Cloud Services in the Guifi.net Community Network

Mennan Selimi[a,*], Amin M. Khan[a], Emmanouil Dimogerontakis[a], Felix Freitag[a], Roger Pueyo Centelles[b]

[a] *Department of Computer Architecture, Universitat Politècnica de Catalunya BarcelonaTech, Spain*
[b] *Fundació Privada per a la Xarxa Oberta, Lliure i Neutral Guifi.net, Catalonia, Spain*

Abstract

Internet and communication technologies have lowered the costs to collaborate for communities, leading to new services like user-generated content and social computing and, through collaboration, collectively built infrastructures, such as community networks. Community networks are 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. Internet access is often considered the main service of community networks, but the provision of services of local interest within the network is a unique opportunity for community networks, which is currently predominantly unexplored. The consolidation of today's cloud technologies offers community networks the possibility to collectively build community clouds, building upon user-provided networks, and extending towards an ecosystem of cloud services. We propose a framework for building a collaborative distributed community cloud system that employs resources contributed by the members of the community network for provisioning infrastructure and software services. This framework is tailored to the specific social, economic, and technical characteristics of community networks and requirements for community clouds in order to be successful and sustainable. We materialise this framework in the implementation of the *Cloudy* distribution. We conduct real deployments of these clouds in the Guifi.net community network and evaluate cloud-based applications such as service discovery and distributed storage. This deployment experience supports the feasibility of community clouds and our measurements demonstrate the performance of services and applications running in these community clouds. Our results encourage the development and operation of collaborative cloud-based services using the resources of a community network. We anticipate that such services can effectively complement commercial offers and have the potential to boost innovation in application areas in which end-user involvement is required.

Keywords: cloud computing, community networks, community cloud, service discovery, distributed storage

1. Introduction

Wireless community networks are an emergent model of infrastructure that aims to satisfy a community's demand for Internet access and ICT (information, communication and technology) services [1]. Different from the traditional business-focused model applied by telecommunication operators, each user in community networks is an owner of a portion of the total infrastructure, which builds the mesh network. In the early 2000s, community networks gained momentum in response to the limited options for network connectivity in rural and urban communities. Using off-the-shelf network hardware and the open, unlicensed wireless spectrum, volunteers teamed up to invest in, create, and run wireless networks in their local communities as an open telecommunication infrastructure based on self-servicing and self-management by the users. Community networks primarily use wireless technology to interconnect nodes. However, with the commoditisation of optical fibre, some community networks have also started providing broadband services combining both technologies. Many of these bottom-up initiatives have proven to be quite successful. For example, currently the largest community network in the world, Guifi.net [2] connects more than 28,000 locations (nodes) with wireless and optical fibre. A few other examples of successful community networks are Athens Wireless Metropolitan Network (AWMN) [3] in Greece, FunkFeuer [4] in Austria and Ninux [5] in Italy.

Community networks are a successful case of resource sharing among a collective, where not only is networking hardware shared, but also time, effort, and

*Corresponding author. Email: mselimi@ac.upc.edu

knowledge are contributed by its members, which are required for maintaining the network. In practice, from the hardware perspective, resource sharing in community networks refers to the sharing of the nodes' interconnectivity and bandwidth. This sharing enables the traffic created at the user's node to be routed over the neighbouring nodes towards those of different owners, allowing community networks to successfully operate as IP networks. Despite achieving the sharing of bandwidth, community networks have not been able to widely extend this sharing to other computing resources, such as processing time or storage, which is a common practice in today's Internet through cloud computing. For instance, when we look at the user-oriented services currently consumed in Guifi.net, more than 50% are gateway proxies to provide Internet connectivity, as shown in Section 2. This implies that community members connect to the Internet and take advantage of the services available over the Internet. There are various reasons that other services have not been developed within community networks or have not gained traction among the members of community networks, but we believe that the main reason is the lack of streamlined mechanisms to exploit all the resources available within the community networks. As a result, the development of these types of services can be very challenging.

To overcome this obstacle, we propose that members of community networks should share resources based on a collaborative cloud computing model. In such a model, community network members can provide their excess capacity to others as the demand fluctuates and in return take advantage of services and applications offered by others that would be difficult or expensive to bring together by a single user due to the limited resources or the inflexibility of the environment.

The concept of a community cloud has been introduced in its generic form previously, e.g. [6, 7], as a cloud deployment model in which a cloud infrastructure is built and provisioned for 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 these. The authors present an abstract architecture together with a set of requirements that future implementations should consider. In our work, we refer to a specific kind of community cloud where computing resources are shared inside a community network, while considering the application models of cloud computing in general. We propose the community cloud as the platform that will enable cloud-based services in community networks.

The main contribution of this paper is to show the feasibility of community clouds, which we demonstrate by the development of a community cloud platform, the real deployment of community clouds in the Guifi.net, and the performance measurements of applications running on these cloud infrastructures. We begin by presenting a framework for collaborative community clouds. This framework integrates the resources contributed by community network members to the cloud. We materialise this framework by the implementation of the *Cloudy* distribution [8–10]. With the Cloudy contribution, we aspire to encourage community network users to join and participate, ultimately creating an ecosystem of services.

Cloudy is designed as a convenient method to package together different cloud services for streamlined development and delivery of value to the end users. Cloudy is based on existing open-source software and provides applications and services that add value for the users; many of these software programs are already being used to varying extents over the wider Internet. However, the challenge for Cloudy remains to analyse how well these applications behave in community networks. To this end, Cloudy is a work in progress, and we continue to integrate new services and applications. However, we have identified two services to focus our work on to start. At the system-level, we consider discovery services, such as Avahi [11] and Serf [12], since we think it is necessary for users to be able to easily discover available services and for the system to keep track of the status of the services. For the users, we focus on distributed storage service, such as Tahoe-LAFS [13], for a number of reasons, as the storage service is comparatively easy to develop, integrate, and support, and its value easily makes sense to the end users. In addition, other cloud services need storage for the back-end in order to function.

The rest of the paper is organised as follows. Section 2 presents the current state of service deployment in community networks through a study of Guifi.net as well as related research on community clouds. Section 3 presents the design and implementation of our framework for community clouds as well as the deployed services. In Section 4, we conduct experiments with the deployed community cloud and present results on the performance of our framework and services. Section 5 discusses and positions our results. Finally, Section 6 concludes and indicates future research.

2. Background

The predominant trend in many community networks is to use the available resources as a means to access external services provided elsewhere on the Internet. This might be seen as a contradiction to their spirit since traffic within the community network is freely available,

while Internet access is charged or restricted to some extent (bandwidth limitations apply or packets exit through proxies). Ubiquitous cloud services, such as private data storage and backup, instant messaging, media sharing, social networking, etc., are generally operated by well-known Internet service vendors. Community network participants are thus increasingly affected by the problems and disadvantages of this model (privacy, security, property, legislation, dependency, etc.).

In some cases, Internet cloud services have equivalent alternatives that are owned and operated at the community level; in other cases, however, there are no locally driven alternatives, yet. Possible reasons for the absence of these community-owned services can be found in the difficulty to deploy such services and the shortage or lack of individuals, organisations, or companies interested in the commercial operation of these services.

As we describe next, since community networks have become popular, there have been efforts to develop and promote different services and applications from within community networks but without significant adoption. One of the reasons identified is the technological barrier. Before providing content, users willing to share information with the community must first take care of the technical aspects, such as the deployment of a server with a set of services. For example, the key characteristic of the Guinux [14] distribution, explained below, was a set of scripts that automatised the configuration process. End users were only asked for a few parameters, such as their e-mail address and the node identifier. Shortly after the distribution was made available, the number of end users sharing resources proliferated. Thus, it became clear that lowering (or removing) the technological entry barrier encouraged users to provide more services and share their resources with the community. Nevertheless, community networks are still typically used to access external Internet services, as presented later. We believe that this is a result of the lack in number, diversity, and user-friendliness of services as well as performance disadvantages of non-commercial distributed applications compared to Internet cloud services.

In this section, we investigate the challenges and requirements involved in providing a community cloud system for the Guifi.net community network, considering the issues described previously. To achieve that, we present past efforts from within the Guifi.net community network for providing services. We also discuss other initiatives in community cloud computing portraying the state-of-the-art. This provides the context and motivates our proposal for a community cloud system in Section 3.

Services	**Catalonia**	
Network graph server	219	39.24%
DNS server	198	35.48%
NTP server	96	17.20%
Bandwidth measurement	36	6.45%
Logs server	4	0.71%
LDAP server	3	0.53%
Wake on LAN	2	0.35%
Total	**558**	

Table 1: List of network-focused Guifi.net services in Catalonia area

2.1. Current State of Service Deployment in Guifi.net

To obtain the dimension of the current situation, we analyse the list of services published (i.e., publicly announced) by the Guifi.net community network. We do so by means of the list of services available on the Guifi.net web page for the Catalonia region of Spain, the origin and most dense location of Guifi.net [15].

