Leveraging Low-power Devices for Cloud Services in Community Networks

Nuno Apolónia, Roshan Sedar, Felix Freitag, Leandro Navarro
Universitat Politècnica de Catalunya, BarcelonaTECH
Barcelona, Spain
{apolonia, rsedar, felix, leandro}@ac.upc.edu

***Abstract*—Community networks are IP networks constantly being improved that evolve into large-scale computing platforms. This has resulted from the effort to adapt the cloud computing model towards services that can operate and utilize the resources inside the community network. The network and its infrastructure are contributed by individuals, companies, organizations and are maintained by the community itself. Community cloud devices are often low computing resource devices, such as home gateways, with limited capabilities. Currently, these devices are configured to run community services only. This has become a drawback for further adoption because of contributor's difficulty to also use the donated cloud device for private purposes. We apply container-based virtualization for the problem of resource sharing in low-capacity devices in order to create a multi-purpose execution environment in a single device. Thus, a single device can be configured to deliver to the user and the community a multi-purpose environment, such as personal and public, isolated from one another, while preserving the community cloud services. Our comparative analysis with the current infrastructure in community networks gives evidence that the capability of the devices to run concurrent services is maintained.**

Keywords: virtualization, community networks, cloud services, computing constrained-devices

I. Introduction

Community networks are large-scale, self-organised and decentralised communication infrastructures built and operated by the community itself. They are open, free and neutral IP networks. As of now, hundreds of community networks are operated across the world, which are geographically distributed in different parts of the world without relying on any specific social or economic reasons. The larger networks have from 500 to 28,000 nodes, such as Guifi.net[1], FunkFeuer[2], AWMN[3], Freifunk[4]. The infrastructure is contributed by individuals, companies and organizations in a joint effort.

Resource sharing within the community networks refer in practice to the sharing of network bandwidth from each device. This enables traffic from devices to be routed through others to its destination. The sharing of services, such as video streaming, storage, VoIP, which through cloud computing have become common practice in the Internet, hardly exists in community networks. The community cloud model could suit to accommodate services and/or resource sharing among community members.

The CONFINE project[5] was initiated to advance the understanding of the community network model, to explore the ways of improving the users quality of experience in a shared platform and to measure the sustainability of the network. For this purpose, the CONFINE project has developed an infrastructure called Community-Lab [1], which provides an environment to perform community network related experiments. The physical devices in the Community-Lab are mini-PCs and are low-power devices. This complements the energy-efficient hosting of services and/or sharing resources among other community members.

Currently, the Community-Lab infrastructure has a limited number of physical devices, therefore the scalability for experimentation is limited. Besides, all these devices are configured to perform only community cloud services and this has become a drawback for the device owners. The result is that the owners of the devices have been restricted from accessing resources for their private purposes. We address this limitation by creating an environment in a single device, which can benefit both the community and the device owner. Consequently, this produces a multi-purpose environment in a single device, such as one environment for the device owner and other environments to be shared with the community network.

Our main contributions include:

- The creation of container-based resource virtualization on top of low-power devices, enabling a multi-purpose environment isolated from each other;
- Users can share a portion of the device resources, preserving the owners multi-service allocated virtual space in the devices;
- Evaluation of the performance of devices when running services on the virtual environments, and comparison with the current settings of the Community-Lab devices.

In order to validate the proposed approach, we create a small scale physical Community-Lab infrastructure using several low-power devices together as compute and storage devices and, separately, deploy the Community-Lab controller in a virtual machine inside a desktop PC. In this experimental system we deploy two applications, Tahoe-LAFS [2], [3] and PeerStreamer [4], [5], as community cloud services and measure the performance of these applications in the environment given by our approach.

The rest of the paper is organised as follows. Section II explains the Community cloud and relates to the CONFINE project, including the formation of Community-Lab testbed with its functionality and the technologies that are being deployed. Section III describes the system architecture and the deployment approaches. Section IV refers to the experimental setup used. The evaluation and results are presented in section

[1] http://guifi.net
[2] http://www.funkfeuer.at
[3] http://www.awmn.gr
[4] http://freifunk.net/
[5] http://confine-project.eu

V. The related work is described in section VI. Conclusion and future work is explained in section VII.

II. Community Clouds

Community clouds are formed by a collaborative effort to create a computing platform where the infrastructure is shared between a number of organizations in order to provide a platform for common computing concerns. One particular objective is to provide a scalable computing platform to conduct community network related research. In this section, we explain the system overview of the CONFINE Community-Lab infrastructure.

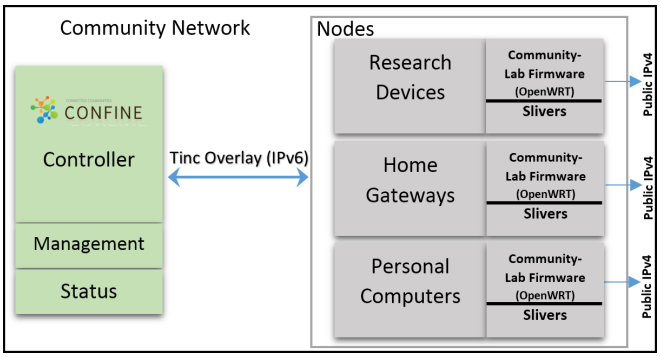

Fig. 1. System Overview of the Community-Lab Testbed

A. CONFINE Project: Community Networks Testbed for the Future Internet

The CONFINE project goal is to augment the capabilities of community networks by providing a platform for the existing community network in which users can use services with ease [6], as if they were deploying in any commodity cloud platform. Members of the CONFINE community network testbed are privileged to get a set of IP addresses (IPv4) in order for them to be able to run multiple services, as they may require. This enables members to run multiple services on top of the existing network, while sharing the resources with the community.

