Distributed Architecture for Cloud System tailored for Wireless Community Networks

Technical Report #UPC-DAC-RR-XCSD-2013-4 – May 22, 2013

Amin M. Khan, Ümit C. Büyükşahin, Felix Freitag

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

***Abstract*—In community networks, individuals and local organizations 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. Most current community networks use wireless links for the node interconnection, applying off-the-shelf wireless equipment. While IP connectivity over the shared network infrastructure is successfully achieved, the deployment of applications in community networks is surprisingly low. To address the solution of this problem, we propose in this paper an architecture for building a cloud system that will provide Infrastructure-as-a-Service (Iaas) using resources from the community network. Our focus is also to incentivize the contribution of computing and storage as cloud resources to community networks, in order to stimulate the deployment of services and applications. Our final goal is the vision that in the long term, the users of community networks will not need to consume applications from the Internet, but find them within the wireless community network.**

***Index Terms*—wireless mesh networks; community networks; cloud computing; incentive mechanisms**

I. Introduction

Wireless community networks are an emergent model of infrastructure that aims to satisfy a community's demand for Internet access and ICT services. Most community networks originated in rural areas which commercial telecom operators left behind when deploying the broadband access infrastructure for the urban areas. Different stakeholders of such a geographic area teamed up to invest, create and run a community network as an open telecommunication infrastructure based on self-service and self-management by the users [1].

Current community networks use mainly wireless technology to interconnect nodes. With the commoditization of optical fiber, some community networks however have also started providing broadband services combining both technologies (e.g. guifi.net with fiber to the farm, FTTF [2]).

Community networks share, to a greater or lesser extent, the following common characteristics:

- they apply network neutrality such that the bandwidth capacity is limited only by the physical constraints of the deployed technologies.
- are public utilities available for use on equal terms by any party (private, public, commercial) connected to it within the community it serves.
- provide infrastructure which on the macro-level is community-owned, while on the micro-level of equipment is owned by the individual participants that contributed it.

Community networks are a successful case of resource sharing among a collective. The resources shared are networking hardware but also each community network participant's time he/she donates, in different extent, for maintaining the network. While the community network infrastructure is the sum of the individual contributions of wireless equipment, the network operation is achieved by the contribution of time and knowledge of the participants, even under the decentralized management of the equipment, since the node owner ultimately has the full access and control of his/her network device.

Resource sharing in community networks from the equipment perspective refers in practice to the sharing of the nodes' bandwidth. This sharing enables that traffic from other nodes is routed over the nodes of different node owners. This is done in a reciprocal manner which allows community networks to successfully operate as IP networks. Computing and storage resource sharing, such as is now common practice in today's Internet through Cloud computing, hardly exists in community networks. So any service offered in community networks runs on machines exclusively dedicated to a single member.

In Figure 1, some node types of a community network are depicted. The picture shows typical community nodes with a router and some server (or client machine) attached to it. A community network distinguishes between super nodes and client nodes. Super nodes have at least two wireless links, each to the other super nodes. Some super nodes are placed strategically

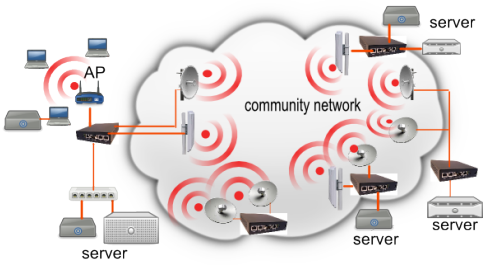

Figure 1. Nodes in a Community Network

in some geographic area to improve the community network's backbone and thus consist only of the wireless router. Other super nodes are installed in the community network participant's premises. In that latter case, such as shown in some nodes in Figure 1, servers behind the router are connected to offer services and applications to the community network. Client nodes only connect to a super node, but do not route any traffic. In Figure 1, some clients nodes are shown which are connected to the access point (AP) of a super node. Topological analysis of the Guifi.net community network [3] indicates that from approximately 17000 analysed nodes of Guifi.net, 7% are super nodes while the others are client nodes.

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 wireless routers. 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 community nodes. We note that client nodes could principally also contribute cloud resources. We therefore centre the contribution of this paper towards how to incentivize to bring together these computation and storage hardware already attached to the wireless routers of the community network into a community cloud.

In the following sections we present our proposal of the main components that the architecture for a community cloud should have. In section II we discuss the requirements for tailoring a cloud system to the community networks. In section III we describe how this architecture would be applicable to the topology of current community network deployments in which the interconnection and traffic routing is done by super nodes. In section IV we describe the conceptual overview and in section V overall design and architecture of a cloud management system that can provide services to the applications. In section VI we discuss the related work, and in section VII we conclude our findings and discuss about future work.

II. Requirements

A community cloud is a combination of a number of cloud systems running independently by the different community members. Moreover, the amount and quality of the resources available at each individual cloud can vary a lot. This is very much different from the existing commercial public clouds which are deployed on data centres using clusters of mostly homogeneous computers. This is also different from private and hybrid clouds where resources, though not as abundant as data centres, are still grouped into larger entities.