Tables 1 and 2 [15] indicate the network-focused and user-focused services, respectively, of Guifi.net and the proportion of each service in the services offered. We consider that the number of instances of a service implies the demand of the service inside the network. Comparing the tables, we notice that the services related to the network operation itself slightly outnumber the services intended for end-users. Considering that network management is of interest only to a fragment of the network members compared to user-focused services, which could be of interest to all users, we would expect user-focused services to be more developed. Moreover, the most frequent of all the services, whether user-focused or network-focused, are the proxy services. Specifically for the user-focused services, the percentage of Internet access services (proxies and tunnel-based) is higher than 55%, confirming that the users of Guifi.net are typically interested in accessing the Internet. We can also claim that there is a diverse set of services inside Guifi.net, even though their adoption is overshadowed by Internet access.

It is important to point out that this situation is not unique to Guifi.net. Other community networks exhibit similar situations, where the network is typically used to access the Internet, and the few services available within the community network are similar to those available in Guifi.net. For instance, Elianos et al. [16] presented similar information regarding AWMN. The authors mainly focused on user-oriented services, which are quite simi-

Services	Catalonia	
Proxy server (Internet access)	275	53.50%
Web pages	57	11.08%
VoIP / audio / video / chat / IM	48	9.33%
Data storage server	41	7.97%
Radio / TV stations	18	3.50%
P2P server	17	3.50%
Linux mirrors	15	2.91%
Webcam	12	2.33%
Tunnel-based Internet access	10	1.94%
Mail server	6	1.16%
Weather station	6	1.16%
Games server	5	0.97%
CVS repository	2	0.38%
Server virtualisation (VPS)	2	0.38%
Total	**514**	

Table 2: List of user-focused Guifi.net services in Catalonia area

lar to the Guifi.net services. The most popular services are web hosting, data storage, VoIP, and video streaming.

The services provided by Guifi.net can be categorised under cloud computing service models (though not following the traditional cloud elastic on-demand service approach): Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS). Concerning IaaS, following the global trend, the popularity of virtualisation technologies is rising in Guifi.net. Currently, almost all critical services are run on virtualised environments, frequently using Proxmox [17]. Guifi.net also provides specific hardware infrastructure and software, supporting virtual networks and tunnelling. Additionally, some efforts have been made in the past to provide the end users with tools to help them with the deployment and expansion of the community network from the software and services perspective. This was the case of Guinux, a GNU/Linux distribution for end users allowing them to deploy servers with services useful for community networking, namely Proxy, DNS, and SNMP (Simple Network Management Protocol) graphs servers. Similarly to PaaS, a diverse set of services has been deployed, such as automated node configuration, user authentication, service monitoring (servers and network), and an on-line service directory as well as network information and administration databases. Finally, taking into account the SaaS model in the context of community networks, data storage services have been sporadically deployed by enthusiastic users who wanted to share some of their content (pictures, documents, etc.) with the rest of the community [18]. In some cases, users have also enabled uploading to folders, allowing other users to upload their files for sharing with the community. Despite this, it should not be considered a data storage service for end users. Moreover, Guifi.net users have developed GuifiTV [19], a project initially conceived to harmonise the captured video formats and the content from seminars and workshops, which later included video streaming services.

The services described above are representative examples of those usually deployed in community networks. Nevertheless, both network-oriented and user-oriented services are centralised and offered by individuals. Distribution and decentralisation are concepts that are closely related to the community network philosophy; nonetheless, since centralised solutions are generally much easier to develop and deploy, in most of the cases they end up being implemented according to the classical client-server approach. As a result, basic cloud service requirements, as described in [6], are not fulfilled. Most importantly, there is no common pool of resources but instead a set of separate resources, since the same services are deployed independently and not coordinated in a common way. As a result, service coordination and resource sharing mechanisms are the first milestones towards creating cloud services for community networks, which are provided by the cloud framework we propose.

Our effort targets fostering the deployment of clouds and cloud-based services on top of community networks. Based on the experiences of current community networks, providing end users with the appropriate applications has proven to be an effective way of encouraging them to use, provide, and promote services. We exploit our experiences and the above analysis to design, implement, and evaluate the proposed community cloud framework for community networks.

2.2. Related Work

Carving our path towards community clouds in community networks, we must consider the cloud essential characteristics, as described in [6]. *Broad network access* is already offered by the community networks and *resource pooling* should be an outcome of the resource sharing described above. *Measured services* are very important in order to guarantee that the services will not disrupt the proper function of the network. *On-demand self-service* is a higher-level concept concerning the responsiveness and the transparency of the system; thus, this is an important feature but of secondary priority. Similarly, *rapid elasticity* of the resources offered is a

welcome property; yet, it should not be considered an essential one due to its complexity because of the highly distributed environment.

The idea of collaboratively built community clouds follows earlier distributed voluntary computing platforms, such as BOINC [20], Folding@home [21], PlanetLab [22], and Seattle [23], which largely rely on altruistic contributions of resources from users, functioning as research platforms. There are only a few research proposals for community cloud computing [7], and most of them do not go beyond the architecture level, whereas very few present a practical implementation.

The Cloud@Home [24] project aims to harvest resources from the community for meeting the peaks in demand, working with public, private, and hybrid clouds to form cloud federations. The Clouds@Home [25] project focuses on providing guaranteed performance and ensuring quality of service (QoS), even when using volatile Internet volunteered resources. The P2PCS [26] project has built a prototype implementation of a decentralised peer-to-peer cloud system. It uses Java JRMI technology and builds an IaaS system that provides very basic support for creating and managing virtual machines. These implementations, to our knowledge, are not actually deployed inside real community networks, considering the infrastructure diversity, and are not aiming to satisfy end-user needs.

Social cloud computing [27] is a relevant research field that takes advantage of the trust relationships between members of social networks to motivate contribution towards a cloud storage service. Users trade their excess capacity to earn virtual currency and credits that they can utilise later, and consumers submit feedback about the providers after each transaction, which is used to maintain the reputation of each user. Social clouds have been deployed in the CometCloud framework by federating resources from multiple cloud providers [28]. The social compute cloud [29], implemented as an extension of the Seattle platform [23], enables the sharing of infrastructure resources between friends connected through social networks and explores bidirectional preference-based resource allocation.

Among federated cloud infrastructures, Gall et al. [30] have explored how an InterCloud architecture [31] can be adapted to community clouds. Further, Esposito et al. [32] presented a flexible federated Cloud architecture based on a scalable 'publish and subscribe' middleware for dynamic and transparent interconnection between different providers. Moreover, Zhao et al. [33] explored efficient and fair resource sharing among the participants in community-based cloud systems. In addition, Jang et al. [34] implemented personal clouds that federate local, nearby, and remote cloud resources to enhance the services available on mobile devices.

Service discovery is an important component of service coordination. Specifically for wireless mesh networks, the work of Dittrich et. al. [35] evaluated the responsiveness of service discovery. They ran their experiments in the wireless DES testbed at Freie Universitat Berlin. They showed how service discovery responsiveness is affected by the position and number of requesters and providers as well as the load in the network. In wireless mesh network settings, the work of Wirtz [36] proposed DHT-based Localised Service Discovery(DLSD), an hierarchy of localised DHT address spaces that enable localised provision and discovery of services and data. Another work [37] proposed a stochastic model family to evaluate the user-perceived responsiveness of service discovery and the probability to find providers within a deadline, even in the presence of faults.

From the review of related work, we find that none of the above research has implemented an operational prototype for community networks. In the community cloud system that we present, we consider the socio-technical factors that characterise community networks; therefore, the framework we propose is tailored to the specific context of deploying cloud-based services in community networks.

3. Community Cloud Framework and Services

A community network is owned by the community and the nodes are managed independently by their owners. As a result, the network devices or nodes in a community network vary widely in their capacity, function and capability. 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, which is usually intended to be stable with permanent connectivity. Others act as client nodes and are only connected to the access point (AP) of an SN as demonstrated in Figure 1. A topological analysis of the Guifi.net community network [38] indicates that from approximately 17,000 analysed nodes of Guifi.net, 7% are SNs while the others are client nodes.

From the node types illustrated in Figure 1, it can be seen that the hardware for computation and storage is already predominantly available in community networks, consisting of servers (e.g., home gateways, laptops etc.) attached to the networking nodes. No cloud software or 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.

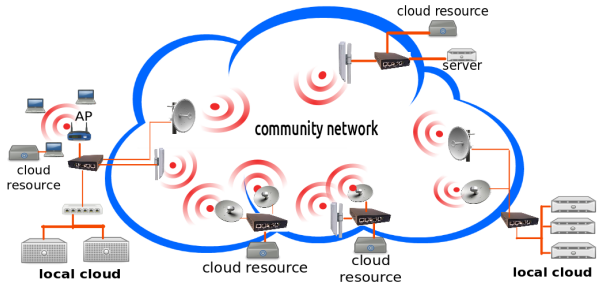

Figure 1: Nodes in a community network with cloud resources

Our vision is that community wireless routers will tend to have cloud resources attached, in order to build the infrastructure for a community cloud formed by several cloud resources distributed among several nodes. We note that the contributed cloud resources could be principally located at client nodes, where the actual users of the community network exist. Based on the structure, topology, and socio-technical characteristics of community networks, we identify a community cloud consisting of multiple local clouds, where an SN is responsible for the management of a set of attached nodes contributing cloud resources. These multiple local clouds are then connected to form a federated community cloud, where SNs connect physically to other SNs through wireless links and logically in an overlay network to other SNs that manage local clouds.

