Requirements for a holistic abstract network and service architecture

Deliverable D3.1

Date: 30 June 2013

Version: 1.0
This page is left in blank intentionally.
This deliverable describes the scenarios of community networks with regards to the features which are considered relevant for the construction and deployment of community clouds. Given these conditions and their meaning, an architecture is proposed that takes into account these circumstances. Requirements at different levels are then described which the community cloud solution should be able to satisfy in order to successfully deploy and maintain clouds in community networks.
Executive Summary

In order to derive the requirements for community clouds, the features and characteristics that are considered relevant for community clouds are explained. A remarkable finding is the observation that while most nodes are contributed completely by a member, some nodes are crowd-funded. This fact is important since it allows to see different options for resource contribution and organization of community clouds.

The perspective of the topology of community networks is also studied. Since the community cloud is to be deployed in a real network, it must fit into the physical conditions of this system. This analysis allows to envision a distributed architecture for the community cloud.

High and low-level requirements are then described which the community cloud should satisfy in order to be feasible, accepted and sustainable in community networks. A review of existing systems is performed, while it is noted that none of the related works was designed for our scenario. Based on the previous analysis, our overall approach foresees then the options to extend off-the-shelf cloud management platforms which could be federated and other more decentralized management and cloud organizations in community networks.
# Table of Contents

1. **Introduction** ........................................................................................................... 7

1.1. **Contents of the deliverable** ............................................................................... 7

1.2. **Relationship to other COMMUNITY deliverables** ........................................... 7

2. **Scenarios Of Community Clouds** ........................................................................ 8

2.1. **Sharing of resources** ....................................................................................... 8

2.1.1. **Shared bandwidth** ..................................................................................... 8

2.1.2. **Shared Internet access with federated proxies** ........................................... 8

2.1.3. **Shared time and knowledge** .................................................................... 8

2.1.4. **Crowd funding** .......................................................................................... 8

2.2. **Social aspects** .................................................................................................... 9

2.3. **Users** .................................................................................................................. 9

3. **Distributed Architecture Using Super Nodes** ...................................................... 10

3.1. **Node types** ........................................................................................................ 10

3.2. **Distributed architecture in Community Network topology** ............................. 11

3.3. **Local and federated clouds** ............................................................................ 11

3.3.1. **Only ordinary nodes** ................................................................................ 12

3.3.2. **Super node with multiple ordinary nodes** ................................................ 12

3.3.3. **Only super node** ...................................................................................... 13

3.3.4. **Super node and ordinary nodes as a single box** ....................................... 13

4. **High-level Requirements** .................................................................................... 14

4.1. **Distributed node ownership** ........................................................................... 14

4.2. **Cloud resource management and autonomy** .................................................. 15

4.3. **Distributed and centralized resource location** .............................................. 15

4.4. **Support for heterogeneity** ............................................................................. 15

4.5. **Free and Open Source Software** .................................................................. 15

4.6. **Familiarity with system** ................................................................................ 15

4.7. **Ease of use** ...................................................................................................... 16

4.8. **Utility** ............................................................................................................... 16

4.9. **Self-management** ........................................................................................... 16

4.10. **Developer community** .................................................................................. 16

4.11. **Standard API** ............................................................................................... 16

4.12. **Security** .......................................................................................................... 17

4.13. **Incentives for contribution** ............................................................................ 17

4.14. **Summary** ........................................................................................................ 17

5. **Low-level Requirements** ................................................................................... 18

5.1. **Hardware layer** ............................................................................................ 18

5.2. **Core layer** ....................................................................................................... 19

5.3. **Cloud coordinator** ........................................................................................ 19

5.4. **Economic engine** .......................................................................................... 19

5.5. **Social engine** .................................................................................................. 19

5.6. **Frontend layer** ............................................................................................... 20

6. **Candidates And Comparison** ........................................................................... 21
Figures

Figure 1. Topology Of Guifi.net In The Area Of Barcelona And Super Nodes In A Community Network.

Figure 2. Super Nodes Connect To Form Federated Community Cloud.

Figure 3. Conceptual Overview Of The Community Cloud Manager.

Tables
1. Introduction

1.1. CONTENTS OF THE DELIVERABLE

The content of this deliverable consists in a definition of requirements for the abstract service architecture considering the scenarios of community clouds, including the perspective of usage, users, operational environment and ecosystem, and the inter-operation or federation with other clouds, which could be community clouds or other types of clouds.