This means that in a hybrid cloud, there are a handful of partners, each with may be a few hundred to thousands of machines. In contrast, in community clouds there may be tens or even hundreds of partners, but each partner may have only a few tens of machines. This particular make-up of a community cloud requires careful attention to a different set of requirements.

Following requirements provide the foundation for the design and architecture of the community cloud system, that we discuss in sections IV and V. These need to be satisfied for a community cloud to be deployed and adopted successfully by the community.

A. Autonomy

Individual cloud systems are set up and managed independently by different owners. We cannot assume or require prior coordination or even trust between different cloud owners. This means that each cloud owner can take decisions about his or her cloud set-up without negotiating with other partners beforehand. The main requirement for a cloud owner for participating in a community cloud is that the local cloud set-up should adhere to the public API provided by the community cloud. In addition, it should contribute some set of mutually agreed upon resources to the community.

B. Security

With multiple independent cloud operators, security becomes even more important in a community cloud [4]. The data and applications running on different cloud systems should be protected from unauthorized access. Similarly, the cloud applications should not adversely affect the local machines. There are many other security challenges [5] that need to be addressed for ensuring users' trust in the system.

C. Self-Management

Different nodes can join and leave the community cloud at any time. Community cloud should self-manage itself and continue providing services without disruption. One important aspect is the coordination between different cloud owners that become part of the *federated* community cloud.

D. Utility

For wide adoption of community cloud, it should provide applications as a service that are valuable for the community. The driving factor for the growth of the community cloud would be the number of useful application available. Installing and using the applications should be straightforward and should require little overhead from the users.

E. Ease of Use

The majority of the users of the community cloud will be the non-technical ones who should not be required to understand the intricacies of the cloud infrastructure. Setting up nodes for deployment should be simple and straightforward. Similarly, managing and updating the cloud software on the nodes should be as automatic as possible.

F. Incentives for Contribution

A community cloud builds on the contribution of the volunteers in terms of computing, storage and network resources. For community cloud to be sustainable, incentive mechanisms are needed to encourage users to actively participate in the system and dedicate resources to the cloud.

G. Support for Heterogeneity

The hardware in a community cloud can have quite varying characteristics. There are powerful machines with good network connectivity and abundant storage space. On the other end, there are less powerful machines with limited CPU, RAM, disk space and bandwidth. The software for community cloud should handle this heterogeneity 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 *meta-cloud* [6] that obviates the need to customize the applications specific to each cloud architecture.

This is essential for the community cloud which results from federation of many independently managed clouds. Each cloud may be using a different virtual machine manager (VMM) that may provide a different set of API. Providing a standard API for the community cloud ensures that applications:

- written for one community cloud can also be deployed for another community cloud in the future
- can be easily deployed on new cloud architectures as they are integrated into the community cloud

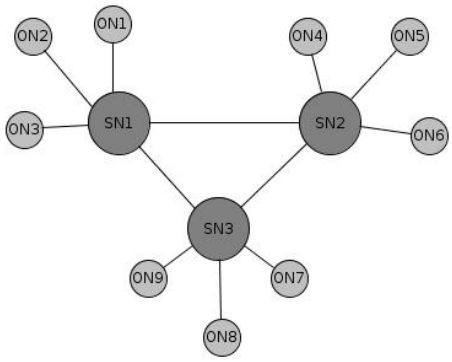

Figure 2. Overlay Network of Super Nodes and Ordinary Nodes in a Community Cloud

III. Distributed Architecture using Super Nodes

A. Context of the community network topology

The community mesh network generally includes different node types and each type plays a different role in the network. For example, Guifi.net [7], which is considered the largest meshed community network worldwide, includes two main types of nodes, according to [3]: *terminal nodes* which represent the end user nodes, and *hubs* which serve traffic to end users. In this network, each terminal node has a unique connection to a hub that routes traffic, and hubs can have many connected terminal nodes [3].

The architecture of a community cloud has to consider the topology of the community network which the cloud will be deployed on. Considering the typical community nodes explained above and the analysis of the community network topology [3], a hierarchical architecture [8] for community clouds is suggested. In this architecture, each super node is responsible for the management of a set of attached nodes. From the perspective of the attached nodes, these super nodes act as a centralized unit to manage the cloud services. These super nodes connect physically between other super nodes and logically in an overlay network to other cloud managing nodes.

This hierarchical architecture can be classified into the two main classes of fully decentralized and centralized systems [8]. If the design of the architecture is done towards a centralized systems, advantages include efficient search and control, while if it is decentralized, load-balancing, robustness and failure tolerance would be the benefits. There are several large-scale distributed applications that using a hierarchical designs achieved great success, such as Kazaa[1] and Skype[2].

B. Architecture and design

Figure 2 depicts the overlay network that results from the hierarchical architecture of the community cloud,

[1]http://www.kazaa.com
[2]http://www.skype.com

having ordinary nodes (ON) and super nodes (SN). ONs behave both as provider and requester in the cloud system. That means, at different times they can both request a resource or provide a resource. SNs are dedicated machines which are mainly responsible for coordinating and managing the ONs.