3.1. Requirements for a Community Cloud System

A community cloud is a cloud infrastructure which is run and managed independently by various community network members. The community cloud bridges different aspects in 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). For the community cloud management system, we are targeting the Guifi.net community network, and we consider the following requirements to be a foundation and guideline for its design. We believe that, if addressed, among other challenges, these requirements can largely provide a cloud system that is deployed and adopted successfully by the community.

- Security
  Privacy and security are of great importance in community clouds, as users share their resources and data between them. 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. For instance, the data and applications running on different cloud systems should be protected from unauthorised access.

- Self-Management
  The highly distributed nature, typically wireless environment, and heterogeneity of community networks require that a community cloud platform be self-managed on the cloud and node level in order to continue providing services without disruption when nodes go offline. Self-management should also help in the coordination between various cloud owners that become part of a federated community cloud. The most relevant aspects for the desired framework are self-configuration and self-healing.

- Utility
  The bottom-up nature of community networks drives its evolution and development. As a result, the community cloud should provide applications that are valuable for the specific user community. Nevertheless, there exist applications necessary for the majority of community networks, as we have presented in Section 2.1, such as Internet connectivity, file sharing, video streaming, and VoIP services. Successful applications will increase the usage, strengthening the value of the community cloud, thus motivating its maintenance and upgrade.

- Ease of Use
  Most of the users of the community cloud will not be proficient in cloud technologies; therefore 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 this cloud model will be adopted.

- Incentives for Contribution
  The community cloud builds upon the 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 community clouds to be sustainable, incentive mechanisms are needed to encourage users to actively contribute towards the system.

- Support for Heterogeneity
  As previously explained, community networks are very heterogeneous in diverse levels. Thus, the hardware and software used by members in a community cloud can have varying characteristics, and the cloud system should handle this seamlessly.

- Standard Application Programming Interfaces (API)
  The cloud system facilitates the ability of application programmers to transparently design their applications for the underlying heterogeneous cloud infrastructure. The API should provide the appearance of a middleware that obviates the need to customise the applications to each specific cloud architecture. This is essential for community clouds when these result from the combination 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.
- QoS and SLA Guarantees
  The community cloud system requires mechanisms for ensuring the quality of service (QoS) and enforcing service level agreements (SLA).

Based on these requirements, in terms of an institutional policy, we design a community cloud framework, which leads to an implementation of the community cloud system that will be responsible for joining and consuming cloud services almost automatically with little user intervention.

3.2. Framework for Distributed Community Cloud System

We foresee realising the community cloud by deploying a community cloud platform tailored to the specific infrastructure and context of community networks. A standard cloud platform is usually a centralised platform designed to perform resource management. There are quite a few well known cloud platforms for managing public and private clouds, like OpenStack [39] and OpenNebula [40] among others. In our effort nevertheless, we focus on providing a framework that would allow users to share resources and access collaboratively-built services in a distributed manner. For instance, a community cloud platform would require incentive mechanisms inspired by the social nature of community networks integrated into resource regulation components to encourage contribution from the members of the community network [41].

3.2.1. Layers of Community Cloud System

We propose a framework that can serve as the core of a community cloud system. Our community cloud framework is a distributed bottom-up resource sharing and collaborative services platform. This is achieved by adopting a layered architecture, as shown in Figure 2.

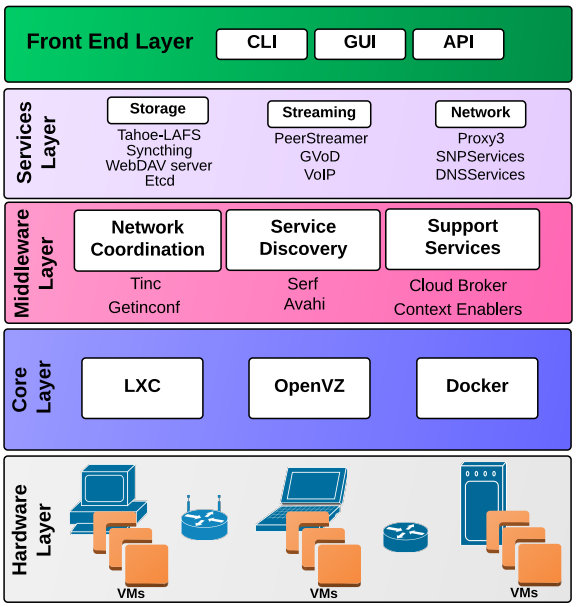

Figure 2: Framework for community cloud management system

1. The **Hardware layer** provides the physical infrastructure needed to run the cloud services and applications. The hardware in the community networks customarily consists of SNs, client nodes, routers and the communication infrastructure, along with any computation, storage and other resources attached to the nodes.

2. The **Core layer** is responsible for managing the hardware as virtualised resources. It consists of components, such as a manager for the hosts and the network as well as a controller, scheduler, monitor, and data storage for virtual instances. Many popular open-source software programs can be integrated to provide virtualisation, for instance LXC (Linux Containers) [42], OpenVZ [43], and Docker [44], etc.

3. The **Middleware layer** amalgamates the resources from multiple local community clouds, providing an integrated and consistent view of the cloud system to the cloud services. This requires a network coordination component to identify and manage different local clouds and a service discovery component to keep track of the services provided by the various clouds. Other support services can include cloud coordinator and services broker components for assisting in combining resources from multiple cloud providers, and social and economic

context enablers [45] that take advantage of the social and economic characteristics of the community networks to encourage participation from community network members and to ensure sustainability of the community cloud model.

4. The **Services layer** integrates useful services and applications providing utilities for the community network members to encourage their participation. Common services include storage, video streaming, video on demand, voice over IP (VoIP), and network applications.

5. The **Front-end layer** provides the interface to interact with the infrastructure of the community cloud, including command line interfaces (CLI), graphical user interfaces (GUI), API, and any other tools for assisting in the development of cloud services and applications.

3.3. Cloudy: Community Cloud-in-a-Box

As we discussed earlier in Section 2, we believe that the failure of services gaining traction in community networks was largely due to the difficulty of implementing the services and for the end-users to consume these services. To overcome these issues, we choose to implement the proposed framework as a GNU/Linux distribution, code named Cloudy, to provide a convenient mechanism for developing and consuming cloud services in community networks, with the hope that Cloudy can encourage the adoption and uptake of cloud services among the users.

Cloudy [8] is a distribution based on a Debian GNU/Linux aimed at end users to foster the transition and adoption of the community cloud environment. Cloudy is the implemented prototype of our community cloud framework described in Section 3.2. The current prototype of Cloudy implements the modules/layers shown in Figure 2. A Cloudy instance can be run directly on a bare metal machine or on a virtual machine. Independent of the hardware that Cloudy runs on, connectivity to other Cloudy instances is needed in order to fully exploit the potential of Cloudy.

3.4. Cloudy Services

Cloudy comprises a number of services, designed to help build cloud-based services in community networks. Cloudy's main components can be considered a layered stack with services residing both inside the kernel and higher up at the user-level. All of the software included in the Cloudy platform is open-source. All service accesses are assisted and managed through the main panel of the Cloudy GUI. The following three groups classify Cloudy services.

3.4.1. Infrastructure Services

Virtualisation is the main enabling technology for cloud computing. As such, providing community network users the resources to deploy virtual machines with a few clicks is a very convenient way to bring the cloud closer to their premises. This allows the non-experienced user to focus on the services and applications themselves rather than on learning how to cope with the underlying infrastructure.

OpenVZ [43] is an operating system-level virtualisation technology for Linux based on containers. OpenVZ allows creating multiple secure, isolated operating system instances called containers (commonly known as VPSs) on a single physical machine enabling better server utilisation and ensuring that applications do not conflict with each other. Each container performs and executes exactly like a stand-alone server (which can have root access, users, IP addressing, memory, files, etc.) and can be started and stopped independently from the others and from the host machine. OpenVZ is the preferred solution for providing virtual machines in Cloudy with low to mid-end hardware as only a negligible portion (1-2%) of the CPU resources is spent on virtualisation. The Cloudy distribution includes a script that downloads and installs all the required OpenVZ packages in one click and Cloudy instances can be run on the virtual machines created using the OpenVZ Web Panel.

Other virtualisation methods used in Cloudy are LXC and Docker. This approach adds special support for IaaS, as the cloud nodes are able to create multiple virtual machine instances for other purposes in addition to the ones dedicated to Cloudy. The infrastructure services of Cloudy enable resource sharing inside the community network.

3.4.2. Service Discovery and Network Coordination Services

Cloudy provides custom decentralised services for network coordination and service discovery. Network coordination ensures visibility between the nodes that participate in the cloud. Service discovery is a crucial building block in Cloudy for enabling distributed services to be orchestrated to provide platform and application services. Service discovery is based on the network coordination component.

For service discovery, Cloudy includes a customised version of Avahi [11] to provide decentralised service discovery at Layer 2, which is needed to discover other services that will be used to provide higher-level services.

The multicast-based design does not allow the Avahi service to reach beyond the local link, which is the case in community networks, where services are spread over different nodes that belong to different broadcast domains. While in this environment, it would not be possible for Avahi packets to be directly exchanged from one node to another; this problem is solved by the network coordination component.

