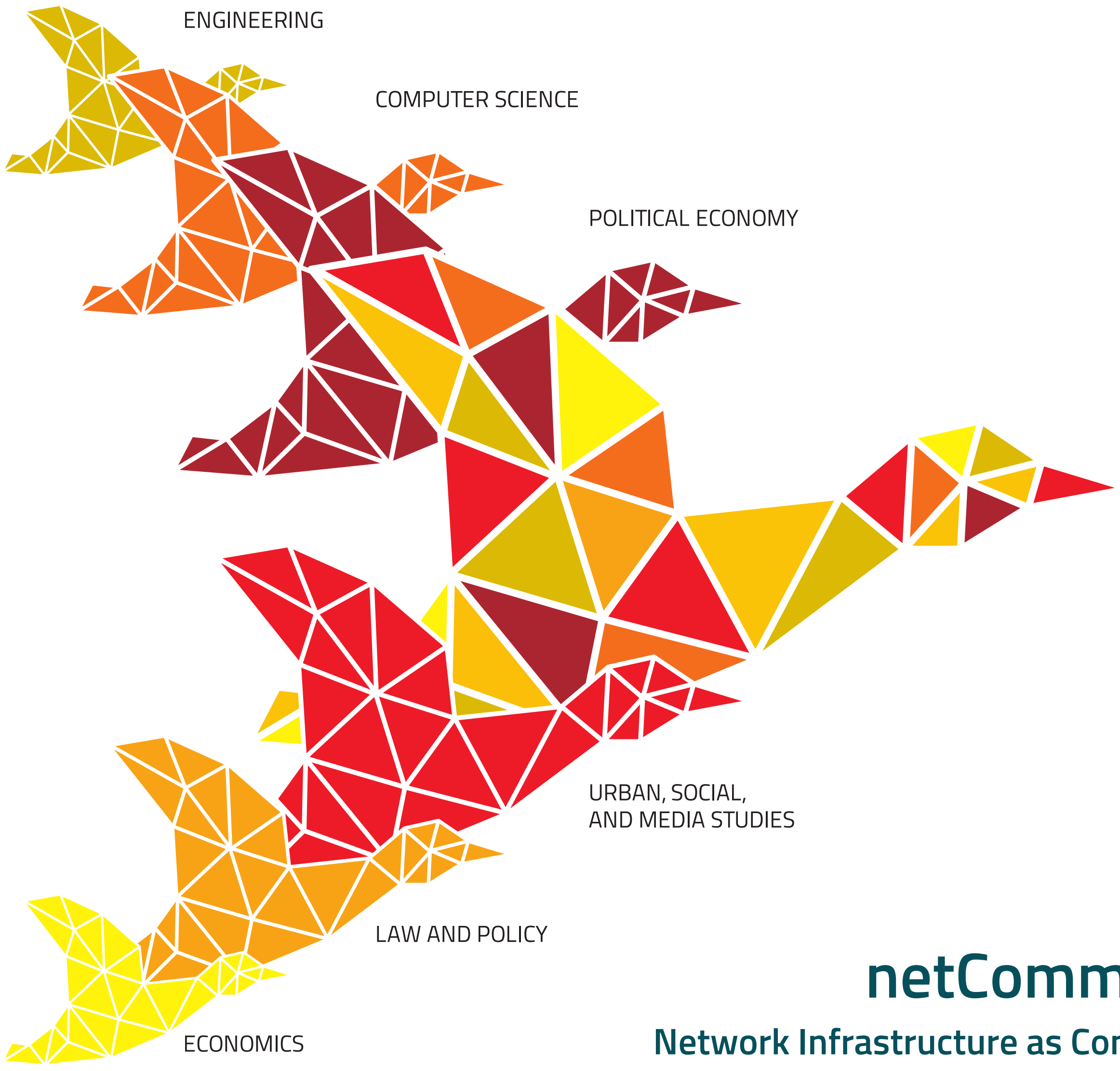


netCommons

Network Infrastructure as Commons

Report on the Results of the Socio-Technological Experimentation of Open Source Software

Deliverable Number D3.5
Version 1.0
January 24, 2019

Co-Funded by the Horizon 2020 programme of the European Union
Grant Number 688768

netCommons.eu

Project Acronym: netCommons
Project Full Title: Network Infrastructure as Commons.
Call: H2020-ICT-2015
Topic: ICT-10-2015
Type of Action: RIA
Grant Number: 688768
Project URL: http://netcommons.eu