Community-Lab, shown in Fig. 1, is an infrastructure that provides a set of tools allowing researchers to easily deploy, run, monitor and experiment community cloud services, protocols and applications in a real community IP network (Guifi.net, FunkFeuer, AWMN and Freifunk) instead of simulated environments.

The platform is monitored by a single entity named Community-Lab controller, which allows users to lease the resources from the network, and deploy their experiments on the selected nodes. Particularly, users can choose geographically distributed computing resources through the controller and are able to customize the deployment according to their specific requirements. In addition, users are allowed to choose the appropriate configurations for the computing resources, i.e., to have public IPv4 addresses, which can be used to communicate within the community network.

Each Community-Lab node can contain several slivers, shown in Fig. 1, which are grouped at a higher level in slices. As such, a slice is defined as a set of resources spread across several physical devices in the testbed which allows users to run experiments over it. A sliver is defined as the partition of the resources (or virtual machine) of a community node assigned to a specific slice.

The purpose of the controller is to manage and control the testbed through simple operations such as managing users, nodes, slices and slivers. This controller provides an aggregation point where members can register their devices as Community-Lab nodes. In the web interface researchers can choose geographically dispersed nodes to create slices for their experiments. The nodes retrieve the given information to deploy local slivers acting as containers in the devices. Whenever users make a request to deploy a new sliver the controller creates a Linux container on the node by allocating the resources required to run the new sliver. Therefore each sliver runs on the node isolated from one another.

Linux containers guarantee isolation in terms of security and resources however the host kernel system is shared between all containers. In this way, users can deploy many slivers in a single node to run many services concurrently.

To be part of the Community-Lab infrastructure the devices require to operate a specific operating system, based on OpenWrt[6] configured to provide automatically an open network connection with the Community-Lab controller, becoming part of the testbed for the experiments. These devices serve as the infrastructure layer of the Community-Lab and most are low-resource devices which can be affordable to have at the edges of these types of networks.

B. CLOUDY: Community Cloud Distribution

Cloudy Distribution[7] (Cloudy OS) has been created under the CLOMMUNITY project[8] to provide community networks an easy way to manage and deploy cloud infrastructures and interfaces for service discovery and deployment. The result is that any user can enjoy the benefits of cloud services which are freely available in the community without relying on any specific cloud infrastructure. The Cloudy OS is a free and open source software and is a customized version of Debian Linux. By default, it comes with an installation of tinc[9] VPN daemon which creates a secured private overlay network between hosts on the Internet. With the help of tinc VPN, networked nodes can communicate securely with each other.

In addition, Cloudy OS uses Avahi[10] (or Serf[11] in latest versions), which is a zero-configuration networking implementation, to publish and discover the services in the community. For the simplicity of service discovery, the Cloudy OS provides an interface that fetches the services in the overlay and lists the services in order for the users to easily connect them.

C. Virtualization Systems

Most of the Community network devices that are used in the Community-Lab infrastructure are low-power devices such as Home gateways, set-up boxes, research devices. However these devices are capable of running multiple community cloud services simultaneously. For instance, the Cloudy OS comes with a few pre-installed services such as Tahoe-LAFS [12] distributed file storage system and PeerStreamer [13]: P2P video

[6] http://openwrt.org
[7] http://cloudy.community
[8] http://clommunity-project.eu
[9] http://www.tinc-vpn.org
[10] http://www.avahi.org
[11] https://www.serfdom.io
[12] http://tahoe-lafs.org
[13] http://peerstreamer.org

streaming framework. As we discussed previously, these devices are configured to serve only community cloud services. We can say that, as an entry-point, virtualization can give us the means to create multi-purpose environments in a single device. Therefore, we can study two main virtualization techniques and measure the system behaviour of each case in terms of performance and complexity. Users can choose one of these mechanisms considering the complexity of the platform configurations, system performance and the hardware support that the devices have.

One type of virtualization system is called Virtual machines (or machine emulation) such as QEMU which is an open source machine emulator, used to run virtual machines on top of an operating system such as Linux. It is also capable of direct virtualization when using the KVM (Kernel-based Virtual Machine) kernel module in Linux and having hardware compatible with virtualization technology. Otherwise, it can only emulate machines, and thus the virtual machines created cannot directly access some of the hardware which can provide a better guest performance.

Another virtualization technology, also available with most Linux kernels, is called Linux Containers (LXC), and it is comparable to other virtualization technologies. However it may lack some of the security and isolation methods that other virtualization technologies have, such as OpenVZ.[14] Also, it can be more lightweight since it uses the already in place features of the Linux kernels that adopted this type of virtualization. It separates the user context for each container and maintains a shared link to the host kernel in order to run multiple systems in an OS-Level virtualization method.

III. System Architectures

The architecture for our proposed system includes two approaches for virtualization. In the first approach we used a virtual machine (QEMU) in order to separate the context in which a service can be deployed. In the second approach we used a LXC based approach in which the virtualization is lightweight in order to have a better performance while running services.