For the network coordination component, we adopt TincVPN [46], a virtual private network (VPN) daemon that uses tunnelling and encryption to create a secure private Layer 2 network between hosts of different domains. This Layer 2 connectivity is needed between nodes, since they may reside on different administrative domains and even be located behind firewalls. The TincVPN is automatically installed and configured on every Cloudy node, ready to be activated. After its activation, a VPN daemon is started in order to reach other Cloudy instances via the established Layer 2 network; thus, Avahi can communicate transparently with other nodes. To easily install and configure a system with TincVPN, a tool called Getinconf [47] has been developed, which is integrated into Cloudy.

Cloudy also includes Serf [12], a lightweight tool to announce and discover services in community networks. Serf is a decentralised solution for cluster membership, failure detection, and orchestration. It relies on an efficient and lightweight gossip protocol to communicate with other nodes that periodically exchange messages between each other. This protocol is, in practice, a fast and efficient method to share small pieces of information. An additional by-product of having this service distributed all over the community cloud is that it allows the evaluation of the quality of the point-to-point connections between different Cloudy instances. This way, Cloudy users can decide which service provider to choose based on network metrics, such as round trip time (RTT), number of hops, or packet loss. The combination of Avahi, TincVPN, Getinconf, and Serf in Cloudy facilitates the coordination of the resources and the services in the community cloud.

3.4.3. User Services

Platform as a Service (PaaS). Providing attractive platform services to community members, such as a distributed file system, highly available key-value store, file synchronisation, video streaming, video-on-demand, VoIP, network address translation (NAT) traversal support, and many more, is of high importance.

One of the promising services for storage is Tahoe-LAFS [48]. Tahoe-LAFS is a free, open, and secure cloud storage system. Tahoe-LAFS allows community users to share their storage with other members. A Tahoe-LAFS cluster consists of a set of storage nodes, client nodes, and a single coordinator node called the introducer. The storage nodes connect to the introducer and announce their presence, and the client nodes connect to the introducer to obtain the list of all connected storage nodes [49]. The configuration of Tahoe-LAFS and the process of deploying a whole storage grid are assisted by the Avahi and Serf service discovery tools using the web interface of Cloudy, where the user only needs to introduce some basic information. The Tahoe-LAFS service can also be used to provide higher-level file sharing applications.

Etcd [50], a highly available key value store for shared configuration and service discovery, and Syncthing [51], an open-source file synchronisation client/server application, are already included in the Cloudy distribution.

Software as a Service (SaaS). Cloudy allows the user services to be present inside the community network and to be easily deployed and managed via the Cloudy interface. Users can deploy their preferred services and share them with others. One of these multimedia services included in Cloudy is PeerStreamer [52], an open source live streaming platform. PeerStreamer includes a streaming engine for the efficient distribution of media streams, a source application for the creation of channels, and player applications to visualise the streams. Streaming is assisted by Cloudy by supporting the user in publishing a video stream or connecting to a peer (assisted by Serf or Avahi). Services that enable users to find and watch video content on-demand at any time, such as Gvod [53], a decentralised search service, such as Sweep [54], and a distributed key-value store, such as CaracalDB [55] are additional services that are part of the Cloudy.

4. Experiments

In this section, we explain our work on deploying and evaluating the community cloud. In order to have a realistic community network setting, which includes geographically distributed nodes, we have used the Community-Lab [56] testbed nodes for setting up our community cloud infrastructure. Community-Lab is a distributed infrastructure developed by the Community Networks Testbed for the Future Internet (CONFINE) project [1], where researchers can deploy experimental services on several nodes deployed within merged community networks. The Community-Lab testbed is currently deployed on the nodes from Guifi.net and the AWMN community networks. This allows us to run our experiments on nodes from both the community networks, which has the added advantage that we can test

how Cloudy performs in a combined community network environment, as well as how Cloudy services perform over large geographical distances.

In the Community-Lab, Guifi.net and the AWMN community networks are connected on the IP layer though the Federated E-infrastructure Dedicated to European Researchers (FEDERICA) [57], enabling the federation of both networks. Most Community-Lab nodes are built with a Jetway device that is equipped with an Intel Atom N2600 CPU, 4GB of RAM and 120GB SSD. Nodes in the Community-Lab testbed run a custom firmware (based on OpenWRT [58]) provided by CONFINE, which allows running several virtual machine instances on one node simultaneously implemented as LXC. We deploy the Cloudy distribution in these virtual machines on the nodes in Community-Lab.

We exhibit results from our experiments related to service discovery and distributed storage service. We choose to experiment with service discovery, based on Avahi-TincVPN and Serf, since it is a crucial building block of Cloudy that enables distributed services to be orchestrated in order to provide platform and application services. All the services inside Cloudy use these two service discovery protocols to publish and discover services. Our goal is not to compare Avahi-TincVPN and Serf, as both of them are available in Cloudy and can be used by users for different scenarios. One of them is lightweight and fast (Serf), the other is not scalable and is suitable and preferable for environments with a smaller number of nodes (Avahi-TincVPN). In order to test distributed storage behaviour in Cloudy, we evaluate the Tahoe-LAFS distributed storage. Tahoe-LAFS has features that are very important for the community network environment, such as like data encryption at the client side, coded transmission and data dispersion among a set of storage nodes. This approach of Tahoe-LAFS results in high availability (e.g., even if some of the storage nodes are down or taken over by an attacker, the entire file system continues to function correctly, while preserving privacy and security).

4.1. Service Discovery Experiment

Cloud service discovery is essential for allowing cloud usage and user participation. Service discovery involves service providers publishing services and clients being able to search and locate service instances. Since community cloud nodes are distributed all over the network and administrated by their owners, a mechanism is needed that allows the cloud users to discover services offered by other community cloud nodes and announce their own services. We experiment with the Avahi-TincVPN and Serf search service of Cloudy that offers service announcement and discovery to community users.

Table 3: Nodes, their location and RTT from the client node

Number of nodes	Community Network	Location	RTT
13	Guifi.net (UPC)	Barcelona	1–7 ms
7	Guifi.net	Catalonia	10–20 ms
5	AWMN	Athens	90–100 ms

4.1.1. Experiment setup

For our service discovery experiment we use 25 nodes spread between two community networks (Table 3). We use 20 nodes from the Guifi.net community network, where 13 of the nodes are located in the city of Barcelona (UPC) and seven of them are located in the Catalonia region of Spain. From AWMN we use five nodes, which are located in Athens, Greece.

Figure 3 shows the throughput of three categories of nodes in our cluster. Network measurements have been obtained connecting by SSH to each node and measuring the average aggregated throughput from the client nodes. For some scenarios we use more than one client node. The clients are located in the Guifi.net community network. Connections to the nodes have been done every half hour (12 hours per day), during the entire month of January 2015 where 720 samples are obtained for each category of nodes. All obtained samples are plotted in the graph. The average throughput obtained for the Guifi.net nodes in UPC is 10.5 Mbps, Guifi.net nodes is 4.8 Mbps, and AWMN nodes is 1.9 Mbps.

The objective of the experiments is to understand the responsiveness of the discovery mechanism. We consider responsiveness to be the probability of successful operation within deadlines, which, when applied to our case, refers to successful service discovery within the given time limits. Furthermore, we attempt to understand how the clients perceived responsiveness changes when they are located in different parts (zones) of the Guifi.net community network.

We run the discovery requests from three client nodes that are searching and locating service instances. All other nodes acted as service providers responding to discovery requests. All service providers are spread between two community networks. Discovery times are measured on the clients directly before the request was sent and directly after responses were received to measure user-perceived responsiveness. No nodes joined or left the network; therefore, no configuration on the net-

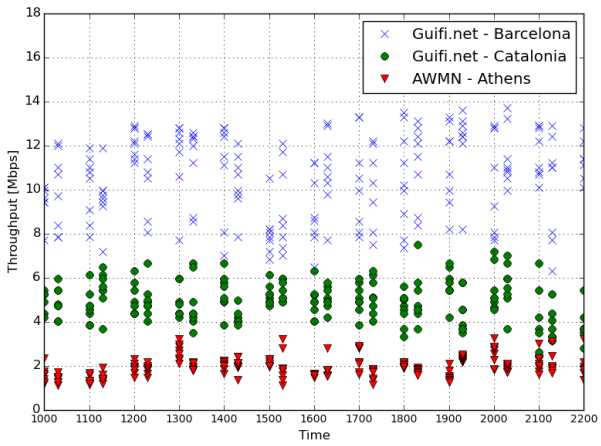

Figure 3: Throughput of the nodes

work layers occurred during measurements which would interfere with the discovery operation. We consider the discovery successful when all instances have been discovered. Discoveries were aborted and considered failed if no responses arrived until an experiment ran with a deadline of 25 seconds in the Community-Lab testbed. This value was chosen because in Zeroconf [59], the time between retries doubles after each retry to reduce the network load. Therefore, for example, after 20 seconds, we have reached five discovery requests, and the next one would be sent after 41 seconds. Depending on the scenarios that we consider, each service discovery experiment is comprised of several runs (normally between 15 to 20) and is averaged over all the successful runs. Each run consists of 20 repetitions.