1) Ordinary Nodes (ONs): Each ON is assigned to a SN called *parent-SN* and holds the necessary information about it. In addition, each ON maintains locally a list called *ON_SNList* which contains the metadata of other SNs. With the information in this list, an extended registration of an ON in the system can done.

When an ON needs some resources, it sends a request to its parent-SN. Moreover, ONs periodically send a heartbeat message to their parent-SN to inform about their aliveness and inform about their current status.

2) Super Nodes (SNs): Each SN is responsible for a set of ONs and stores their metadata in a structure called *SN_ONList*. This list has to be updated after each resource sharing operation. Besides, each SN has another list called *SN_SNList* which holds the metadata of other SNs. This list is refreshed periodically to update which SN can supply how much amount of resources. For this purpose, each SN publishes their own status to other SNs, e.g. by gossiping [9].

C. Coordination in Federated Clouds

Multiple super nodes in a community network can connect and form federated clouds [10]. Such federated clouds are transient and can grow or shrink and merge or split or form larger cloud systems [11]. When there is a sufficient number of sites in a federated cloud, some of the more resourceful SNs can take additional responsibility of the management and coordination for neighbouring SNs.

Figure 3 explains a possible scenario of how super nodes can become part of the community cloud. In the beginning, as shown in Figure 3(a), super nodes SN1 through SN3 have set up cloud systems at their sites. Super node SN4 is still going through the process as more ONs are going to connect to SN4. Next in Figure 3(b), SN1 and SN2 have connected to form a sub-cloud, whereas SN3 and SN4 have also connected to form a separate sub-cloud.

In Figure 3(c), all the four SNs are now part of a single community cloud. In addition, SN3 has comparatively more resources so it has taken additional management tasks for the cloud, thus becoming *hyper node*. In Figure 3(d), SN3 gets disconnected from the network. The community cloud now consists of nodes SN1, SN2, and SN4 only, and SN2 starts acting as a hyper node.

IV. Overview of Community Cloud Manager

A community network is managed and owned by the community. Nodes are managed independently by their owners. Nodes principally can enter and leave the system, with or without any notice. Nodes that form the backbone however, i.e. super nodes, are usually intended to be stable with permanent connectivity. Ordinary nodes do more frequently change their connectivity status. An architecture for the cloud platform that manages such infrastructure needs to be robust, self-managing and resistant to such churn. It should enable a user to connect his or her machines for using or contributing to the cloud at any instant.

The option for enabling a community cloud in a wireless mesh network on which we focus here is to deploy a *cloud managing* platform tailored to community networks on a super node. Existing open and commercial cloud managing platforms are, for example OpenNebula [12], OpenStack [13], Euclyptus[3], Nimbus[4], Aeolus[5], etc.

The conceptual overview for the cloud managing platform that we propose for community networks is shown in Figure 4. The ordinary nodes of the wireless mesh network are the hosts of the cloud and form the physical layer of the cloud architecture. The core layer residing in the super node contains the software for managing and monitoring the virtual machines on ordinary nodes. The front end layer provides the interface of the infrastructure service (Infrastructure-as-a-Service, IaaS) provided by the super node.

The components cloud coordinator, economic engine and social engine provide additional services for customizing cloud infrastructure to the community networks and are discussed in detail in section V, see Figure 6.

A. Super Nodes and Ordinary Nodes

The main difference between super nodes and ordinary nodes, from the point of view of cloud management, is that SNs support greater functionality for handling VMs. A super node has full installation of the cloud management software and so enables the user to manage and monitor VMs. In most cases, super node will be a comparatively stable node, most likely a hub from the wireless mesh network.

Ordinary nodes, on the other hand, only act as hosts for executing VMs. There is no VMM software present, so the VMs cannot be controlled from the ordinary nodes. ONs directly connect to some SN on the network which is responsible for management of VMs. Note that the ONs can belong to a home network and be a desktop or laptop, or they can be a cluster running at a university consisting of many servers.

A user can connect multiple machines as ONs to a super node on the local network. We term this setup a *cloud site* which is an independent installation of the cloud management software on the local SN. The cloud management software is responsible for handling all the

[3]http://www.eucalyptus.com
[4]http://www.nimbusproject.org
[5]http://www.aeolusproject.org