1.2. RELATIONSHIP TO OTHER COMMUNITY DELIVERABLES

The content of deliverable D3.1 is input for WP2, in particular for task T2.1. D3.1 has received input from WP4 task T4.1 on the cases of the pilot studies.

The initial set of requirements is documented in this deliverable D3.1 (M06), while revisions of them will be reported in D3.4 (M24).

Based on the requirements of D3.1, a more consolidated definition of a holistic and abstract network and service architecture for community clouds is to be reported in D3.2 (M12) and a final version will be reported in D3.4 (M24).
2. Scenarios of Community Clouds

2.1. SHARING OF RESOURCES

Community networks are a successful case of resource sharing among a collective. The resources shared include networking, hardware but also each community network participant’s time he/she donates, in a different extent, for maintaining the network. Computing and storage resource sharing through networks, such as is common practice nowadays through Cloud Computing, however is lacking in community networks. Thus, any service offered in community networks runs on machines exclusively dedicated to a single member.

2.1.1. SHARED BANDWIDTH

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.

2.1.2. SHARED INTERNET ACCESS WITH FEDERATED PROXIES

Internet access in the Guifi community network is offered through a group of federated proxies. There are currently more than hundred Internet proxies declared in Guifi.net. Proxies are hosted and maintained voluntarily by various individuals.

In Guifi.net, a registered member can gain access to an Internet proxy by requesting being part of the group of users that can gain Internet access. A separate user account is being issued for accessing those proxies and is explicitly used for Internet access.

A registered proxy user can have access to any of the federated proxies in Guifi.net. Furthermore, user account information is maintained via the Lightweight Directory Access Protocol (LDAP).

2.1.3. SHARED TIME AND KNOWLEDGE

The community network’s operation and maintenance is the product of the aggregated effort of its individual members in the same way as the network infrastructure is the aggregated wireless equipment contributed by its members.

Individuals effort translates into time and knowledge spent to configure and further improve network operability. In such a decentralized infrastructure, community network individuals are responsible for their own network devices on which they have full control. Therefore, the community network as a whole achieves a level of sustainability via the collaborated effort of the member contributions.

2.1.4. CROWD FUNDING

Typically, each node of a community network belongs to one owner (in most cases it’s the person or organization who bought it and on whose premises the node is located). However, there are nodes in Guifi.net that have been crowd funded instead. Crowd funding of nodes has been adopted in various cases due to necessary infrastructure needs over groups of people. For example, users of isolated geographical zones of Guifi.net co-established super nodes, which will
D3.1 Requirements for a holistic network and service architecture

be explained below, to gain connectivity to other zones. In such cases, the overall cost of a node setup was shared over multiple individuals. The location of such crowd funded nodes often follows strategic considerations, aiming to improve or optimise marginal resources in terms of performance or connectivity with the help of the added infrastructure. As a concept, crowd funding is yet another example of a collaborative contribution towards marginal benefits.

2.2. SOCIAL ASPECTS

As a means to deal with physical constraints Guifi.net is organized into zones. Practically, a zone corresponds to a spatial topology such as a village, a small city, a region, or district of a larger city. Each zone might consist of one to an arbitrary number of geographically related interconnected nodes.

From a social perspective, there are two types of user communities, or social networks formed throughout Guifi.net, each maintaining its own mailing list. First, there exists a global network community where general technical and organisational issues are resolved. On that global level there are participant members from the whole Guifi.net network in addition to external participants who have an interest and motivation to contribute to the community network's affairs. Additionally, there are local groups, or social networks, formed by node owners within a specific zone or neighboring zones. Members of local groups participate actively in zone related decisions, attending weekly meetings and collaborating through the respected local mailing lists. The organization and activity of such local groups is not strictly defined and is mainly driven by its members’ interests, their available time and knowledge. In general, the main networking aspects such as address range allocation matters are resolved in the global community level since they affect the overall community network whereas local organisational aspects are discussed in the local group level and always driven by individuals.

2.3. USERS

Members in community networks principally act as both consumers and producers. First, members can be considered as producers since they provide infrastructure and time to the networks. Furthermore, members can also be regarded as consumers since they utilize the network's available services.

However, a community network is not solely based on infrastructure resource contributions. Users are also expected to contribute time and knowledge to manage those resources. Time is needed for instance for maintenance tasks (tasks which might require technical knowledge or not). Additionally, technical knowledge is needed as well to properly configure given IP network components.