In Avahi, when publishing and discovering, no entries are cached per interface; thus, no caching is used. After service discovery, a client should have enough information to contact a service instance. Hence, discovery in our case means resolving the IP address and port for every service instance. During the experiments we use different Cloudy services to publish and discover as summarised in Table 4.

4.1.2. Our Scenarios

In order to judge the applicability of decentralised discovery mechanisms in community networks, three scenarios are chosen that reflect common use cases of service discovery.

Scenario 1: Single service discovery. Our first goal is to measure the responsiveness of single service discovery. In this scenario, the service network consists of one client and one provider. The client is allowed to wait up to ten seconds for a positive response. This is a common scenario for service discovery and can be considered the baseline. Only one answer needs to be received and there is enough time to wait for it. In this case, the client discovers a Tahoe-LAFS distributed storage service and contacts the service. For this scenario, a service provider from the Guifi.net community network is considered. Both Avahi-TincVPN and Serf are used for this scenario.

Table 4: Services used for the experiments

Service	**Description**
PeerStreamer	Live-video streaming service
Tahoe-LAFS	Decentralised cloud storage service
Syncthing	File synchronisation service
Serf	Cluster membership and discovery service
OWP	Container-based virtualisation service
Proxy3	Guifi.net proxy service
SNP Service	Guifi.net network graph service
DNS Service	Guifi.net DNS service

Scenario 2: Timely service discovery of the same service type. Service networks are populated with multiple instances of the same service type. The clients need to discover as many instances as possible and will then choose one that optimally fits their requirements. The faster discovery is better. In this scenario, we have one service client and 25 service providers (from Guifi.net and the AWMN community network). The discovery is successful if all provided service instances of the same type have been discovered. We measure how responsiveness increases with time. The faster we reach a high value, the better. In this scenario, the providers publish one or more PeerStreamer live-video streaming services. The client waits 15 seconds to receive responses. In this scenario only the service discovery based on Serf is used. The total number of services published by providers is 40.

Scenario 3: Client perceived responsiveness. In this scenario, we have three service clients and 20 service providers that publish five popular services of Cloudy such as PeerStreamer, Tahoe-LAFS, Syncthing, DNSService, and OpenVZ. The total number of the five Cloudy services published is 23 (seven PeerStreamer services, three Tahoe-LAFS services, six Syncthing services, three DNSServices and four OpenVZ services). The clients are located in different parts of Guifi.net, and they need to discover 23 instances of different service types. Considering the dynamic environment of the Guifi.net community network, the discovery is success-

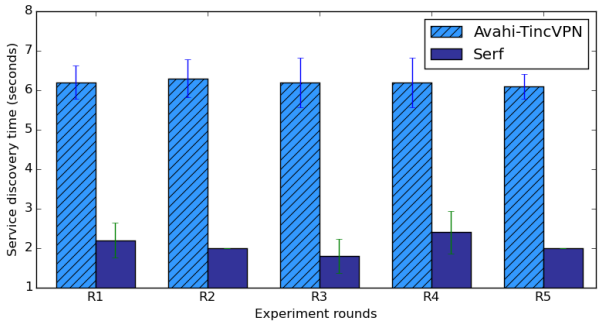

Figure 4: Single service discovery time (Scenario 1)

ful if all clients discover the same number of services (23) published by service providers. For this scenario, the service discovery based on Serf is used. The clients are allowed to wait 20 seconds.

4.1.3. Experimental Results

In this section, the results for the three scenarios described above are presented.

Scenario 1: Single service discovery. The discovery of a single service instance within ten seconds proved to be reasonably responsive. This experiment is comprised of 15 runs, where each run has 20 repetitions. In Figure 4, the standard deviation error bars per round are plotted on the mean values obtained. The values are obtained using Avahi-TincVPN and Serf. Due to the efficient and lightweight gossip protocol that Serf uses, it decreases the discovery time for 3x, reaching an average of two seconds for a single service discovery compared to the Avahi-TincVPN combination that reaches six seconds.

Scenario 2: Timely service discovery of the same service type. Figure 5 illustrates that the discovery of services increases rapidly with time. The standard deviation error bars are plotted on the mean values. In the first six seconds the client discovers 75% of the published services, which is equal to 30 PeerStreamer video-streaming services. The last 25% of the services are discovered from seconds six to ten. These ten services are from the AWMN community network. The eventually consistent gossip model of Serf with no centralised servers allows the client to discover in a very fast and extremely efficient way all the nodes for a PeerStreamer service based on the tags their agent is running. However, the structure and diameter of the community network graph (topology), fluctuations in the network due to load, and faults can increase the discovery time [38].

Figure 6 shows a partial screenshot of the Cloudy web interface, depicting the service discovery section. The five PeerStreamer video-streaming services discovered

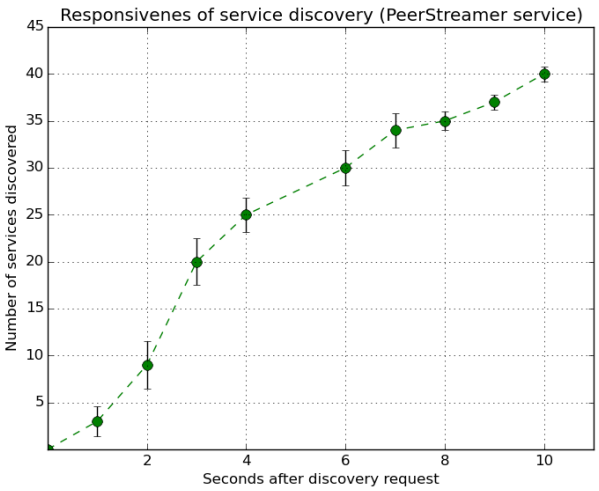

Figure 5: Responsiveness of service discovery (same service type) (Scenario 2)

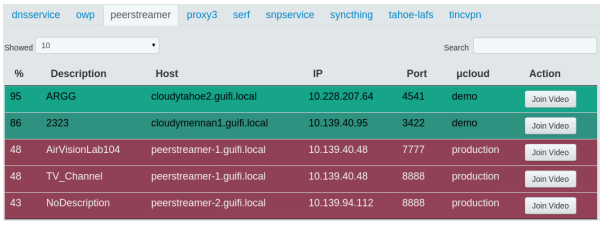

Figure 6: Partial screenshot of Cloudy service discovery section (Scenario 2)

are shown. The user will choose one that optimally fits the user's requirements. The services are ranked according to Cloudy's QoS-aware service selection algorithm, where colour represents service quality: darker colour indicates poorer service quality.

Scenario 3: Client perceived responsiveness. Figure 7 demonstrates the number of services discovered by three clients. The standard deviation error bars are plotted on the mean values. As shown, the three clients perceive a different number of services. Only Client 2 discovers all services (23). Client 1 is missing one PeerStreamer service and one DNSService. Client 3 is missing just one PeerStreamer service. Missing services can be subject to the high diversity of the quality of wireless links, the availability of nodes, and the location of client nodes. Heterogeneous low-resource hardware, slow wireless links, and packet loss between nodes also can impact the service performance [60].

4.2. Distributed Storage Experiment

Allowing users in a community network to share and use the storage of other users in a reliable, secure, and

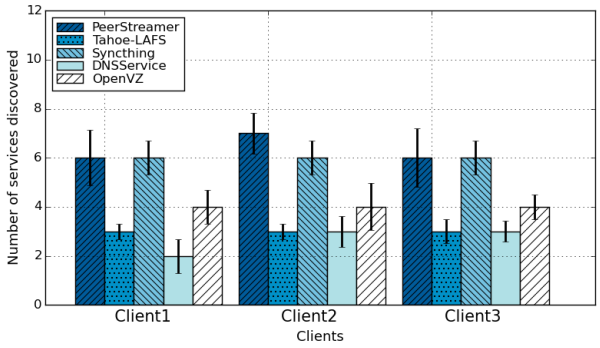

Figure 7: Number of Cloudy services discovered by different clients (Scenario 3)

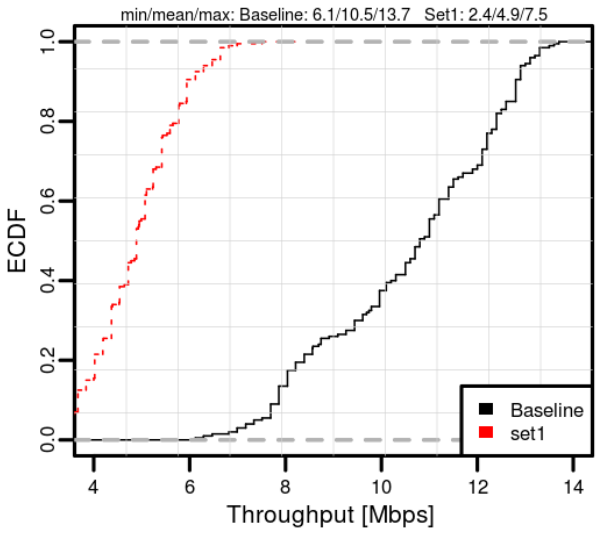

Figure 8: ECDF of the average throughput for two clients

privacy-preserving way, is of a great importance. For this reason, we use Tahoe-LAFS as a main storage service in Cloudy. Understanding the performance of Tahoe-LAFS from an experimental scenario that represents real use case situations is highly important because it informs the end users regarding the application performance they will receive. Such performance results are needed to pave the way for bringing applications such as Tahoe-LAFS as well as other applications into community networks. In our previous work [49], we conducted a performance evaluation of Tahoe-LAFS in a federated community network environment, where we considered smaller workloads. The approach for the performance assessment of Tahoe-LAFS that we consider here is to set the experimental conditions as seen from the end user, to experiment in production community networks, and focus on metrics that are of interest for end users (considering bigger workloads). For the distributed storage experiment, we consider only the Guifi.net community network.