Editor:	Leonardo Maccari, UniTN
Deliverable nature:	Report (R)
Dissemination level:	Public (PU)
Contractual Delivery Date:	December 31st, 2018
Actual Delivery Date	January 24, 2019
Number of pages:	115
Keywords:	community networks, applications, open source, participatory development, Cloudy, PeerStreamer, AppLea
Authors:	Leonardo Maccari, Luca Baldesi, Renato Lo Cigno, UniTN Felix Freitag, Leandro Navarro UPC Merkouris Karaliopoulos, Aris Pilichos AUEB-RC
Peer review:	Leonardo Maccari, Leandro Navarro, Merkouris Karaliopoulos, Renato Lo Cigno

History of Revisions

Rev.	Date	Author	Description
v0.1	5/12/2018	Leonardo Maccari	First draft
v0.2	17/12/2018	Leandro Navarro, Felix Freitag	Cloudy related chapter, general updates
v0.3	20/12/2018	Leonardo Maccari	PeerStreamer-ng, review of Cloudy
v0.4	27/12/2018	Leonardo Maccari	letters, conclusions and intro
v0.5	28/12/2018	Leandro Navarro, Felix Freitag	Reworked first part of Cloudy chapter
v0.6	31/12/2018	Merkouris Karaliopoulos, Aris Pilichos	Added section 4.1 and first part of screenshots for the AppLea chapter
v0.7	4/1/2018	Merkouris Karaliopoulos, Aris Pilichos	Added section 4.2 and additional screenshots
v0.8	11/1/2018	Merkouris Karaliopoulos, Aris Pilichos	Added section 4.3 and final round of screenshots, reworked section 4.2
v0.9	14/1/2019	Leonardo Maccari	review of sect. 4
v1.0	22/1/2019	Renato Lo Cigno	Final proof reading and harmonization with other deliverables

Executive summary

This report is the follow-up of D3.2 and D3.4 in which we documented the state of the software development carried on in netCommons at the end of M12 [1] and at the end of M24 [2]. In these reports we introduced and described the advances in the Cloudy, PeerStreamer and the CommonTasker application (now re-named AppLea), that are the open source applications being developed in netCommons. While the first two were started in previous EU-financed projects (the CLOMMUNITY and NAPA-WINE FP7 projects) the latter is a new application developed from scratch in netCommons. This deliverable reports on the development of these platforms in the third year of the project, in which we tested them with the help of some selected communities, and continued their development. It also gives additional feedback to the methodology described in T3.1. As such, this deliverable describes the lessons learned from the usage or our open source software in the communities, the feedback we received and how it helped the software to become fully usable.

Summary of Development The Cloudy platform was further expanded to support the IoT world. Chapter 2 reports on this new direction we opened in order to make Cloudy even more appealing to the users of Guifi Community Network. Furthermore, Cloudy was at the center of the development of two new technologies, Information-Centric Networking (ICN) and Blockchains. In the first case, the Umobile project[1] used Cloudy as the base platform to test, for the first time the use of ICN on mesh networks. We report on these advances and current ongoing publications. In the second case, UPC was involved into the experimentation of Blockchain for mesh networks by the AmmbrTech start-up, which is a recently created company investing in Blockchain-based mesh networks.

The PeerStreamer-ng application was completed with PartyHub, a component to realize many-to-many video-conference. PartyHub is now included in the main tree of PeerStreamer-ng and it can be used both on-line and within community networks. We have tested PeerStreamer-ng and PartyHub in several occasions and it was used in two groups of the ninux community network.

Finally, AppLea has been completed. The app was largely modified compared to the initial design, thanks to constant interaction and feedback received from the members of Sarantaporo-gr community network. It is now stably used by a group of farmers in the Sarantaporo-gr community network.

Furthermore, additional developments emerged in the interaction with CNs (ninux in particular) that netCommons followed and nurtured, fostering dissemination and impact of the project.

As for D3.2 and D3.4 all the source code released during the project is available in our Github organization[2]

Summary of Adoption Our initial goals for WP3 was to have at least two community networks to test and adopt our software. In this deliverable, (together with D3.6 for the part dealing with the participatory methodology) we describe how we were able to involve four community networks from three countries to adopt, use and co-develop our software. We show that not only we had our open source software used by a community of users, but also how the influence of the project and the methodologies we proposed were of interest outside of these specific communities and outside the world of community networks.

Update on the Multi-Disciplinary Methodology for Applications Design Albeit the work of T3.1 officially ended with D3.6, NetHood continued the development of the methodology to produce a stand-alone booklet with the latest version of the methodology. The booklet is an investment we deemed necessary to produce a publication that is easy to use and read for any CN member, without the involvement of people from the project. The current version of the booklet (which by nature will change and update in the future) is included in this deliverable.