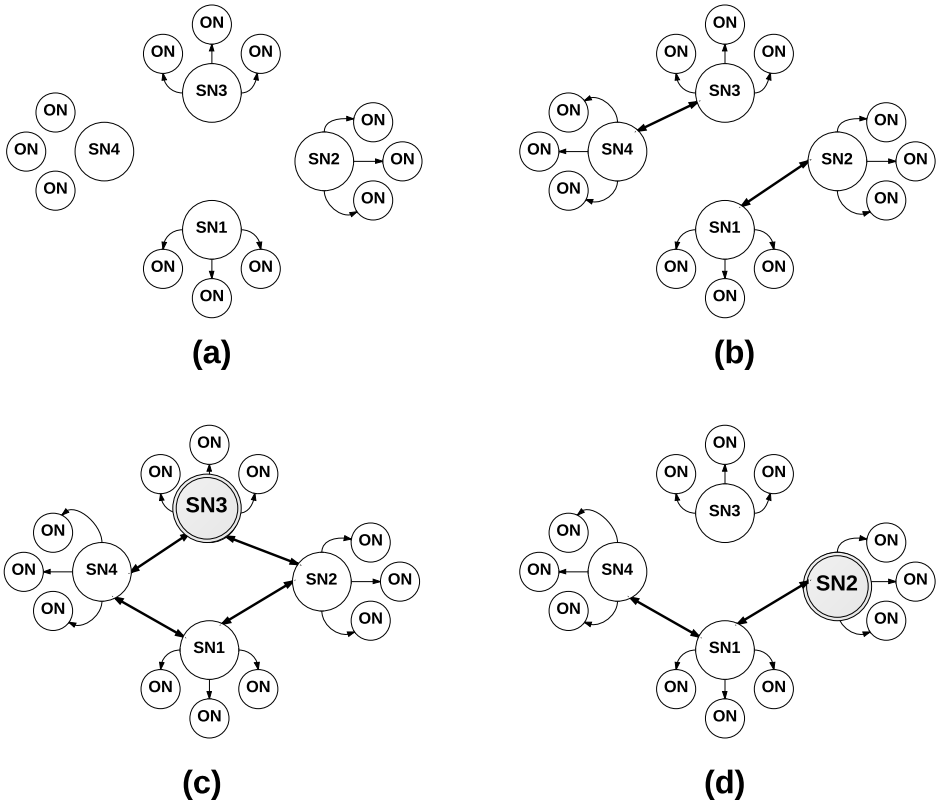

Figure 3. Super nodes connect to form community cloud. (a) SN1 through SN4 set up cloud software. (b) SN1 connects with SN2 and SN3 connects with SN4 forming sub-clouds. (c) SN1 through SN4 form a single cloud with SN3 as hyper node. (d) SN3 leaves community cloud and SN2 is the new hyper node.

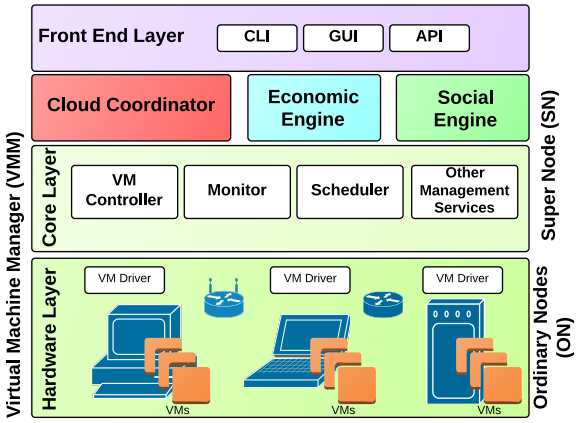

Figure 4. Conceptual Overview of the Community Cloud Manager

on-site ONs. The super node works independently from other SNs in the community network since the only resources it directly manages are the local ONs.

B. Different Configurations of SNs and ONs

In practice, the distinction between SN and ON may not be always clear because users can enable their machines for the community cloud in a variety of ways. Figure 5 shows some of the possible scenarios as explained below.

1) Only ordinary nodes: Consider that you have a couple of free machines that you want to dedicate to the community cloud. You do not want to concern yourself with the management and running of these machines. You can reset both the machines and install a specific operating system distribution provided by the community cloud. Your both machines will act as ordinary nodes (ONs), and you will configure your machines to connect to a nearby super node.

2) Super node with multiple ordinary nodes: Consider that you have a few machine available at your research lab. You want to set up a cloud infrastructure in your lab and you also want to contribute these resources to the community. You want full control of the VMs running on your machines and want to actively manage and monitor them.

You select one of the more powerful machines and install the cloud management software on it and this will become a super node in your local network. You reset the rest of the machines and install the operating system distribution provided by the community cloud.