4.2.1. Experiment Setup

All tests were conducted using the IOzone cloud storage benchmark [61]. IOzone is a filesystem benchmark tool, which generates the cloud storage workload and measures various file operations. The benchmark tests file input/output (I/O) performance of many important storage-benchmarking operations, such as read, write, re-read, re-write, random read/write, etc. We run all 13 IOzone tests and vary the file size from 64KB to 128MB and record length of 128 KB. An *-a* flag allows us to run all 13 tests. We add the *-b* flag to write the test output in binary format to a spreadsheet. We use a FUSE (Filesystem in Userspace) kernel module in combination with SSHFS (SSH Filesystem), an SFTP client that allows filesystem access via FUSE, to mount a Tahoe-LAFS directory to the local disk of the client. Tahoe's SFTP frontend includes several workarounds and extensions to make it function correctly with SSHFS. When mounting with SSHFS, we disable the cache and use direct I/O and synchronous writes and reads, using the parameters *-o cache=no, big_writes, direct_io, and sshfs_sync*. We observe that the *-o big_writes* option to SSHFS improves write performance without affecting the read operations [62]. Results presented in this paper with regard to performance are measured in MB/s and are referred to as operation speed. Tests with concurrent reading and writing were not conducted.

To better understand the impact that the network imposes on a community network environment, we established a Tahoe-LAFS cluster of 30 nodes geographically distributed in the Guifi.net community network and connected to the outdoor routers [60]. Connections to the nodes from the clients have been done hourly (ten samples obtained per day), during the whole month of February 2015, where 300 samples are obtained for each client. Figure 8 shows the empirical cumulative distribution function (ECDF) of the average throughput for the two clients. On the top of the figure, the minimum/mean/maximum throughput values are shown.

Two sets of tests were conducted. One is when the writes/reads are initiated from a client that has the best connectivity in the network, such as best RTT to other nodes and the best throughput, and this is our baseline case; the other is when they are initiated from a client node, which is the farthest node in the network (in terms of number of hops, RTT, and throughput to other nodes), and this is referred as a set 1 case in the graphs.

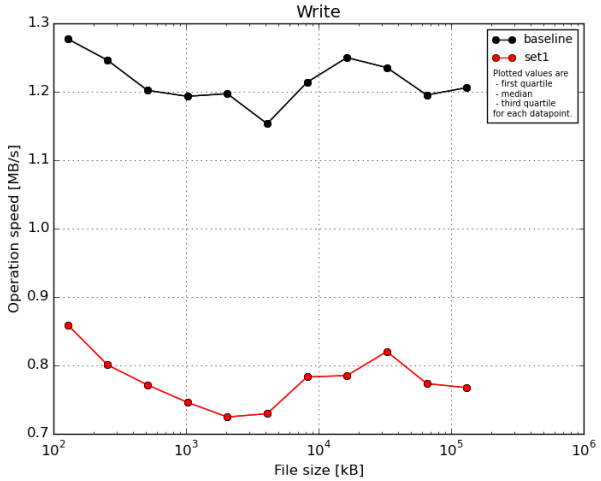

Figure 9: Performance of write operation in community network.

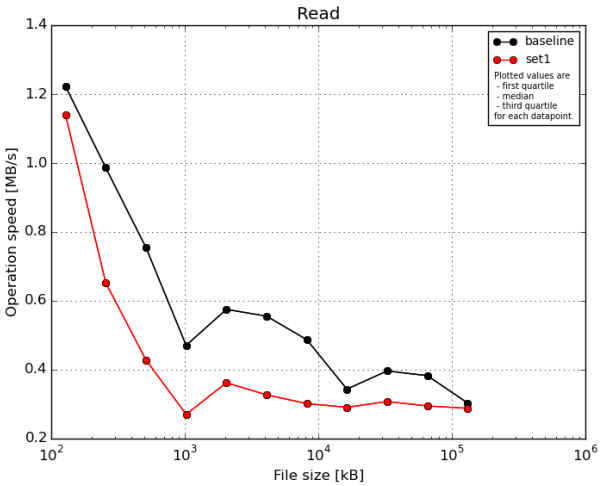

Figure 10: Performance of read operation in community network.

4.2.2. Experimental Results

Figures 9 and 10 show the best and worst client write/read performance. Figure 11 depicts the summary of all tests performed with the IOzone benchmark. Median, first and third quartile values of read and write operations are plotted in Figure 11. A few observations are noted below.

- In terms of network connectivity, both clients in the community network perform differently. This is related to the fact that the two clients are not connected in the same way to other nodes. The client in the baseline is better connected and is much closer in terms of RTT to the other nodes than the client in set 1.

- In terms of write performance, the baseline client performs better. Write performance for the baseline is higher and more stable than the read performance. As the file size increases, the write performance of the baseline client slightly decreases (minimum throughput achieved is 1.15 MB/s when writing a 4 MB file). The higher throughput is achieved (1.28 MB/s) when writing a small file (128 KB file), as shown in Figure 9. It is interesting to note that when writing smaller files, Tahoe-LAFS performs better, and this can be attributed to the fact that the default stripe size of Tahoe-LAFS is well optimised for writing small objects (the stripe size determines the granularity at which data is being encrypted and erasure coded). The same thing happens with the set 1 client, where the maximum write throughput achieved is 0.86 MB/s when writing a 128 KB file, and the minimum write throughput is 0.72 MB/s when writing a 2 MB file. Furthermore, write performance is affected by another factor; when writing new objects, Tahoe-LAFS generates a new public/private key, which is a computationally expensive operation.

- Read operations are accomplished by submitting the request to all storage nodes simultaneously; hence, the relevant peers are found with one round-trip to every node. The second round-trip occurs after choosing the peers from which to read the file. The intention of the second round-trip is to select which peers to read from, after the initial negotiation phase, based on certain heuristics. When reading from the storage nodes, the performance of both clients drops significantly as the file size increases as shown in Figure 10. This is because when reading a file of 128 MB, a client must contact more Tahoe-LAFS storage peers in order to complete the shares of the file. In addition, reading the file system meta-object (i.e., the mutable directory objects) every time an object is accessed results in overhead, thus influencing the results.

- Figure 11 shows the summary of all tests performed with the IOzone benchmark. The benchmark tested file I/O performance for the 13 operations as shown in Figure 11. As shown, the baseline client performs better than the set 1 client, reaching an average operation speed of 0.74 MB/s for all 13 tests performed.

To summarise, Tahoe-LAFS is a relevant application for community networks, since it offers privacy and security, as it encrypts data already on the client side, and it offers fault-tolerance regarding storage node failures due to erasure coding (replication factors). A general important result from the experiments is that Tahoe-LAFS performed correctly in uploading and retrieving all the different file sizes under the challenging conditions of the community network, which make Tahoe-LAFS a promising application to consider for preserving privacy, and secure and fault-tolerant storage in the dynamic environment of community networks Furthermore, the process of deploying a whole storage grid is assisted by Cloudy, which allows a user easily to join or offer a storage grid.

5. Discussion

In this section we bring together the results achieved by the presented community cloud and elaborate on our current position and opportunities.

5.1. Fit of the Community Cloud to the Socio-technical Conditions of Community Networks

The community cloud was developed considering the specific requirements given in the context of community networks. The features of the presented cloud system in particular satisfy the requirements of openness, freedom of usage, and technological neutrality. The community cloud software given by the Cloudy distribution is completely open source and integrates primarily software components that have a consolidated developer community. The installation of Cloudy has been achieved on different types of hardware, ranging from low-cost SBC to high-end computers. It was demonstrated that Cloudy could be installed on bare metal but also in virtual machines provided by any cloud management system. Technological neutrality is given by not relying on any specific component or library that might make the Cloudy system susceptible to vendor lock-in.

Particular features that were achieved include user-friendliness and self-management. Regarding user-friendliness, many functionalities of Cloudy can be managed through a web GUI, which enables less technically skilled users to be able to install and operate a Cloudy-based community cloud node. Providing tools suitable for technically unskilled users facilitates the growth of the community cloud ecosystem. To this end, the Cloudy web-based management platform integrates all the installation and configuration steps for the cloud services enabled in the Cloudy distribution. A simple web interface is available to the end user as an easy way to configure, administer, and monitor the cloud services running in the node.

Self-management capabilities were integrated in terms of a cloud search service, which dynamically publishes and updates the presence of nodes and services available in the cloud. Many services of Cloudy are based on decentralised mechanisms, reducing the need for centralised components, which may be difficult to permanently maintain in the community network context.

5.2. Sustainability of the Community Cloud Ecosystem

The sustainability of the community cloud system depends on usage and user contribution. Several elements of the presented community cloud support sustainability, given that from the beginning the community cloud was developed to fit to the community network context.