A. Services Deployment

The proposed approaches, leveraging virtualization resources, are enhanced with different contexts where owners can share (shared context) only part of the devices resources while also accessing their own (private) context, where they can run their own independent services, as demonstrated in Fig. 2.

B. QEMU deployment approach

Our initial approach includes a deployment of virtual machines with the use of QEMU, installed on top of the Cloudy OS. This allows us to concurrently run services while still isolating the device to be used by its owner. In a physical device we install Cloudy OS where we can execute several instances of QEMU creating virtual environments in order to run multiple services. As an example, Fig. 3 shows the deployment of this scenario using two Community-Lab nodes and its slivers as independent services on top of the Cloudy OS. This is done in order to gain control over the device to enable multiple services running from different user contexts while maintaining the owners' ability to execute his own services.

[14]http://openvz.org

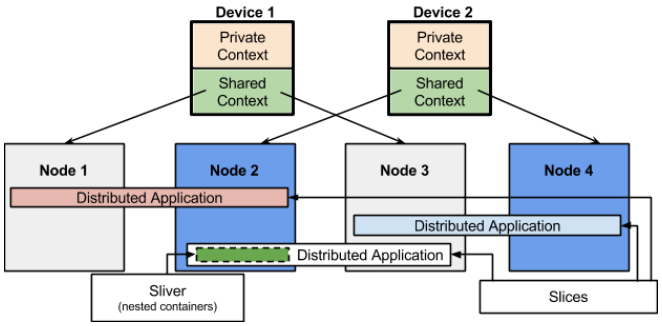

Fig. 2. Example of two device deployment. Devices are divided in two contexts for owner and shared to the CN. In shared context several slivers run as nested containers belonging to each slice deployed.

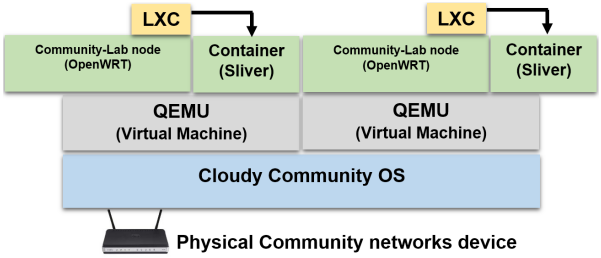

Fig. 3. Example of QEMU deployment approach. Two services running (serving as Community-Lab nodes) on virtual machines. LXC runs from within the Community-Lab node to create slivers.

The performance of QEMU is greatly increased when running it with kernel integration (KVM) or virtualization support from the hardware. This guarantees that each virtual machine has higher performance and some of the physical resources can be directly utilized by the virtual machines. However in low-power devices such option may not be available, instead QEMU has to emulate the whole process of the virtual machine which hinders the performance of the virtual machine.

Under this scenario within certain conditions (such as KVM enabled or virtualization support) we can achieve a separation of services and utilize a physical device to be shared between the owner and the community network. This type of deployment is a quick and easy way of fostering multiple services in a single device. Furthermore, this approach can be fairly enough when services need higher isolation from the host system by enhancing its security or when the service performance is not affected.

C. Linux Containers deployment approach

Our proposed approach includes a deployment of Linux containers (LXC). This allows running concurrent services with a low overhead of the virtual resources and processes. LXC creates different user contexts in the host machine to deploy separate systems, but shares the same kernel (OS-Level virtualization)

In this approach we consider the isolation of the services while still guaranteeing a higher performance for services since there is no emulation involved. The virtualization layer considered is in the OS-Level, where the host kernel is shared between containers and the host system. In Fig. 4 we show an example of the deployment of this approach using the Cloudy OS as the host system and deploying a Community-Lab node in a LXC container. It is worth mentioning that the Community-Lab nodes deploy their own slivers as LXC

containers and with our approach this does not change. This is provided by tuning the configurations of LXC to deploy nested containers, while adding the AppArmor [15] security policies for the security concerns.

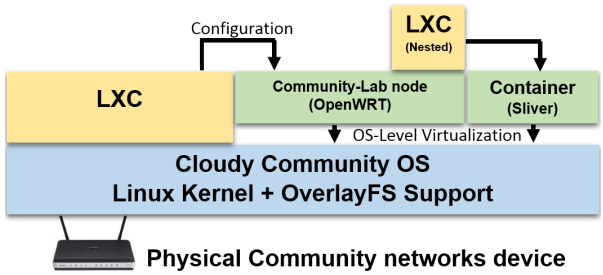

Fig. 4. Example of LXC deployment approach. One service running (serving as Community-Lab node) on a container virtualization occurring in OS-level. Nested LXC runs from Community-Lab node only configuration is needed to run nested containers.

With this approach, we can use the device as a shared environment between the owner and the community cloud environment maintaining the isolation from each user context. The security issues that arise from such usage are respectfully handled by LXC or the Linux security policies. Therefore the containers have access to the host kernel which may prove to be a minor isolation for some services, requiring more isolation should be handled with the virtual machines approach.

Under this scenario we can achieve a lighter isolation and a concurrent access to the physical devices. This makes the services running with higher performance than with the QEMU approach.

It is relevant to mention that in this approach the system running in the containers only have access to the host kernel as such the host kernel requires to be compatible (or with the required kernel modules loaded) in order to correctly run the system inside the container, i.e. the Community-Lab nodes require that the overlayFS file system should be natively, enabling it to deploy the requested slivers on the devices.