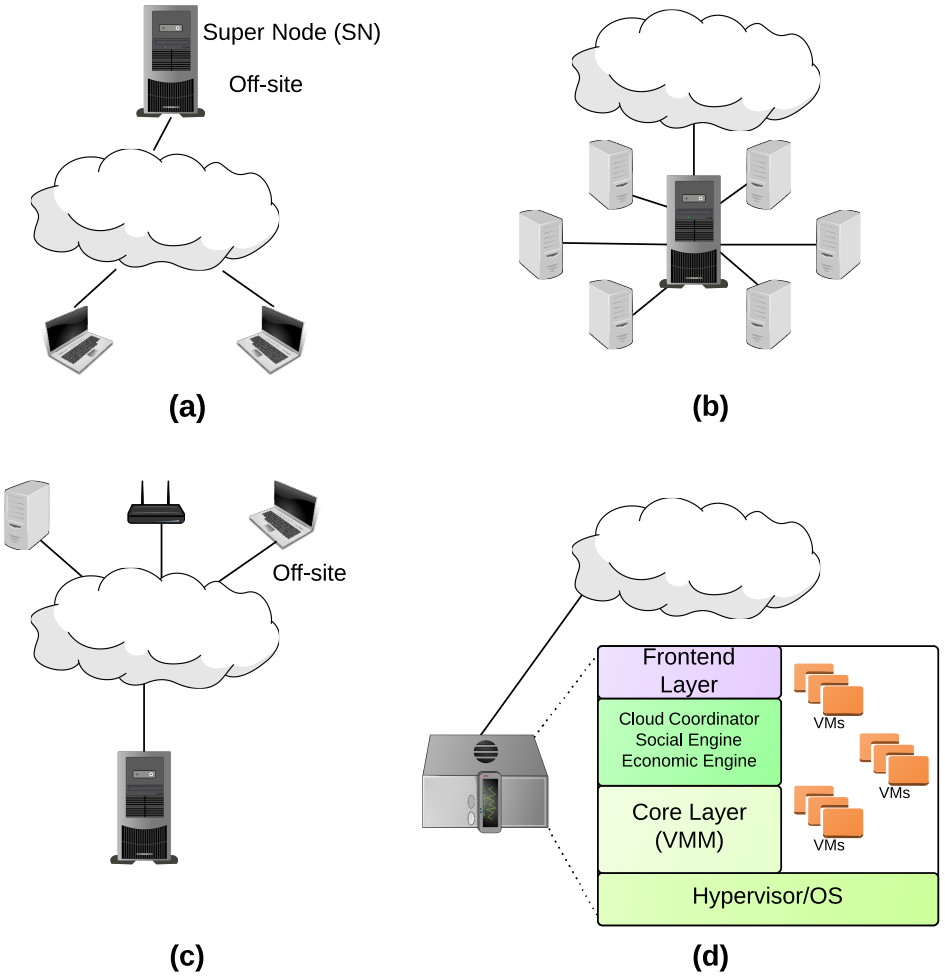

Figure 5. Different configurations of SNs and ONs. (a) Only ONs on-site. (b) SN with multiple ONs on-site. (c) Only SN on-site. (d) Community Cloud Box with SN and VMs

These machines become ordinary nodes (ONs), and they are configured to be controlled by your local SN.

Your local SN will publish the information about your new setup to other SNs in the community cloud. The cloud coordinator in your SN will allow your applications to take advantage of the resources contributed by others in the community.

3) Only super node: Consider that you have a hub of wireless mesh network in your apartment building. It is always on and connected to the Internet and has good bandwidth available. You want to deploy community cloud in your neighbourhood and your hub can take the responsibility of managing VMs running on other hosts (ONs) in your network. You can install the cloud management software on your hub and it will become a super node in your local network. Your SN will also coordinate with other SNs in the community cloud.

4) Super node and ordinary nodes as a single box: In community networks there might also be the situation that a clear super node cannot be identified. An example are local wireless mesh networks in cities in which all nodes communicate with all other nodes in signal range. The link characteristics may change dynamically and the connectivity topology of the mesh networks also changes as a result. In this case, while certain node may still assume the role of a super node, e.g. due to a more powerful hardware, decentralizing the role of the super node could be a more suitable approach.

For example, you have a single machine that you want to connect to community cloud but you cannot select a single reliable SN. You reset the machine and install the operating system distribution provided by the community cloud. You also install the cloud management software on the same machine. So you now have a super node that manages the VMs running on the same host.

In Figure 5(d) the architecture for the scenario of a decentralized cloud architecture is shown. It can be seen that the three layers Front End Layer, Core Layer and Hardware Layer are on a single node which we call *community cloud box*, a device attached to each node in a wireless mesh network. The cloud coordinator, economic

engine and social engine are distributed services in which all community cloud boxes participate.

V. Architecture of Community Cloud Manager

The core of cloud manager is the virtual machine manager (VMM) that is responsible for instantiating, scheduling and monitoring virtual machines on the hosts. There are some commercial and open source cloud management platforms available to manage public and private clouds. Among the most consolidated and popular open source tool are currently OpenNebula [10], [12] and OpenStack [13].

The cloud platform can be tailored for community networks by extending these existing tools and building components like cloud coordinator, economic engine and social engine on top of them. The virtual machine manager (VMM) consists of the following layers, which are common to most cloud computing architectures.

A. Hardware Layer

This consists of the physical infrastructure that is needed to run a cloud system. These include PCs, storage, and network. The hardware in the community networks mostly consist of ordinary nodes and wireless links provided by the mesh network.

The machines in the community network will have a standard operating system or some hypervisor software installed. In addition, there will be drivers to support the operation of VMs.

B. Core Layer

The core layer consists of components that are responsible for creation, allocation, scheduling, monitoring and management of VMs on the hosts. This has following main components, some of which are shown in Figure 4.