Community network contributions, both infrastructural and time or knowledge spent by users are purely voluntary, thus, no money rewards are offered. Typically, contributing members dedicate their limited free time at will to the community network. It should be noted that in Guifi.net member expertise varies significantly throughout the owners of more than 20000 nodes and a limited fraction has a technical or engineering background. Therefore, technical contributions originating from the community network itself can only be moderate. In general usage, members target off-the-self deployments of tools and applications to cover their needs.
3. Distributed Architecture using Super Nodes

3.1. NODE TYPES

The nodes in a community network are further distinguished between super nodes (SNs) and client or ordinary nodes (ONs). The main requirement of SNs is the existence of at least two wireless links towards other super nodes. This backhaul consisting of interconnected SNs is typically addressed as the backbone of the community network. Some SNs are strategically placed in geographic areas to improve the community network’s backbone and thus consist solely of a wireless router to serve routing purposes. Other SNs are installed in the community network participant’s premises. In the latter case, sometimes servers attached to the router are connected to offer services and applications to the community network. An ON can join the network, gain access to available services by connecting to a SN that is in turn responsible to route its traffic.

Figure 1 shows some the topology of the Guifi.net community network in the area of Barcelona. The picture illustrates some SNs with servers attached to them. Note that there is still a plethora of SNs in a community network that do not have any servers attached to them.

Figure 1: Topology of Guifi.net in the area of Barcelona and super nodes in a community network.
3.2. Distributed Architecture in Community Network Topology

Any potential 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 [Vega], a hierarchical architecture [Yang] 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 system architecture could be further classified into the two main classes, a centralized and a fully decentralized one [Yang]. A potential centralized architecture design could offer several advantages including efficient lookups and optimized resource management, while a decentralized architecture could benefit from better load-balancing, robustness and failure tolerance. There have been several successful use cases of distributed large-scale applications that achieved good results, such as Kazaa and Skype.

3.3. LOCAL AND FEDERATED CLOUDS

Multiple super nodes in a community network can connect and form federated clouds [Moreno]. Such federated clouds are transient and can grow or shrink and merge or split and form larger cloud systems. 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 2 explains a possible scenario of how super nodes can become part of the community cloud. In the beginning, as shown in Figure 2(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 2(b), SN1 and SN2 SN1 and SN2 have connected to form one federated cloud, whereas SN3 and SN4 have also connected to form another federated cloud.

In Figure 2(c), all the four SNs are now part of a single federated 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 2(d), SN3 gets disconnected from the network. The federated community cloud now consists of super nodes SN1, SN2, and SN4 only, and SN2 starts acting as a hyper node for this cloud.

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. We foresee the following possibilities explained in separate sections:
D3.1 Requirements for a holistic network and service architecture

3.3.1. **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. Both machines will act as ordinary nodes (ONs), and you will configure your machines to connect to a nearby super node.

3.3.2. **SUPER NODE WITH MULTIPLE ORDINARY NODES**

Consider that you have a few machines 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.

Figure 2: Super nodes connect to form federated community cloud. (a) SN1 through SN4 set up local community cloud. (b) SN1 connects with SN2 and SN3 connects with SN4 forming two federated community clouds. (c) SN1 through SN4 form a single federated community cloud with SN3 as hyper node. (d) SN3 leaves federated community cloud and SN2 is the new hyper node.
D3.1 Requirements for a holistic network and service architecture

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. 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.

3.3.3. 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.

3.3.4. 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.
A community cloud is envisioned as a combination of a number of cloud systems functioning independently in different community network zones. Moreover, a high level of heterogeneity is expected between different cloud systems since the amount and quality of the resources available at each individual cloud can vary a lot. Such an environment diverges from conventional commercial public clouds which are mainly deployed on data centres consisting of homogeneous machines. It further exhibits different properties from private and hybrid clouds where resources, though not as abundant as data centres, are still grouped into larger entities. In a hybrid cloud, there are a handful of partners, each with a few hundreds to thousands of machines whereas in community clouds there might be tens or even hundreds of partners where each partner might have only a few tens of machines. This particular structure of a community cloud requires careful attention since it exposes a different set of requirements.