We foresee, however, once the users contribute to the community cloud, that additional mechanisms might need to be considered to better steer contributions and usage, for consolidating sustainability. Explicit incentives might need to be created, to reward contributions. Excessive free riding, if not controlled by an additional mechanism, will consume resources without contribution and might make the community cloud unusable.

To further address sustainability, the community cloud will need to be extended with measurement tools to assess usage and contributions. This improved transparency regarding the system availability to the community might increase confidence and encourage participation. The integration of the community cloud system to complement commercial offers should also be sought as a method that leads to sustainability. Successful and innovative applications which could arise from synergies between both cloud models may ultimately make community clouds a core component of future services.

5.3. Community Clouds beyond Community Networks

Community clouds, understood as clouds with specific features to satisfy the needs of particular communities, have already been developed for several commercial sectors. Such community clouds bridge different aspects of the gap between the public cloud, the general purpose cloud, and the private cloud. These are needed where general purpose cloud solutions provided by the public cloud do not optimally fit to the specific needs of the various user communities, since, for instance, certain security concerns or performance requirements on clouds are only insufficiently addressed by such generic cloud solutions. A community cloud offers features that are tailored to the needs of a specific community. The opportunity within a community cloud lies in being able to

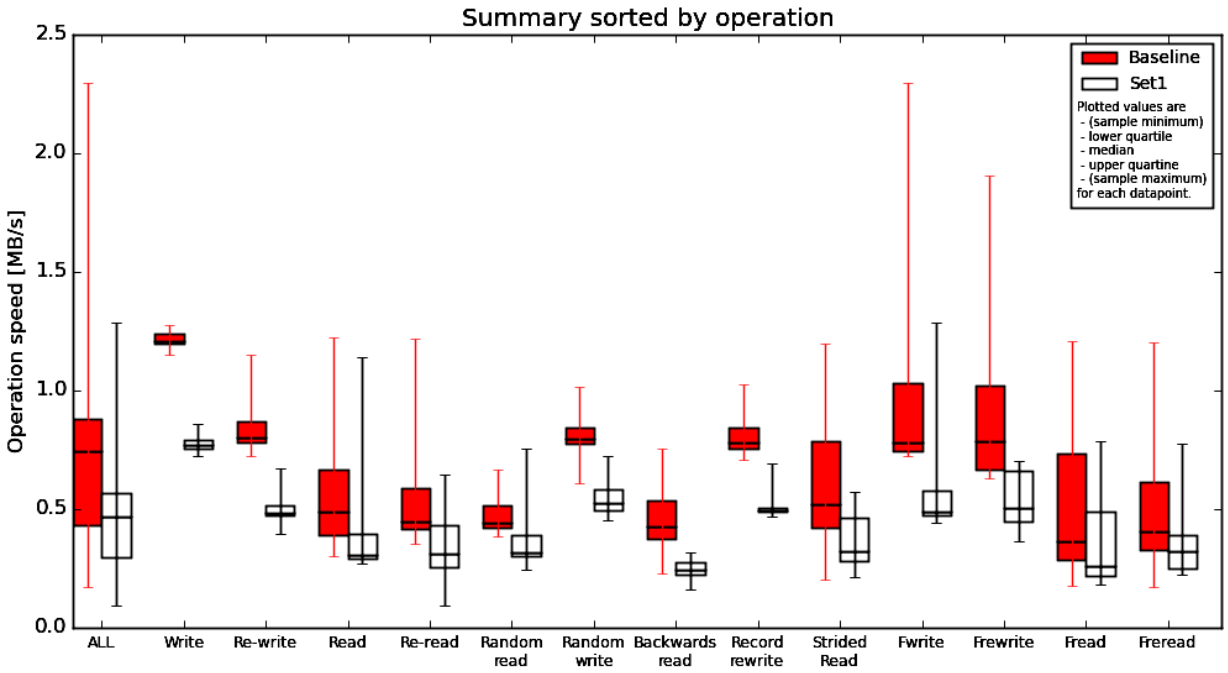

Figure 11: Summary of all storage benchmark operations for different tests in the Guifi.net community network.

offer optimised cloud solutions tailored to specific user communities.

The presented community cloud from this perspective is a community cloud that addresses the needs of citizens. While the presented cloud was deployed using Cloudy within a specific community network (i.e., Guifi.net), Cloudy could principally be deployed in other community networks as well, considering that similar socio-technical conditions are found in many community networks. The presented community cloud system, therefore, could be deployed in other community networks as well.

It is also interesting to discuss whether such a community cloud for citizens could function on top of other communication networks (e.g., those provided by commercial operators), which are not managed by a community. Since many different communities exist, we expect that, from a social perspective, such a community cloud is possible. From a technical perspective, the presented community cloud currently benefits from the availability of network segments given to the users, a situation that occurs in community networks, but is difficult to find in a commercial IPv4 network, where users often do not even have static IP addresses. However, such community clouds operated over commercial networks seem possible, when (through IPv6) end users can have routable network segments for offering services.

6. Conclusions and Outlook

Community networks would greatly benefit from the additional value of applications and services deployed inside the network through community clouds. However, such clouds in community networks have not yet been demonstrated in related studies as operational systems, missing proof of feasibility, which would enable exploring further innovations.

In this paper, a deployed community cloud system operating in the Guifi.net community network was demonstrated, supporting the feasibility of such a system. In addition, performance measurements were conducted, which demonstrate the usability of cloud-based services for end users. The community cloud system was developed from a framework design, which integrates the hardware and software contributions to build the cloud system based on requirements of the specific socio-technical context of community networks. This framework was materialised by the implementation of the Cloudy distribution. Using Cloudy and the hardware infrastructure available in the community network, the community cloud was deployed in Guifi.net, demonstrating the community cloud feasibility.

Cloudy is designed as a convenient way to package together diverse cloud services for streamlined development and delivery of value to the end users. Cloudy is based on existing open-source software, and we analysed

how well these software behave in community networks. We identified two services to focus on, considering their importance for the system operation and the end users. We experimented with service discovery based on Avahi and Serf, and the results exhibited their proper functioning. These system-level services allowed users to discover services offered by other community cloud nodes as well as announce their own services. We also evaluated distributed storage service Tahoe-LAFS for end users. Tahoe-LAFS performed correctly in uploading and retrieving all the files under the challenging conditions of the community network and appeared to be a promising application to consider for preserving privacy, and for secure and fault-tolerant storage in community clouds.

Performance measurement of services and applications provided by this cloud were conducted in order to assess their usability by end users. Our results demonstrated the operation of the community cloud in the community network and the usability of services.

Given the deployment and availability of an operational cloud system, our next step is to engage end users from the community networks to participate in community clouds. This participation can happen as users of the services or as contributors to the cloud. In order to catalyse user participation, the deployed cloud infrastructure continues to be operational, providing a basic set of nodes with stable services, until a sufficiently large number of community contributed cloud nodes has been reached. Users can easily contribute to the cloud by installing the Cloudy distribution on the contributed hardware, available for download and ready to be installed. The services of Cloudy will integrate the new cloud resource into the community cloud.

After the uptake of this cloud by the community network members, we expect that the potential of community clouds can be implemented, by complementing existing public cloud services with collaborative user-shaped applications for local communities and by federating with public cloud services, to enable innovative applications that benefit from end user participation.

Acknowledgements

This work is supported by the European Community Framework Programme 7 FIRE Initiative projects Community Networks Testbed for the Future Internet (CONFINE), FP7-288535 and CLOMMUNITY, FP7-317879. Support is also provided by the Universitat Politècnica de Catalunya BarcelonaTECH and the Spanish Government under contract TIN2013-47245-C2-1-R.

References

[1] B. Braem, R. Baig Viñas, A. L. Kaplan, A. Neumann, I. Vilata i Balaguer, B. Tatum, M. Matson, C. Blondia, C. Barz, H. Rogge, F. Freitag, L. Navarro, J. Bonicioli, S. Papathanasiou, P. Escrich, A case for research with and on community networks, ACM SIGCOMM Computer Communication Review 43 (3) (2013) 68–73.

[2] Xarxa de Telecomunicacions Mancomunada, Oberta, Lliure i Neutral, http://guifi.net/ (2014).

[3] Athens Wireless Metropolitan Network (AWMN) (2014). URL `http://www.awmn.net/`

[4] FunkFeuer (2014). URL `http://funkfeuer.at/`

[5] Ninux.org Wireless Network Community (2014). URL `http://ninux.org`

[6] P. Mell, T. Grance, The NIST Definition of Cloud Computing, NIST Special Publication 800 (145).

[7] A. Marinos, G. Briscoe, Community Cloud Computing, in: 1st International Conference on Cloud Computing (CloudCom 2009), Vol. 5931 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, Beijing, China, 2009, pp. 472–484. ,

[8] Cloudy: A community networking cloud in a box, http://cloudy.community/ (2015).

[9] J. Jiménez, R. Baig, P. Escrich, A. M. Khan, F. Freitag, L. Navarro, E. Pietrosemoli, M. Zennaro, A. H. Payberah, V. Vlassov, Supporting cloud deployment in the Guifi.net community network, in: 5th Global Information Infrastructure and Networking Symposium (GIIS'13), IEEE, Trento, Italy, 2013, pp. 1–3.