- Virtual Machines Controller
- Virtual Machines Scheduler
- Virtual Machines Monitor
- Hosts Manager
- Virtual Network Manager

The functionality of the core layer is already provided by tools like OpenNebula, 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 networks.

C. Cloud Coordinator

The cloud coordinator is responsible for the federation of the cloud resources which are independently managed by different super nodes. 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 with each other in a decentralized manner to exchange relevant information about managing the available resources.

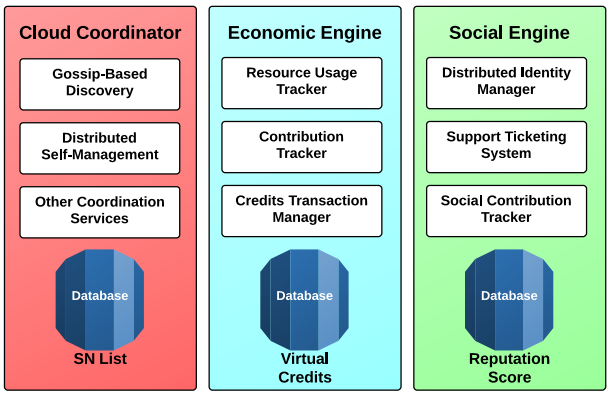

Figure 6. Functional overview of the distributed components of community cloud manager

By default applications running at a cloud site can only consume resources from the ONs directly managed by that particular super node. With the cloud coordinator, the infrastructure service can provide a unified view of the resources contributed by multiple sites. When pooling resources from multiple sites, the cloud coordinator applies a regulation mechanism fed by the economic engine and social engine to perform resource allocation.

The cloud coordinator will require a number of modules, some of which are highlighted in Figure 6.

1) Gossip-Based Discovery: The design of a community cloud manager follows a decentralized approach, so cloud coordinator relies on gossip-based [9] discovery mechanisms to manage overlay network of the super nodes in community cloud. The updated list of neighbouring super nodes is saved in SN_List database.

2) Distributed Self-Management: The community cloud has to efficiently manage the distributed resources in an autonomous way as ordinary and super nodes join and leave the network. This module handles this important function of the cloud coordinator.

D. 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 community cloud. Consider the commercial public clouds where providers get paid in hard currency. In contrast, for community cloud, the incentive for the providers is the utility that they get from the community cloud in return. 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 and we have discussed the important ones below, as shown in Figure 6.

1) Resource Usage Tracker: This modules connects with 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.

2) Contribution Tracker: The information from 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.

3) Credits Transaction Manager: This modules 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 to the community cloud. The challenge for this module is to handle these transactions in a secure manner within a distributed system.

E. Social Engine

The community cloud is as much a social construct as it is a technical construct. The existence of community cloud is not possible if there is a lack of participation from the community. The running of community cloud not only requires supply of technical resources like memory or bandwidth, but also the time and effort of the users who setup and manage the hardware.

Whereas 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 achieve this goal. These modules may not be integral to the cloud managing platform from a technical point of view, but nevertheless provide functionality necessary for the smooth running of the community cloud.

1) Distributed Identity Manager: This module manages the *global identity* of the users in a decentralized manner. This unique system-wide user ID is needed to track the usage and contribution by each user [14].

2) 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 to an increase in virtual credits which the user can spend for consuming resources from the community cloud.

3) 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 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 community cloud.

F. 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 application using the infrastructure service.

G. Interaction between different components

We discuss here some usage scenarios and explain how different components of the community cloud manager 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 frontend layer and submits a request for a VM instance. The request is forwarded to cloud coordinator that checks for availability at local ordinary nodes and also forwards the request to neighbouring SNs using SN_List database.

Cloud coordinator checks with identity manager component of social engine. This authenticates the user and confirms that user has access to the resources.

Cloud coordinator then checks the virtual credits database of economic engine to see if the user has sufficient credits available to fulfil the request. After confirming that user can consume the resources, the request is forwarded to scheduler in the core layer which selects the host on which VM will be executed.

The monitor in the core layer will provide the details of the consumed resources to the resource usage tracker component in the economic engine. Credits transaction manager in the economic engine will update the credits of the requester and provider of VM in virtual credits database. Contribution tracker in economic engine will update the details for the provider.

The requester can use the GUI in the frontend layer to get up-to-date 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 the GUI in the frontend layer and submits the details of his or 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 at this particular node. The cloud coordinator also forwards the details to neighbouring SNs using SN_List database.

Social contribution tracker component updates the values for the user in the reputation score database. It also contacts credits transaction manager component in the economic engine which updates virtual credits database for the user. Social contribution tracker and

credits transaction manager provide the details to the frontend layer and 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 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, support ticketing system keeps track of the feedback. It checks with identity manager component of the social engine to authenticate the user. Social contribution tracker then updates the reputation score database for the user.

Social contribution tracker also asks credits transaction manager to update virtual credits database for the user. The details are provided to the frontend layer and user can check the status of his or her virtual credits via GUI.