The following requirements provide the foundation for the design and architecture of a community cloud system. Mind that such requirements need to take into account the different social and technical aspects present in a community network discussed in the previous section. These requirements need to be satisfied by a potential community cloud deployment and adopted successfully by the community.

4.1. DISTRIBUTED NODE OWNERSHIP

The typical ownership of a Guifi node is that it belongs to the individual who bought the hardware. Exceptions are crowd-funded nodes which, however, are only a small proportion of all Guifi nodes. Being most nodes on premises of the owner, the physical access to Guifi nodes is only possible by and through the node owner.

The node owner can give remote access to the software system to other people. Principally the node owners can have a private root password.

The ownership of community cloud nodes can be seen as follows:

1. private ownership of cloud resources
2. crowd-funded nodes cloud resources

As a consequence of the first case, it is likely the cloud resource will be located on the owner's premises.

In the second case, a strategic location of the cloud resource could be possible, with the goal of improving the performance in a way that is beneficial to all the sponsors. In this case, a data centre location could be the chosen solution in which all sponsored cloud resources are located in one place.

The community cloud architecture must consider both situations, cloud resources localized centrally and cloud resources spread over the nodes of the community network.
Cloud resource contributors might accept a unique resource allocation policy for cloud resources or might prefer a particular policy that fits to their personal conditions.

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 setup without negotiating with other partners beforehand. The main requirement for a cloud owner for participating in a community cloud is that the local cloud setup should adhere to the public API provided by the community cloud. In addition, it should contribute some set of mutual agreement upon resources to the community.

4.3. DISTRIBUTED AND CENTRALIZED RESOURCE LOCATION

In the case of a 100% private ownership of each community cloud nodes, cloud resources will be located at the user's premises. In the case of a shared ownership of each community cloud node (e.g. crowd funded), cloud resources will be located centrally.

4.4. SUPPORT FOR HETEROGENEITY

We may expect different kind of hardware to be contributed as cloud resources on which different OS are installed.

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 cloud resource hardware may support different virtualization solutions.

- Support different hypervisors (LXC for processors without hardware enabled virtualization [LXC]).
- Support different CPU, memory, storage, bandwidth capacity of resources.

The software for community cloud should handle this heterogeneity seamlessly.

4.5. FREE AND OPEN SOURCE SOFTWARE

Reusing of generally accepted approaches, free and open source software is mandatory to ease reviewing and trust, lower the burden for testing and contribution while also aiming for sustainability and acceptance in the free network communities.

4.6. FAMILIARITY WITH SYSTEM

To facilitate the usage and maintenance, the community cloud users should be provided with an environment they are familiar with and which allows them to reuse already existing implementations and prototypes. Such environment is given by a Linux OS in combination with root permissions which has evolved to a de-facto standard environment for open source cloud computing platforms.
4.7. EASE OF USE

The majority of the users of the community cloud are not considered to have a technical background and thus are not required to understand the intricacies of the cloud infrastructure. Setting up nodes for deployment should be a simple and straightforward procedure. Similarly, managing and updating the cloud software on the nodes should be done automatically.

4.8. 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 effort from the users.

4.9. SELF-MANAGEMENT

Wireless community networks are generally unstable due to the complexity of their structure and the frequent link disruptions. Different nodes can join and leave the community cloud at any time. However, a community cloud should be able to self-manage itself and continue providing services without disruptions. One important aspect is the coordination between different cloud owners that become part of the federated community cloud.

4.10. DEVELOPER COMMUNITY

Given that community networks do not have professional software developers, a cloud platform that has a large user community and a developer community, mailing list, repository etc., is needed such that the cloud software is updated, extended and integrates new versions of dependencies.

With such cloud software, the community network cloud platform only needs to be updated with the latest stable version.

4.11. 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 meta-cloud [Satzger] 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
With multiple independent cloud operators, security becomes even more important in a community cloud. 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 [Bernsmed] that need to be addressed for ensuring users' trust in the system.

4.13. INCENTIVES FOR CONTRIBUTION

A community cloud is built 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.

4.14. SUMMARY

Given the above considerations, in order to fit into the community networks, a sustainable solution of a community cloud needs to satisfy the following: 1) must be easy to maintain (no specialist required), and 2) maintenance with least effort (takes not much time).
5. Low-level requirements

The option for enabling a community cloud in a wireless mesh network on which we focus here is to deploy a cloud management platform tailored to community networks on a super node. Figure 3 shows an overview of such a cloud management platform for community networks. The core of the 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. Namely, OpenNebula [OpenNebula] and OpenStack [OpenStack] are considered to be among the most consolidated and popular open source tools.