A main feature of our proposed deployment scenario is the introduction of a virtual environment in which services are able to run with different contexts and maintaining an isolated part of the device to be used by the owner. As a result when using the Community-Lab node as a service each node registered in the Community-Lab controller can deploy its own slivers in the physical device without interfering with other services or the owner usage. This enables sharing of resources for a community cloud environment and maintaining the owners exclusive access to the device.

IV. Experimental Setup

For our experimental setup, we used four physical research devices with different configurations. These devices are built with Intel powered Atom N2600 CPU processors. Two of them have 2 GB of RAM, 60 GB storage and 2 GB of RAM, 120 GB storage, and the other device has 4 GB of RAM with 500 GB of storage disk and linked to the community network (Guifi.net). A desktop computer setup with the Proxmox system[16] and built with an Intel(R) Core(TM) i7-3770 CPU @ 3.40GHz (8 cores), with 1TB of storage disk and 16 GB of RAM was used to deploy a local Community-Lab controller in a virtual machine environment. It was necessary to use such deployment in order to not affect the performance of the current Community-Lab infrastructure, which is in a production state.

For our experiments, we setup a replica of the Community-Lab testbed architecture, such as, one local controller which controls the overall system and a set of computing nodes (Community-Lab nodes) that can be used to deploy slivers from the controller. In this case, the local controller can be deployed either in a container or in a separate virtual machine instance. However, this depends on the users' requirements and the resources which are available.

The reason for the layered virtualization in the physical devices is that we intend to augment the current services of the Cloudy OS such that the users can have a service that deploys Community-Lab nodes in an automatic manner. This allows taking advantage of the virtualization environment in order for users to share their resources through the Community-Lab platform and extending the number of nodes present in the Community-Lab. Furthermore we can examine the applicability and measure the efficiency to deploy community cloud services in order to understand the feasibility for sharing devices in community networks while maintaining the owners' space in the device.

Challenges: One of the main challenges when creating our experimental environment is to set up the Community-Lab controller in a local environment using a virtual machine, and using Cloudy OS as the base or host operating system. The process of deployment of the Community-Lab controller in a local environment can be a challenging task for the common community user. The CONFINE Project wiki[17], however, contains tutorials on how a user can manage to install the controller with ease. It is also noted that some of the steps are dependent on the operating system used, and that it can break the communication overlay if not correctly configured (i.e. Tinc misconfiguration).

Another challenge faced is the fact that the current Community-Lab node system requires specific Linux kernel modules. These must be enabled in the host system, and without it, the slivers may not work as expected and therefore not function correctly when deploying/running the slivers.

V. Evaluation

In the evaluation of the proposed approach (LXC deployment approach) we used physical research devices connected to Guifi.net. These devices are low-power, have less resources compared to desktop PCs, are mostly the same devices used within the Community-Lab infrastructure and are generally similar to the shared devices from community networks. These devices are deployed with scenario LXC, previously described.

A. Experiments

Our experiments were performed in order to evaluate the proposed deployment, summarized in Table I, by utilizing different services with different purposes. In the first evaluation scenario we used Tahoe-LAFS distributed storage, and a storage benchmark application to evaluate the impact of the proposed deployment on the physical devices. In the second evaluation scenario we used PeerStreamer, a peer-to-peer video streaming application, in order to evaluate the impact on services that have time sensitive data processing. The third

[15] http://wiki.apparmor.net
[16] https://www.proxmox.com
[17] http://wiki.confine-project.eu

TABLE I. SUMMARY OF OUR SCENARIOS AND SETTINGS

Scenario	1	2	3
Number of local Community-Lab nodes	8	8	8
Number of slivers	8	8	16
Services deployed	Tahoe-LAFS	Peerstreamer	Tahoe-LAFS and Peerstreamer
Performance Metrics	Storage benchmark	Chunks Received and Played-out	Storage benchmark Chunks Received and Played-out

evaluation scenario combines both services and allows an evaluation of the concurrency of services within the same physical devices with our proposed deployment.

For each scenario we collected results and plotted them against the baseline values. The baseline values were obtained by running the same set of experiments on the Community-Lab infrastructure (from our groups' earlier works [7]) under the same set of configurations. This gives us the behaviour of the proposed system in terms of performance and user experience.

In the Tahoe-LAFS experiments we measured the performance for read and write operations in order to understand the impact these type of operations have on the proposed deployment. In the PeerStreamer experiments we measured the average chunk rates (data that is received in the peers side) and average chunks played out on peers (data that is sent to be watched in the peers side) in order to measure the quality of the video stream. In both cases we measured the CPU utilization to demonstrate that these low-power devices can deliver the multi-purpose execution environment while maintaining the multi-service community cloud model.

First Scenario: In our first evaluation scenario, we created one sliver for each available Commmunity-Lab node (eight slivers in total) and each sliver running the Cloudy OS. For each of the slivers we ran a Tahoe instance, Tahoe-LAFS is a service that comes bundled with the Cloudy OS and is used for distributed storage.

The scenario was tested in several runs using the same configuration for each, extending to an amount of 3 hours each. We used seven slivers as Tahoe-LAFS storage instances (these instances serve as storage for the files written by any client). One of these slivers ran Tahoe's Introducer (a publish-subscribe hub responsible to notify clients and storage nodes about each other) and the eighth sliver operates the Tahoe-LAFS client instance which can write or read files from a mounted folder that accesses directly the Tahoe-LAFS system. This is done through the use of sshfs and fuse kernel module[18] allowing a folder on the Tahoe-LAFS system to become available on any computer system seamlessly.