VI. Related Work

After the prevalence of public clouds [15], there is now increasing interest in providing cloud services by harvesting excess resources from the idle machines connected to the Internet [16]. Commercial clouds have dedicated resources that are financed by the users who pay in hard currency to use the cloud services. In contrast, building a cloud platform within a community mesh network requires incentives to encourage active participation from the members of the community. Previous distributed multi-owned computing platforms like Seti@Home [17] have relied on altruistic contribution of volunteer users. PlanetLab [18] requires for granting resource usage a prior fixed contribution before the services are made available.

At the level of participation in community networks, reciprocal resource sharing is in fact part of the membership rules or peering agreements of many community networks. The Wireless Commons License[6] (WCL) of many community networks states that the network participants that extend the network, e.g. contribute new nodes, will extend the network in the same WCL terms and conditions, allowing traffic of other members to transit on their own network segments.

Regarding incentive mechanisms, in the literature there are various incentive mechanisms which address different requirements [19]–[22]. None of these incentive mechanisms however target the particular situation of wireless community networks.

On the level of complete systems for community cloud computing [16], there are a few research prototypes. Skadsem et al. [23] provide applications for the communities by using local cloud services. Their work is similar to ours though they assume that the social mechanisms like trust in a small community do not require additional mechanisms for incentives. They envisage two usage scenarios for cloud applications in rural communities. First focuses on the distributed storage where they plan to provide an online message board for community for sharing photos and videos. The other deals with compute-intensive operations such as collaborative editing of video footage. They have built distributed storage in P2P systems, and they want to extend that to the community cloud after incorporating virtualization.

The Cloud@Home[7] [24] project has similar goals to harvest in resources from the community to meet peaks in demands. The aim is to work with open, commercial and hybrid clouds to make cloud federations. The system envisages ensuring Quality of Service (QoS) using a rewards and credit system, however the authors have not provided sufficient details to understand how these incentives will be designed.

CuteCloud [25] is an ongoing project that plans to use idle resources form users' commodity machines and dedicated servers. High-demanding jobs are assigned to dedicated servers while excess demand can be met from commodity machines. They presume that resources will be volunteered similar to Seti@Home [17] and do not address the problem of how to encourage users to devote resource. They aim to use the Nimbus[8] as virtual machines manager (VMM), with Xen[9] hypervisor running on dedicated machines and VirtualBox[10] on common users' machines.

Clouds@home[11] [26] project focuses on providing guaranteed performance and ensuring quality of service (QoS) even when using volatile Internet volunteered resources. They do not focus on incentive mechanisms.

P2PCS[12] [27] project has built a prototype implementation of a decentralized Peer-to-Peer Cloud System. It uses Java JRMI technology and build an IaaS system that provides very basic support for creating and managing VMs as. A *slice* is a group of multiple VMs connected in a single virtual network. It manages slices information in a decentralized manner using gossip protocols. They also do not address the issue of incentives.

We notice that none of the found related work propose and discuss clouds within wireless mesh community networks.

[6] http://guifi.net/es/ProcomunXOLN

[7] http://cloudathome.unime.it
[8] http://www.nimbusproject.org
[9] http://www.xen.org
[10] http://www.virtualbox.org
[11] http://clouds.gforge.inria.fr
[12] https://code.google.com/p/cloudsystem

VII. Conclusion

Wireless community networks would have additional value from services deployed on community clouds. A vast amount of applications could be deployed upon community clouds, boosting the usage and spread of the community network model.

We have proposed a distributed service architecture for providing cloud services that is tailored to the unique nature and conditions of community networks. We have discussed different scenarios for setting up cloud infrastructure in community networks and how the distributed service architecture can handle them.

We aim to develop software components on top of existing cloud managing platforms like OpenNebula that provide services for federation of cloud resources with regulated resource sharing to encourage active contributory participation of the community members to form and maintain the cloud infrastructure. Since community networks are volunteer organizations, we consider such services essential to assure a sustainable community cloud within community networks.

While this would assure the community cloud infrastructure to be created by the members' contributions, a next step is to assess the community cloud performance. So in future work we plan to run experiments to investigate the expected performance of such a distributed community cloud in wireless community networks.

Acknowledgement

This work was supported by the European Commission Framework Programme 7 within the Future Internet Research and Experimentation Initiative (FIRE), Community Networks Testbed for the Future Internet (CONFINE), FP7-288535, and CLOMMUNITY, FP7-317879. Support was also provided by the Universitat Politècnica de Catalunya BarcelonaTech and the Spanish Government through the Delfin project, TIN2010-20140-C03-01.

References

[1] F. A. Elianos, G. Plakia, P. A. Frangoudis, and G. C. Polyzos, "Structure and evolution of a large-scale Wireless Community Network," in *2009 IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks & Workshops*. IEEE, Jun. 2009.

[2] "Guifi.net new sections of fiber deployed to the farm," 2012. [Online]. Available: http://en.wikinoticia.com/Technology/internet/122595

[3] 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, Oct. 2012, pp. 612–619.

[4] F. Baiardi and D. Sgandurra, "Securing a Community Cloud," in *2010 IEEE 30th International Conference on Distributed Computing Systems Workshops*. Los Alamitos, CA, USA: IEEE, Jun. 2010, pp. 32–41.