[1] UMOBILE: H2020 #645124 project: https://cordis.europa.eu/project/rcn/194285/factsheet/en, http://www.umobile-project.eu/
[2] See https://github.com/netcommonseu with specific reference to the PeerStreamer-ng, Cloudy-netCommons and AppLea repositories.

Contents

List of Figures

List of Tables

List of Acronyms

API	Application Programming Interface
BASP	Bandwidth and Availability-aware Service Placement
CA	Certification Authority
CDN	Content Distribution Network
CN	Community Network
CMN	Community Mesh Network
CPU	Central Processing Unit
DNS	Domain Name System
DTN	Delay Tolerant Network
DoA	Description of Activity
FOSS	Free and Open Source Software
GUI	Graphical User Interface
HANET	HArdware and NETwork Resources
HLF	Hyper-Ledger Fabric
ICN	Information-Centric Networking
IoT	Internet of Things
ISP	Internet Service Provider
IPFS	InterPlanetary File System
JSON	JavaScript Object Notation
LIDAR	Laser Imaging Detection and Ranging
MAS	Multi-Agent System
MEC	Multi-Access Edge Computing
NAT	Network Address Translation
NDN	Named Data Networking
NFD	Name based Forwarding
PDI	Pentaho Data Integration
PoA	Proof of Authority
PoW	Proof of Work
PS-ng	PeerStreamer-NG
QMPSU	Quick Mesh Project Sants and UPC
QoE	Quality of Experience
QoS	Quality of Service
RTP	Real Time Protocol
SC	Service Controller
SEG	Service Execution Gateway
SME	Small and medium-sized enterprise

TTN	The Things Network
VPN	Virtual Private Network
WCN	Wireless Community Networks
WebRTC	Web Real Time Communications

1. Introduction

WP3 is dedicated to open-source applications for CNs. As described in the Description of Activity (DoA) the work is divided in three tasks T3.2, T3.3, and T3.4 where we develop and experiment three different open source applications for use by CNs. The three tasks delivered the latest version of the software in M24, as reported in D3.4 [2]. During the period M24–M36, in T3.2 we have continued our experimentation with the Cloudy software in the Guifi network; in T3.3 and T3.4 we have continued the development of software, PeerStreamer-NG and AppLea and we have involved communities in the testing and adoption of the proposed solutions. Furthermore, we have produced a simplified version of the methodology studied in T3.1.

The main body of this deliverable is structured in four chapters.

Chapter 2 describes the work in T3.2. The Cloudy software has been under light maintenance in the third year of the project, only with new areas and new mechanisms explored from feedback from the community of users and collaborations with complementary projects and industrial interest. The work in T3.2 focuses on the integration of other technologies into Cloudy, with the cooperation of other research projects and companies that were interested in the use of Cloudy. Among them we mention the integration into Cloudy of technology related to the IoT, and the first implementation of a blockchain-based solution to solve some of the open issues in community networks.

Chapter 3 describes the work in T3.3. This task is devoted to the continuous development of the PeerStreamer-NG (PS-ng) application and its testing and adoption in the ninux network. In this period the application was further developed to support the video conferencing use-case, and in general to stabilize the code base based on the feedback received by the users. We did interact, showcased and support the use of PS-ng in four different scenarios, two public events and two communities. The first public event was the Battle of the Mesh meeting in Berlin in May 2018, in which we deployed PS-ng in the network that the CNs deploy to test their protocol. We tested PS-ng in such a challenging scenario and collected feedback from the participants, that led to the development of new features. The second experimentation was held in Sarantoporo, during the meeting we organized in July 2018. In this case we used PS-ng to stream the public conference we organized from one Sarantaporo node to another, and again, we collected important feedback. Finally, we involved the two ninux “islands” in the adoption of PS-ng. In the first case we used PS-ng in the island of Florence, in which PS-ng was used to publicly stream two “hacknights” organized by the CN together with other organizations (as documented in [3], in the second, we involved the island in Calabria to use PS-ng.

Chapter 4 describes the work in T3.4. In T3.4, we have been developing an Android app for logging and organizing the farming activities of the Sarantaporo.gr community. The last year of the project marked the beginning of the field experimentation with the app, named AppLea under real conditions of intended use. The main actors in this experimentation was the beta testing team that was set up out of members of the local community. Their feedback and ideas were the main driver behind the evolution of the app functionality in this last year. The added functionality mainly relates to the user profiles stored in the app, its privacy control features, the reports that can be produced out of the posts made by farmers and useful statistics that can be shown to them regarding the use of resources in their farming activities. We also assess the impact the app has made on the Sarantaporo.gr CN and iterate on its dependence on non-local tools (e.g., Firebase backend).