Furthermore we ran a well-established disk benchmark application, i.e. IOZone [8], to measure the storage operations of each node in order to evaluate the proposed deployment and compare it with the performance of the same services when using the current Community-Lab, at present time.

Results: Fig. 5 shows the measurements of the baseline system corresponding to the current Community-Lab environment, and the same set of operations on the proposed deployment (named set1). Note that all operations are completed when the transactions between instances are finished therefore the results account with the performance of all Community-Lab nodes used. In the proposed deployment there are two services running on the same physical devices therefore each Tahoe-LAFS system has concurrent access to the resources which is, as expected, reflected by a general lower operation speed for all the operations measured. However this still accomplishes the operations with the approximated speeds as the baseline evaluation.

[18]http://fuse.sourceforge.net/sshfs.html

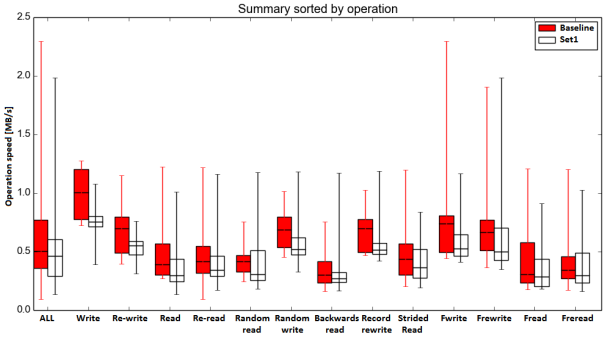

Fig. 5. Performance of Tahoe-LAFS service, baseline as the current deployment and set1 as within the proposed deployment on the first evaluation scenario (operations shown as average All, write, re-write, read, re-read, among others). Representing average, and deviations for each operation.

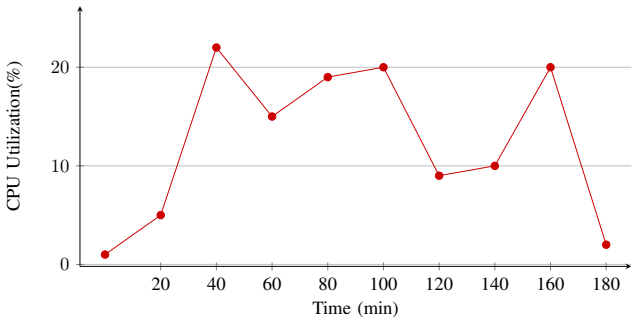

Fig. 6. CPU utilization on Tahoe-LAFS client node in the first evaluation scenario

The benchmark application stresses the device resources in order to find the maximum speed for operations such as reading/writing of files. It is important to notice that while network part is an important process in distributing files throughout the instances, our evaluation accounts more for the resource usage locally. Thus, CPU utilization is important to understand the system behaviour in the proposed deployment.

Fig. 6 shows the average CPU utilization during three hours of running the IOZone benchmarking application in the Tahoe-LAFS client instance. It has consumed around 20% of CPU time on average during the complete test. The client node performs encryption/decryption and erasure code computation on each file block in the phase of writing and/or reading to/from the disk. Furthermore the Tahoe-LAFS client instance consumes more CPU time as opposed to Tahoe-LAFS storages.

Second Scenario: In our second evaluation scenario, we used the same sliver deployment (one per node), running concurrently a Cloudy OS template. In each of these slivers, we ran a PeerStreamer experiment.

For this scenario, we ran four tests in different time periods, with 1 hour each run, in order to account for different network activity. We used seven slivers as peers (each peer retrieves data from the network in order to play out the streaming video locally). PeerStreamer uses an overlay network to exchange data (known as chunks) between its peers. We also setup one

sliver to serve as the source peer which disseminates the source video partitioned in chunks (as default, one frame of the video is one chunk) to be played out by the other peers.

The PeerStreamer source peer gets a live camera stream and sends the chunks to the overlay network between each peer that watches that stream. This results in each peer trying to fetch chunks from other peers to play out the continuous stream and display it locally to the users.

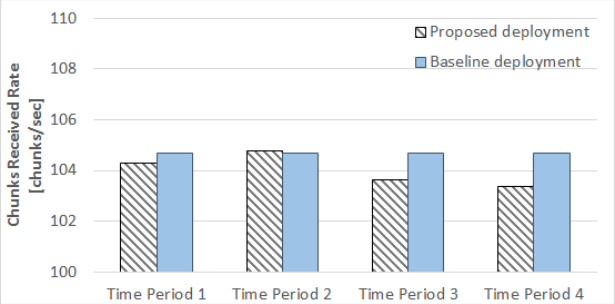

Fig. 7. Average chunks received rate at peers from PeerStreamer execution in the second evaluation scenario. Baseline as the current deployment in Community-Lab.

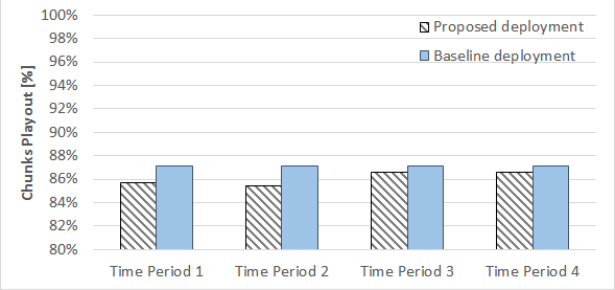

Fig. 8. Average play-out ratio at peers from PeerStreamer execution in the second evaluation scenario. Baseline as the current deployment in Community-Lab.