[10] M. Selimi, F. Freitag, R. P. Centelles, A. Moll, L. Veiga, Trobador: Service discovery for distributed community network micro-clouds, in: 29th IEEE International Conference on Advanced Information Networking and Applications (AINA'15), 2015, pp. 642–649.

[11] Avahi Service Discovery Tool, http://avahi.org/ (2015).

[12] Serf, https://www.serfdom.io/ (2015).

[13] Tahoe-LAFS: The Least-Authority File Store, https://tahoe-lafs.org/trac/tahoe-lafs, accessed: 2015-02-01.

[14] Guinux, https://guifi.net/en/node/29320, accessed: 2015-02-01.

[15] Guifi.net: Services of Catalunya (by zone), https://guifi.net/en/node/2413/view/services.

[16] F. Elianos, G. Plakia, P. Frangoudis, G. Polyzos, Structure and evolution of a large-scale wireless community network, in: IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks & Workshops, (WoWMoM'09), 2009, pp. 1–6.

[17] Proxmox: Server-Virtualization with KVM and Containers, https://www.proxmox.com/, accessed: 2015-02-01.

[18] M. Selimi, F. Freitag, R. Pueyo Centelles, A. Moll, Distributed storage and service discovery for heterogeneous community network clouds, in: 7th IEEE/ACM International Conference on Utility and Cloud Computing (UCC'14), 2014, pp. 204–212.

[19] Guifitv, http://project.Guifi.net/projects/Guifitv, accessed: 2015-02-01.

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

[21] A. L. Beberg, D. L. Ensign, G. Jayachandran, S. Khaliq, V. S. Pande, Folding@home: Lessons From Eight Years of Volunteer Distributed Computing, in: 8th IEEE International Workshop on High Performance Computational Biology (HiCOMB'09), within IPDPS, IEEE, Rome, Italy, 2009, pp. 1–8.

[22] B. Chun, D. Culler, T. Roscoe, A. Bavier, L. Peterson, M. Wawrzoniak, M. Bowman, PlanetLab: An Overlay Testbed for Broad-

Coverage Services, ACM SIGCOMM Computer Communication Review 33 (3) (2003) 3–12.
[23] J. Cappos, I. Beschastnikh, A. Krishnamurthy, T. Anderson, Seattle: a platform for educational cloud computing, in: 40th ACM Technical Symposium on Computer Science Education (SIGCSE'09), ACM, Chattanooga, USA, 2009, pp. 111–115.
[24] S. Distefano, A. Puliafito, Cloud@Home: Toward a Volunteer Cloud, IT Professional 14 (1) (2012) 27–31.
[25] S. Yi, E. Jeannot, D. Kondo, D. Anderson, Towards real-time, volunteer distributed computing, in: 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid'11), 2011, pp. 154–163.
[26] O. Babaoglu, M. Marzolla, M. Tamburini, Design and implementation of a p2p cloud system, in: Proceedings of the 27th Annual ACM Symposium on Applied Computing, SAC '12, ACM, New York, NY, USA, 2012, pp. 412–417.
[27] K. Chard, K. Bubendorfer, S. Caton, O. F. Rana, Social Cloud Computing: A Vision for Socially Motivated Resource Sharing, IEEE Transactions on Services Computing 5 (4) (2012) 551–563.
[28] M. Punceva, I. Rodero, M. Parashar, O. F. Rana, I. Petri, Incentivising resource sharing in social clouds, Concurrency and Computation: Practice and Experience.
[29] S. Caton, C. Haas, K. Chard, K. Bubendorfer, O. F. Rana, A Social Compute Cloud: Allocating and Sharing Infrastructure Resources via Social Networks, IEEE Transactions on Services Computing 7 (3) (2014) 359–372.
[30] M. Gall, A. Schneider, N. Fallenbeck, An Architecture for Community Clouds Using Concepts of the Intercloud, in: 27th International Conference on Advanced Information Networking and Applications (AINA'13), IEEE, Barcelona, Spain, 2013, pp. 74–81.
[31] R. Buyya, R. Ranjan, R. N. Calheiros, InterCloud: Utility-Oriented Federation of Cloud Computing Environments for Scaling of Application Services, Algorithms and Architectures for Parallel Processing 6081 (2010) 20–31. ,
[32] C. Esposito, M. Ficco, F. Palmieri, A. Castiglione, Interconnecting Federated Clouds by Using Publish-Subscribe Service, Cluster Computing 16 (4) (2013) 887–903.
[33] H. Zhao, X. Liu, X. Li, Towards efficient and fair resource trading in community-based cloud computing, Journal of Parallel and Distributed Computing 74 (11) (2014) 3087–3097.
[34] M. Jang, K. Schwan, K. Bhardwaj, A. Gavrilovska, A. Avasthi, Personal clouds: Sharing and integrating networked resources to enhance end user experiences, in: 33rd Annual IEEE International Conference on Computer Communications (INFOCOM'14), IEEE, Toronto, Canada, 2014, pp. 2220–2228.
[35] A. Dittrich, D. Herrera, P. Coto, M. Malek, Responsiveness of service discovery in wireless mesh networks, in: IEEE 20th Pacific Rim International Symposium on Dependable Computing (PRDC), 2014, pp. 234–243.
[36] H. Wirtz, T. Heer, M. Serror, K. Wehrle, Dht-based localized service discovery in wireless mesh networks, in: MASS, 2012, pp. 19–28.
[37] A. Dittrich, B. Lichtblau, R. Rezende, M. Malek, Modeling responsiveness of decentralized service discovery in wireless mesh networks, in: K. Fischbach, U. Krieger (Eds.), Measurement, Modelling, and Evaluation of Computing Systems and Dependability and Fault Tolerance, Vol. 8376 of Lecture Notes in Computer Science, Springer International Publishing, 2014, pp. 88–102.
[38] D. Vega, L. Cerda-Alabern, L. Navarro, 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, IEEE, Barcelona, Spain, 2012, pp. 612–619.
[39] OpenStack: Open Source Cloud Computing Software (2014). URL http://www.openstack.org/
[40] OpenNebula: Open Source Data Center Virtualization (2014). URL http://opennebula.org/
[41] A. M. Khan, U. C. Buyuksahin, F. Freitag, Incentive-based Resource Assignment and Regulation for Collaborative Cloud Services in Community Networks, Journal of Computer and System Sciences (2014).
[42] LXC: Linux Containers, https://linuxcontainers.org/ (2015).
[43] OpenVZ Linux Containers, http://openvz.org/ (2015).
[44] Docker Containers, https://www.docker.com/ (2015).
[45] A. M. Khan, M. Selimi, F. Freitag, Towards Distributed Architecture for Collaborative Cloud Services in Community Networks, in: 6th International Conference on Intelligent Networking and Collaborative Systems (INCoS'14), IEEE, Salerno, Italy, 2014.
[46] TincVPN, http://tinc-vpn.org/ (2015).
[47] Getinconf tool, https://github.com/Clommunity/getinconf.
[48] Z. Wilcox-O'Hearn, B. Warner, Tahoe: The least-authority filesystem, in: Proceedings of the 4th ACM International Workshop on Storage Security and Survivability, StorageSS '08, ACM, New York, NY, USA, 2008, pp. 21–26.
[49] M. Selimi, F. Freitag, Tahoe-lafs distributed storage service in community network clouds, in: 4th IEEE International Conference on Big Data and Cloud Computing (BDCloud'14), 2014, pp. 17–24.
[50] Etcd key-value store, https://github.com/coreos/etcd (2015).
[51] Syncthing, http://syncthing.net/ (2015).
[52] PeerStreamer: P2P Media Streaming, http://peerstreamer.org/ (2015).
[53] G. Berthou, J. Dowling, P2p vod using the self-organizing gradient overlay network, in: Proceedings of the 2nd International Workshop on Self-organizing Architectures, SOAR '10, ACM, New York, NY, USA, 2010, pp. 29–34.
[54] Sweep, http://wiki.clommunity-project.eu/pilots:sweep (2015).
[55] CaracalDB, https://github.com/CaracalDB/CaracalDB (2015).
[56] Community-Lab: Community Networks Testbed by the CONFINE Project, http://community-lab.net/ (2014).
[57] FEDERICA: Federated E-infrastructure Dedicated to European Researchers Innovating in Computing network Architectures, http://www.fp7-federica.eu/ (2015).
[58] OpenWRT, https://openwrt.org/ (2015).
[59] Zero Configuration Networking (Zeroconf), http://www.zeroconf.org/ (2015).
[60] L. Cerdà-Alabern, A. Neumann, P. Escrich, Experimental evaluation of a wireless community mesh network, in: Proceedings of the 16th ACM International Conference on Modeling, Analysis, Simulation of Wireless and Mobile Systems, MSWiM '13, ACM, New York, NY, USA, 2013, pp. 23–30.
[61] IOzone: a filesystem benchmark tool, http://www.iozone.org/, accessed: 2015-02-01.
[62] A. Rajgarhia, A. Gehani, Performance and extension of user space file systems, in: Proceedings of the 2010 ACM Symposium on Applied Computing, SAC '10, ACM, New York, NY, USA, 2010, pp. 206–213.