[5] K. Bernsmed, M. G. Jaatun, P. H. Meland, and A. Undheim, "Thunder in the Clouds: Security challenges and solutions for federated Clouds," in *4th IEEE International Conference on Cloud Computing Technology and Science Proceedings*. IEEE, Dec. 2012, pp. 113–120.

[6] B. Satzger, W. Hummer, C. Inzinger, P. Leitner, and S. Dustdar, "Winds of Change: From Vendor Lock-In to the Meta Cloud," *IEEE Internet Computing*, vol. 17, no. 1, pp. 69–73, Jan. 2013.

[7] "Guifi.net: Open, Free and Neutral Network Internet for everybody." [Online]. Available: http://guifi.net

[8] B. Yang and H. Garcia-Molina, "Designing a super-peer network," in *Proceedings 19th International Conference on Data Engineering*. IEEE, 2003, pp. 49–60.

[9] O. Babaoglu and M. Jelasity, "Self-* properties through gossiping." *Philosophical transactions. Series A, Mathematical, physical, and engineering sciences*, vol. 366, no. 1881, pp. 3747–57, Oct. 2008.

[10] 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.

[11] O. Babaoglu, M. Jelasity, A.-M. Kermarrec, A. Montresor, and M. van Steen, "Managing clouds: A Case for a Fresh Look at Large Unreliable Dynamic Networks," *ACM SIGOPS Operating Systems Review*, vol. 40, no. 3, p. 9, Jul. 2006.

[12] "OpenNebula - Open Source Data Center Virtualization," http://opennebula.org. [Online]. Available: http://opennebula.org

[13] OpenStack, "Open Source Cloud Computing Software," http://www.openstack.org. [Online]. Available: http://www.openstack.org

[14] J. Chen, X. Wu, S. Zhang, W. Zhang, and Y. Niu, "A Decentralized Approach for Implementing Identity Management in Cloud Computing," in *2012 Second International Conference on Cloud and Green Computing*. IEEE, Nov. 2012, pp. 770–776.

[15] M. Armbrust, I. Stoica, M. Zaharia, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, and A. Rabkin, "A view of cloud computing," *Communications of the ACM*, vol. 53, no. 4, p. 50, Apr. 2010.

[16] A. Marinos and G. Briscoe, "Community Cloud Computing," *Cloud Computing*, vol. 5931, pp. 472–484, 2009.

[17] D. P. Anderson, J. Cobb, E. Korpela, M. Lebofsky, and D. Werthimer, "SETI@home: an experiment in public-resource computing," *Communications of the ACM*, vol. 45, no. 11, pp. 56–61, Nov. 2002.

[18] B. Chun, D. Culler, T. Roscoe, A. Bavier, L. Peterson, M. Wawrzoniak, and M. Bowman, "PlanetLab: An Overlay Testbed for Broad-Coverage Services," *ACM SIGCOMM Computer Communication Review*, vol. 33, no. 3, pp. 3–12, Jul. 2003.

[19] K. Zhang, N. Antonopoulos, and Z. Mahmood, "A taxonomy of incentive mechanisms in peer-to-peer systems: Design requirements and classification," *International Journal on Advances in Networks and Services*, vol. 3, no. 1, pp. 196–205, 2010.

[20] M. Babaioff, J. Chuang, and M. Feldman, "Incentives in peer-to-peer systems," in *Algorithmic Game Theory*. Cambridge University Press, 2007, pp. 593–612.

[21] M. Feldman and J. Chuang, "Overcoming free-riding behavior in peer-to-peer systems," *ACM SIGecom Exchanges*, vol. 5, no. 4, pp. 41–50, Jul. 2005.

[22] Y. Tang, H. Wang, and W. Dou, "Trust based incentive in P2P network," in *IEEE International Conference on E-Commerce Technology for Dynamic E-Business*, no. 90104020. IEEE Comput. Soc, 2004, pp. 302–305.

[23] M. K. Skadsem, R. Karlsen, G. Blair, and K. Mitchell, "Community Cloud - Cloud Computing for the Community," in *1st International Conference on Cloud Computing and Services Science*. Setubal, Portugal: SciTePress, May 2011, pp. 418–423.

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

[25] D. Che, M. Zhu, J. Fairfield, and M. Khaleel, "CuteCloud–Putting "Credit Union" Cloud Computing into Practice," in *Research in Applied Computation Symposium (RACS 2012)*, San Antonio, USA, Oct. 2012.

[26] S. Yi, E. Jeannot, D. Kondo, and D. P. Anderson, "Towards Real-Time, Volunteer Distributed Computing," in *11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid 2011)*. Newport Beach, USA: IEEE, May 2011, pp. 154–163.

[27] O. Babaoglu, M. Marzolla, and M. Tamburini, "Design and implementation of a P2P Cloud system," in *27th Annual ACM Symposium on Applied Computing (SAC '12)*, New York, USA, Mar. 2012, pp. 412–417.