Results: Fig. 7 and Fig. 8 depict the measurements of the average of chunks that a peer can receive and the chunks percentage that were sent to be played out averaged by all peers. As a baseline we have the current deployment from previous experiments with the Community-Lab testbed in which the proposed deployment is able to reach without losing too much data on average. It is also noted that because of the shared resources there is a minimum amount of chunks that are not received in the proposed deployment. These time sensitive applications require that data should arrive on time to be displayed/processed, if the data does not appear in the time allotted it is discarded. Therefore in the proposed deployment this amount does not vary much from the baseline.

Fig. 9 shows the average measurements of CPU utilization during an hour, when the service runs continuously. The alterations we see in the CPU utilization is in fact because the PeerStreamer neighbourhood size changes over time (the overlay network is constantly updated even when there are no new peers) and therefore the utilization of the resources change when the operations of redoing the topology of the network is performed. Moreover the CPU utilization on the source node is higher than on the peer nodes since it transcodes the video stream and partitions it into chunks that will be sent to the

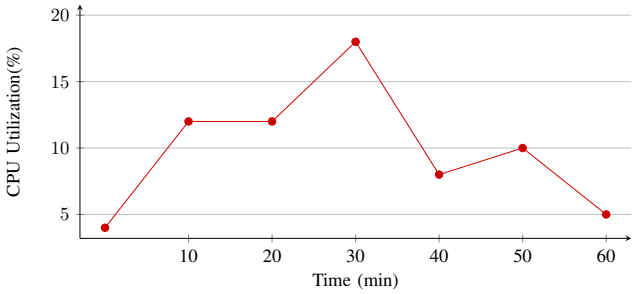

Fig. 9. CPU utilization on PeerStreamer (PS) source node on second evaluation scenario

peers. Thus the CPU utilization on the peers is considerably lower and do not interfere as much with other services running concurrently. This is a result of the peers running a more lightweight process such as gathering and decoding of chunks.

Third Scenario: In our third evaluation scenario, we ran both services (Tahoe-LAFS and PeerStreamer) in separate slices, running concurrently in a Cloudy OS template on the same local Community-Lab nodes. We measured the performance of our proposed approach while running different services in a concurrent way.

In this scenario we ran the same number of experiments as before, while cutting down the Tahoe-LAFS test to 1 hour. It also uses the same deployment of slivers as before, however, each physical device runs concurrently four slivers each with its own service and groups of two from the same slice.

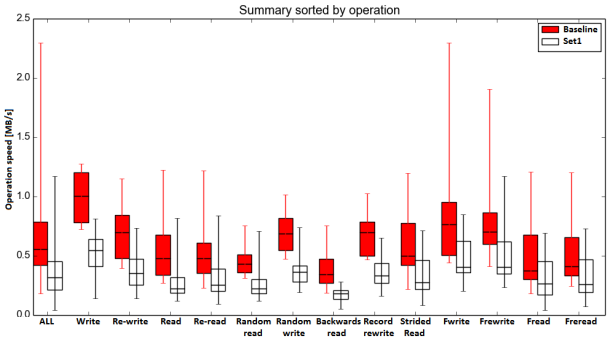

Fig. 10. Performance of Tahoe-LAFS service, baseline as the current deployment and set1 as within the proposed deployment in third evaluation scenario (operations shown as Average All, write, re-write, read, re-read, among others). Representing average, and deviations for each operation.

Results: Fig. 10 shows the results from the current deployment in Community-Lab as baseline, against the proposed deployment in Set1. It also shows that when different concurrent services are running on the same device, the impact on the services are noticeable, when there is a higher load of the system. However, in a normal usage of services the impact on the operations is attenuated with time, or with scheduling.

Fig. 11 and Fig. 12 show the results of the current deployment on the Community-Lab as a baseline against the proposed deployment results. We can see a noticeable, to a certain degree, variation of the chunks played out which affects the video quality perceived. This is due to the CPU time being shared among more processes. However, while concurrent services may differ, for our results we can say that the loss is minimal when using the proposed deployment against the current Community-Lab deployment.

The average CPU utilization of the Tahoe-LAFS client instance is shown in Fig. 13. These measurements were taken

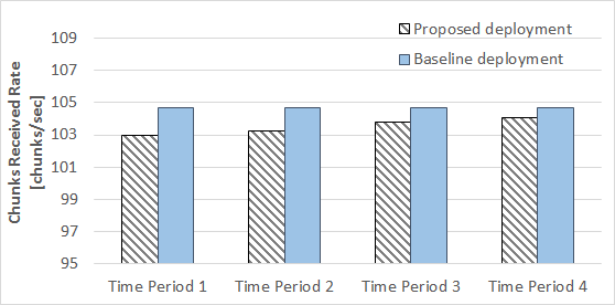

Fig. 11. Average chunks received rate at peers when Tahoe-Lafs is also running (third evaluation scenario). Baseline as the current deployment in Community-Lab.