Chapter 5 reports a continuation of the work in T3.1. Going beyond the DoA, T3.1 produced also a simplified version of the participatory design methodology in a published booklet, which we include in the appendix of this document and briefly describe in Chapter 5.

2. Update on the Cloudy community cloud distribution

This chapter describes the evolution of the Cloudy community cloud distribution following the feedback and needs raised from interaction with the guifi.net community network and other interested parties and initiatives. Although we describe in more detail the work in the last year, we provide first a perspective of the evolution of Cloudy to clarify the context and starting point. As the development effort for Cloudy in netCommons was essentially finished at M24, most of the work presented derives from analysis of scenarios and research in application contexts, new enablers, as well as in possible future extensions of the platform.

2.1. Positioning the evolution of Cloudy in the context of guifi.net and beyond

After an initial development supported by the Clommunity project[1], Cloudy started its deployment in guifi.net in 2015. The number of Cloudy instances raised quickly to around some tenths of deployed nodes (around 50) as described in more detail in Section 6 of [4]. The number of instances (operational servers) we observed during the last years is quite steady and has not changed significantly[2], and a progressive replacement of the low cost devices being used. Typically we can observe (from the node monitoring system) between thirty to fifty nodes, as the discovery protocol is gossip based and network delays or partitions can affect the group membership view. Cloudy nodes were and are typically installed by the most active guifi.net members[3], possibly a number that is not fluctuating much in the community, and not by those community network members which use the guifi.net network just to access the Internet. Without a dedicated development and support team the maintenance effort is manageable by the community of developers, researchers and users. Without logs to quantify new device installations, we estimate that at least 1/3 of the current Cloudy nodes where installed since the beginning of netCommons. We include in Sec. 2.9 questions and replies from a few Cloudy users at end of 2018, that volunteered to provide feedback about their own experience. They belong to a rural (Ribagorza valley) and a urban (Barcelona) community, part of the guifi.net CN.

In the early version of Cloudy, a fixed set of predefined services were given. Each service had a user-friendly installation option. These services could be classified into two groups, namely services for network management and applications for users. We observed that the network management service had some adoption by the users, who appreciated the much easier installation of guifi.net internal DNS servers and network monitoring servers which Cloudy provided. The predefined user-oriented applications, however, had little success with regards to take-up. Applications included file synchronization, decentralized storage and video-streaming, all based on reasonably consolidated third party open-source software integrated in Cloudy.

We extended Cloudy during 2016 and 2017 with container-based deployment mechanisms, taking advantage of the consolidation of Docker and the increasing number of tools to deploy applications by container instances. The shift from predefined applications to enabling new additions with containers made any application available as Docker image with a web user interface deployable in Cloudy. As a consequence, the scope of applications which could be run in Cloudy became much larger. The consolidation of the docker-compose tool was picked up for the Cloudy development and we integrated in 2017 into Cloudy. Compared to deploying with just Docker, the compose functionality makes it more easy to deploy complex applications consisting of several interrelated Docker images. Overall, during 2016 and 2017 more features were added to Cloudy which kept

[1]Clommunity: A Community networking Cloud in a box, 2013-2015, http://clommunity-project.eu/

[2]As the number of participants in the support mailing list, quite even, with new participants compensating those those leaving or becoming silent.

[3]Willing to experiment and run guifi.net services like local Domain Name System (DNS), web proxies offering free Internet/Web access, network monitoring servers, or application-oriented experimental applications for messaging, file sharing, etc.

Cloudy up to date and ready with regards to the overall adoption of container technology for the deployment of applications in cloud infrastructures. In 2018 no new features were released in the public distribution, being in a maintenance phase.

The feasibility analysis of community clouds in general and Cloudy in particular shows the value of the scenario of 'platform commons', but also its difficulties. The complexity of development, integration and maintenance of a community cloud platform is another major difficulty and risk. As we described in [4], software complexity requires a committed and balanced community of developers, maintainers, and users. The guifi.net experience, our experience and the experience of other CNs shows the need for a combination of volunteer and professional developers. In fact, it is clarifying to realize that the community of developers involved directly or indirectly in the development of Cloudy components is much larger (at least two orders of magnitude[4]) larger than the current user community: just the Debian community, the basis for Cloudy, involved 1625 people and 19 teams in 2018[5], or the Docker community has more than 3,000 contributors, and Docker Inc. has more than 250 employees. To this end, it is necessary to develop a value chain that ensures a minimum revenue stream to pay the professional developers and volunteers.

