in Clouds for Community Networks," in *Economics of Grids, Clouds, Systems, and Services*, ser. Lecture Notes in Computer Science, J. A. Altmann, K. Vanmechelen, and O. F. Rana, Eds. Zaragoza, Spain: Springer International Publishing, Sep. 2013, vol. 8193, pp. 197–211.

[6] U. C. Buyuksahin, A. M. Khan, and F. Freitag, "Support Service for Reciprocal Computational Resource Sharing in Wireless Community Networks," in *5th International Workshop on Hot Topics in Mesh Networking (IEEE HotMESH 2013), within IEEE WoWMoM*. Madrid, Spain: IEEE, Jun. 2013, pp. 1–6.

[7] A. M. Khan and F. Freitag, "Exploring the Role of Macroeconomic Mechanisms in Voluntary Resource Provisioning in Community Network Clouds," in *Distributed Computing and Artificial Intelligence, 11th International Conference*, ser. Advances in Intelligent Systems and Computing, S. Omatu *et al.*, Eds. Salamanca, Spain: Springer International Publishing, Jun. 2014, vol. 290, pp. 269–278.

[8] A. M. Khan, L. Sharifi, L. Veiga, and L. Navarro, "Clouds of Small Things: Provisioning Infrastructure-as-a-Service from within Community Networks," in *2nd International Workshop on Community Networks and Bottom-up-Broadband (CNBuB'2013), within IEEE WiMob*. Lyon, France: IEEE, Oct. 2013, pp. 16–21.

[9] J. Jiménez *et al.*, "Deploying PaaS for Accelerating Cloud Uptake in the Guifi.net Community Network," in *International Workshop on the Future of PaaS 2014, within IEEE IC2E*. Boston, Massachusetts, USA: IEEE, Mar. 2014.

[10] ——, "Supporting cloud deployment in the Guifi.net community network," in *5th Global Information Infrastructure and Networking Symposium (GIIS'13)*. Trento, Italy: IEEE, Oct. 2013, pp. 1–3.

[11] M. Selimi *et al.*, "Experiences with Distributed Heterogeneous Clouds over Community Networks," in *SIGCOMM Workshop on Distributed Cloud Computing (DCC'14), within ACM SIGCOMM*. Chicago, USA: ACM, Aug. 2014.

[12] M. Selimi and F. Freitag, "Towards Application Deployment in Community Network Clouds," in *14th International Conference on Computational Science and Its Applications (ICCSA'14)*. Guimaraes, Portugal: Springer, Jul. 2014.

[13] M. Selimi *et al.*, "Cloud-based extension for Community-Lab," in *22nd International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS'14)*. Paris, France: IEEE, Sep. 2014.

[14] D. P. Anderson, "BOINC : A System for Public-Resource Computing and Storage," in *5thIEEE/ACM International Workshop on Grid Computing*, Pittsburgh, USA, Nov. 2004, pp. 4–10.

[15] A. L. Beberg *et al.*, "Folding@home: Lessons From Eight Years of Volunteer Distributed Computing," in *8th IEEE International Workshop on High Performance Computational Biology (HiCOMB'09), within IPDPS*. Rome, Italy: IEEE, May 2009, pp. 1–8.

[16] B. Chun *et al.*, "PlanetLab: An Overlay Testbed for Broad-Coverage Services," *ACM SIGCOMM Computer Communication Review*, vol. 33, no. 3, pp. 3–12, Jul. 2003.

[17] J. Cappos, I. Beschastnikh, A. Krishnamurthy, and T. Anderson, "Seattle: a platform for educational cloud computing," in *40th ACM Technical Symposium on Computer Science Education (SIGCSE'09)*. Chattanooga, USA: ACM, Mar. 2009, pp. 111–115.

[18] X. Shen, H. Yu, J. Buford, and M. Akon, *Handbook of Peer-to-Peer Networking*. Springer Heidelberg, 2010, vol. 1.

[19] S. Distefano and A. Puliafito, "Cloud@Home: Toward a Volunteer Cloud," *IT Professional*, vol. 14, no. 1, pp. 27–31, Jan. 2012.

[20] K. Chard, K. Bubendorfer, S. Caton, and O. F. Rana, "Social Cloud Computing: A Vision for Socially Motivated Resource Sharing," *IEEE Transactions on Services Computing*, vol. 5, no. 4, pp. 551–563, Jan. 2012.

[21] M. Punceva *et al.*, "Incentivising resource sharing in social clouds," *Concurrency and Computation: Practice and Experience*, Mar. 2013.

[22] M. Gall, A. Schneider, and N. Fallenbeck, "An Architecture for Community Clouds Using Concepts of the Intercloud," in *27th International Conference on Advanced Information Networking and Applications (AINA'13)*. Barcelona, Spain: IEEE, Mar. 2013, pp. 74–81.

[23] R. Buyya, R. Ranjan, and R. N. Calheiros, "InterCloud: Utility-Oriented Federation of Cloud Computing Environments for Scaling of Application Services," *Algorithms and Architectures for Parallel Processing*, vol. 6081, pp. 20–31, Mar. 2010.

[24] R. Moreno-Vozmediano, R. S. Montero, and I. M. Llorente, "IaaS Cloud Architecture: From Virtualized Datacenters to Federated Cloud Infrastructures," *Computer*, vol. 45, no. 12, pp. 65–72, Dec. 2012.

[25] B. Satzger *et al.*, "Winds of Change: From Vendor Lock-In to the Meta Cloud," *IEEE Internet Computing*, vol. 17, no. 1, pp. 69–73, Jan. 2013.

[26] D. Vega, L. Cerda-Alabern, L. Navarro, and R. Meseguer, "Topology patterns of a community network: Guifi.net," in *1st International Workshop on Community Networks and Bottom-up-Broadband (CNBuB 2012), within IEEE WiMob*. Barcelona, Spain: IEEE, Oct. 2012, pp. 612–619.

[27] R. Birke *et al.*, "A delay-based aggregate rate control for P2P streaming systems," *Computer Communications*, vol. 35, no. 18, pp. 2237–2244, Nov. 2012.

[28] A. H. Payberah, F. Rahimian, S. Haridi, and J. Dowling, "Sepidar: Incentivized Market-Based P2P Live-Streaming on the Gradient Overlay Network," in *International Symposium on Multimedia (ISM'10)*. IEEE, Dec. 2010.