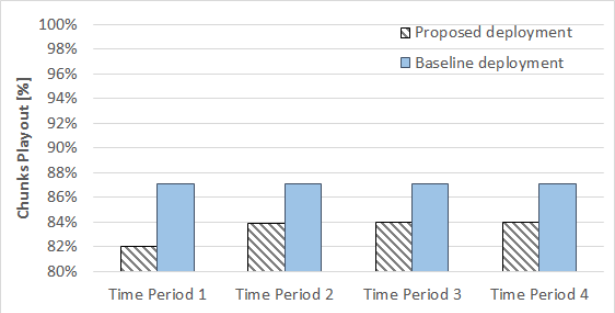

Fig. 12. Average chunk playout at peers when Tahoe-LAFS is also running (third evaluation scenario). Baseline as the current deployment in Community-Lab.

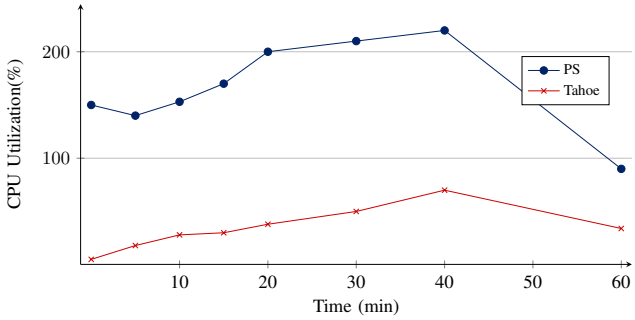

Fig. 13. Average CPU utilization of Tahoe-LAFS and PeerStreamer (PS) in third evaluation scenario

over one hour period in the third evaluation scenario. The figure shows the CPU utilization of the Tahoe-LAFS client instance is around 35% - 40% on average over the course of experiment, meaning that it uses at most half of one core of the device. This is a result of the added processing in the client instances while the storage instances have a lower process utilization, performing only read and write operations.

On the other hand, the average CPU consumption for PeerStreamer is nearly five times higher than what Tahoe-LAFS uses. This is because while running the last scenario we also recorded the video to the disk (in all peers) influencing more the resource utilization for the service in order to get the most utilization for each service. The CPU utilization on the peers remains lower than in the source peer and still feasible to be running concurrently with other services.

B. Discussion

We could observe that the proposed deployment has an impact on the service performance. However, we can say that it is minimal while achieving more services within one physical device. We can also state that both scenarios, LXC and QEMU, can be used according to the resources available. QEMU is suitable when services require higher security features, higher isolation of the kernel system and when resources are not constrained (high-power devices with hardware-enabled virtualization). LXC targets low-capacity devices to achieve concurrency between services.

The quality impact that the services have is minimum, when using services that do not require high demand of the resources. In our experiments the resource utilization was key to understand the impact and how we can optimize the shared resources. The results of our evaluation indicate variability on the execution of concurrent services. One of the factors for this is the user context has to be changed for each process. This can result in minor delays until the service can fully utilize the CPU.

Moreover we can say that using such deployments may guarantee isolation of services and maintain mostly the same quality of service. Therefore, our approach showed to be feasible for the current deployment of devices on community networks with minimal performance loss in the services. Also achieving user context isolation on the devices for multiple purposes and guaranteeing the owners isolation and utilization of the devices. Further study to achieve and optimize the maximum amount of contexts or services within a device will be dealt in future work.

VI. Related Work

In this section we review some works which address similar issues as this paper. In [9] the authors report on how the deployment of the cloud model on top of IP-based community networks in the case of Guifi.net was undertaken. They elaborate a system for the proposition that the users can benefit from cloud-based services inside of the network without having to consume them from the Internet. Instead they can utilize the resources that already exist in their community network, while also granting access to cloud-based services.

The area of Fog computing [10] is related to our work in the concept of integrating edge devices. To this end, Fog computing aims to extend data center-based cloud computing by integrating hosts at the edges of the networks into the cloud process. Edges are seen as being proactive components for services, data usage and storage.

In the Personal clouds proposal [11], the authors enhance the capabilities of mobile devices, seen as low-powered devices, by using either remote or nearby cloud resources, instead of having to process data locally and thus reducing consumption within the mobile devices. Furthermore, this work explains how network resources are to be integrated in a heterogeneous environment, while enhancing the user experience. In this way, each device can be seen as a single device cloud and active participants can run the services which are of interest to the end users.

The work in [12] shows how clouds that have under-utilized resources can be enhanced by sharing these resources with other communities, while still maintaining the same aspects that the cloud owners have agreed upon.

Work reported in [13] describes techniques that may satisfy the offload computation of mobile devices. A comparison is provided to better understand and succeed when doing processing on low-powered devices. This system only accounts for service concurrency and not user independent, thus only

considering the virtualization layer for service to run concurrently in a low-powered device.

In the Paradrop system [14] the authors have created a platform in which low powered resources are used, such as home gateways, in order to deploy different services running and processing concurrently and with different data. Trying to get the most use out of such devices, while also taking advantage of the parallelism that devices can create between computations. This system only accounts for service concurrency and not user independency, thus it only considers the virtualization layer for services to run concurrently in a low-powered device.

The work in [15] demonstrates that container-based system virtualization is performed well over hypervisors by giving the opportunity of isolation in terms of security and resources. Their results show that container-based system virtualization provides up to 2x the system performance of hypervisors for server-type workloads and scale further while maintaining the system performance at a higher level.

VII. Conclusion and Outlook

Community networks are IP-based networks that can have many nodes that are connected by wireless links through a territory. These networks were initially started to give their members access to the Internet. We argue that these networks are underutilized if only Internet access is targeted. The resources in the network can actually support more services, which can give community members advantages and benefits when participating in shared services.