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

### 5.1. HARDWARE LAYER

The Hardware Layer consists of the physical infrastructure and basic virtualization support. In the context of wireless community networks the physical infrastructure might consist of ordinary nodes and wireless links provided by the mesh network. To offer virtualization capability the underlying operating system should be able to give support for virtual environments that is typically achieved via a service called hypervisor on top.

A deployed hypervisor is capable of managing a single host's network and computational resource fragmentation, isolation and exposure. Typically, such limited isolated environments are given in terms of virtual machines (VMs), also called slivers. Additional drivers should also be installed to allow further operations on VMs by a VMM’s Core Layer.
The Core Layer consists of a series of components responsible for the creation, allocation, scheduling, monitoring and management of VMs offered by the hosts' Hardware Layer. There are several components involved in the management of VMs, some of which are shown in Figure 3.

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

5.3. 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 the 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.

The cloud coordinator requires a number of modules such as gossip-based discovery and distributed self-management.

5.4. 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 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 such as resource usage tracker, contribution tracker and credits transaction manager.

5.5. 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 set up 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. Some of the modules to achieve these goals are such as distributed identity manager, support ticketing system and social contribution tracker. 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.
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. Different components require to interact with each other. For example, a user, to make a request for a VM from the community cloud, requires to connect to GUI in frontend layer and submits a request for a VM instance. The request needs to be forwarded to the cloud coordinator to check for the availability of the resources. In order to authenticate the user, cloud coordinator checks with identity manager component of social engine. Indeed, cloud coordinator checks the virtual credits database of economic engine to ensure the user has the sufficient credits to fulfil the request.
On the level of complete systems for community cloud computing [CCC], there are a few research prototypes. Skadsem et al. [Skadsem] 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 [CHome] 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 [CCloud] is an ongoing project that plans to use idle resources from 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 [Seti] and do not address the problem of how to encourage users to devote resource. They aim to use the Nimbus 9 as virtual machines manager (VMM), with Xen 10 hypervisor running on dedicated machines and VirtualBox 11 on common users’ machines.

Clouds@home 12 [Clouds] 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 13 [P2PCS] 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.

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 7 (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 [incent1]–[incent4]. None of these incentive mechanisms however target the particular situation of wireless community networks.

We notice that none of the found related works proposed and discussed clouds within wireless mesh community networks.
7. Conclusions

There is an evident need today to offer cloud services to existing community networks and thus make use of their unutilised underlying computational and storage resources. An architecture design is needed that takes into consideration the nature of such networks. In this work we offered a description of solution scenarios and proposed the high and low level requirements of a community cloud based on existing community networks semantics. The proposed design is flexible and allows federations of heterogeneous clouds with self-management capabilities. By considering node role properties there is a potential to offer sustainable bottom-up cloud operability and distributed virtualised management of resources that can also be adaptive to changes.

For the high level requirements we considered user-related concepts such as the need of incentives for contribution, security and ease of use for adaptability. In terms of low level requirements, we further proposed a layer stack where each layer addresses specific issues such as virtualisation support, discovery and self-management, resource allocation and member contributions tracking. Incentives can be offered to users to encourage contributions to improve the community network. A Support Ticketing System could be used that rewards with virtual credits which further translate into cloud service benefits. Finally, the overall approach includes off-the-shelf cloud management platforms, potentially federated with other cloud organizations, and further extends them to provide higher level services to the members of community networks.
D3.1 Requirements for a holistic network and service architecture

8. References


[LXC] LXC Linux Containers: http://lxc.sourceforge.net/


[OpenWrt] OpenWrt: https://openwrt.org/

[PlanetLab] PlanetLab, an open platform for developing, deploying, and accessing planetary-scale services: https://www.planet-lab.org/

[SFA] SFA: http://groups.geni.net/geni/wiki/SliceFedArch

[SFAWrap] SFAWrap: http://sfawrap.info/


[tinc] tinc: http://www.tinc-vpn.org/


D3.1 Requirements for a holistic network and service architecture


D3.1 Requirements for a holistic network and service architecture

Licence

The CLOMUNITY project, June 2013, CLOMUNITY-201306-D3.1:

This work is licensed under a Creative Commons Attribution-ShareAlike 4.0 Unported License.