The container-based lightweight virtualization approach proposed in this paper for contributed low-capacity devices, maintains the benefits of running concurrent services on these low-power devices and delivers a multi-purpose system. Also it features isolation of both resources and security whereby guaranteeing a better overall system performance even on resource-constrained devices. This paper also addressed the current limitations of community cloud infrastructures such as the scalability of the existing infrastructure, and the performance of low-power devices.

We demonstrated that through virtualization we can deliver a multi-purpose environment. This leads us to run multiple services in low-power devices (which are similar to the resources shared in community networks) to guarantee the community cloud environment remains feasible while giving the owners a space for their own usage. We replicated the Community-Lab infrastructure to deploy different services in the devices available in order to understand the performance issues that our proposed approach can have.

From the evaluation performed, we showed that the impact on the performance of running concurrent services, while in different user contexts is minimum compared to the current state of the devices, deployed throughout the community network. In fact, in the proposed approached the evaluated services perform much the same way as in the current Community-Lab approach, suggesting that the proposed approach is feasible to deploy on such infrastructures.

Moreover the proposed approach can augment the established network infrastructure of community networks and the ability for users to have access to more computing power, exploring the community cloud environments and achieving a multi-purpose system. This way we can add value to the community networks, without deploying more physical devices, and lowering the impact on the services when running concurrently within the community network infrastructure.

As a next step, we consider to deploy and monitor resources using the proposed approach in the Community-Lab testbed, such that we can identify configuration or security issues that may hinder the performance of the services. Also, we will consider different types of services running concurrently to specifically improve and optimize their performance applying the proposed approach.

Acknowledgement

This work was supported by the European Framework Programme 7 FIRE Initiative projects CONFINE (FP7- 288535), CLOMMUNITY (FP7-317879), Universitat Politcnica de Catalunya-BarcelonaTECH and by the Spanish government under contract TIN2013-47245-C2-1-R.

References

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

[2] Z. Wilcox-O'Hearn and B. Warner, "Tahoe: The least-authority filesystem," in *Proceedings of the 4th ACM Int. Workshop on Storage Security and Survivability*, ser. StorageSS '08. New York, NY, USA: ACM, 2008, pp. 21–26.

[3] M. Selimi and F. Freitag, "Tahoe-LAFS Distributed Storage Service in Community Network Clouds," in *4th IEEE Int. Conference on Big Data and Cloud Computing 2014*. Sydney, Australia: IEEE, Dec. 2014.

[4] L. Baldesi, L. Maccari, and R. Lo Cigno, "Improving P2P Streaming in Community-Lab Through Local Strategies," in *10th IEEE Int. Conference on Wireless and Mobile Computing, Networking and Communications*, Larnaca, Cyprus, October 2014, pp. 33–39.

[5] R. Birke *et al.*, "Architecture of a network-aware p2p-tv application: The napa-wine approach," *IEEE Communications Magazine*, vol. 49, pp. 154–163, 06/2011 2011.

[6] M. Selimi *et al.*, "Cloud-based extension for community-lab," in *22nd Int. Symposium on Modelling, Analysis Simulation of Computer and Telecommunication Systems, IEEE*, Sept 2014, pp. 502–505.

[7] M. Selimi, F. Freitag, R. Centelles, and A. Moll, "Distributed Storage and Service Discovery for Heterogeneous Community Network Clouds," in *7th IEEE/ACM Int. Conference on Utility and Cloud Computing (UCC'14)*. London, UK: IEEE/ACM, Dec. 2014.

[8] W. D. Norcott and D. Capps, "Iozone filesystem benchmark," *URL: www. iozone. org*, vol. 55, 2003.

[9] J. Jimenez *et al.*, "Supporting Cloud Deployment in the Guifi.net Community Network," in *5th Global Information Infrastructure and Networking Symposium (GIIS 2013)*, Trento, Italy, Oct. 2013.

[10] F. Bonomi *et al.*, "Fog computing: A platform for internet of things and analytics," in *Big Data and Internet of Things: A Roadmap for Smart Environments*, ser. Studies in Computational Intelligence, N. Bessis and C. Dobre, Eds. Springer Int. Publishing, 2014, vol. 546, pp. 169–186.

[11] M. Jang *et al.*, "Personal clouds: Sharing and integrating networked resources to enhance end user experiences," in *INFOCOM, 2014 Proceedings IEEE*, April 2014, pp. 2220–2228.

[12] B. Saovapakhiran and M. Devetsikiotis, "Enhancing Computing Power by Exploiting Underutilized Resources in the Community Cloud," in *IEEE Int. Conference on Communications (ICC 2011)*, Kyoto, Japan, Jun. 2011.

[13] K. Kumar *et al.*, "A survey of computation offloading for mobile systems," *Mob. Netw. Appl.*, vol. 18, no. 1, pp. 129–140, Feb. 2013.

[14] D. F. Willis, A. Dasgupta, and S. Banerjee, "Paradrop: A multi-tenant platform for dynamically installed third party services on home gateways," in *Proceedings of the 2014 ACM SIGCOMM Workshop on Distributed Cloud Computing*, ser. DCC '14. New York, NY, USA: ACM, 2014, pp. 43–44.

[15] S. Soltesz *et al.*, "Container-based Operating System Virtualization: A Scalable, High-performance Alternative to Hypervisors," in *EuroSys'07, the European Chapter of SIGOPS*. Portugal: ACM SIGOPS, 2007.