Sustainability of a volunteer computing infrastructure that goes beyond hobbyist usage has been demonstrated before by some cases, but specific conditions applied that made it successful. Peer-to-Peer file sharing was taken up by millions of users by the attraction of readily accessible content. Volunteer computing projects which support scientific research gathered large pools of resources from contributors.

guifi.net created the compensation system in order to bring in cost sharing and regulate commercial activity in the volunteer-contributed communication network. As a consequence, volunteers and commercial service providers co-exist in the guifi.net ecosystem, and help to make the infrastructure sustainable and scalable, further developed and discussed in [5].

The lessons of the success of the economic compensation system made us focus, in 2017 and 2018, on exploring means which could make Cloudy sustainable, described and justified in detail in [4], which lead to some of the results described in Sec. 2.2.

2.2. Results in 2018

The core of the work related to Cloudy in the third year of netCommons relates to improvements beyond the implementation of the M24 software release in D3.3 [6], and the detailed definition of the governance, implementation and evaluation in D1.4 [7], and the corresponding journal publication [4].

Beyond that, the Cloudy software has been under light maintenance in the third year of the project, but with new areas and new mechanisms inspired by feedback from the community of users and collaborations with complementary projects and industrial interest in Cloudy. The feedback from the community of users has come through the discussions we had over the year in the weekly informal open lunch meetings on Wednesdays at the UPC North campus, including a variable group of researchers, guifi.net community members, current and former Cloudy developers, and discussions in a few of the weekly guifi-labs in Barcelona[6]. As part of our participatory action research and experimentally-driven research, we have worked with the guifi.net community to understand the needs, and co-create (design, develop, test, and evaluate) the solutions. The group discussions were around the planning, co-design, transformation, and result phases of the action research, with a focus on understanding the needs, the technology choices, algorithms and the software development of services that enable certain applications or scenarios. These results however, are still not integrated in the current Cloudy distribution. Additional non-trivial development and integration effort will be required beyond the timeline and resources of this project. The needs, work and results obtained follow several intertwined threads described next.

[4]https://en.wikipedia.org/wiki/Order_of_magnitude
[5]Debian statistics: https://wiki.debian.org/Statistics
[6]https://exo.cat/guifilab/

The first thread we followed was about **new application domains** for Cloudy which make the platform useful. We selected Internet of Things (IoT) due to its business potential, social relevance and interest in the guifi.net CN among others. This work is in the context of an ongoing collaboration with The Things Network (TTN)[7] in Barcelona (between UPC, guifi.net and TTN) where we have set up a LoraWAN IoT gateway at UPC and several sensor nodes. We demonstrated IoT application usage in Cloudy in [8] with a collaboration with Hitachi, integrating their commercial Pentaho IoT service, summarized in Sec. 2.4.

A second thread of work was **improved applications**, illustrated by the need and opportunity to introduce improvements in Internet gateway selection for beneficiaries of this service in guifi.net. The guifi.net Foundation and members of the guifi.net community maintain hundreds of Web/Internet gateways (web proxies) some of them based on the Cloudy web proxy component. We have worked on improved algorithms to select the best gateways as a way to optimize the perceived Internet-access quality and performance of the selected gateway while minimizing the selection overhead. This is an ongoing work that will be part of a future Cloudy release, detailed in [9] and [10], summarized in Sec. 2.3.

A third thread of work dealt with the need for **coordinated deployment of services involving multiple service instances** on a community cloud. That is the result of a discussion on the fact that that some sensitive services require multiple coordinated instances to deliver a single service. We have evaluated a lightweight service deployment mechanism (algorithm) in micro-clouds [11], summarized in Sec. 2.5, and an evolved mechanism for content and service deployment (PiCasso) in the context of information-centric networks and services [12] deployed in CNs, in collaboration with the UMOBILE project[8], summarized in Sec. 2.6.

The fourth thread followed the idea of guifi.net's **economic compensation system**, which needs the accounting of resource usage and contribution to compute the economic balances across participants. Thanks to a long-standing community of interest in guifi.net around blockchain technology, with discussions around the concept of "guificoin" and alternative currencies [13], and the availability of usable blockchain platforms became an interesting candidate to build a decentralized accounting system by means of distributed ledgers. Progress in this direction was achieved in the evaluation of blockchain platforms reported in [14], summarized in Sec. 2.7, and by the analysis of how Cloudy can support the governance of decentralized edge microclouds with blockchain-based distributed ledgers, elaborated in [15] and summarized in Sec. 2.8.

Combining an open source platform with industrial interest raises opportunities for replication and sustainability to explore. The opportunity to collaborate with the AmmbrTech SRL[9] company on blockchain technology for an economic system for community resource provision became important in order to learn about suitable tools and software, and understand how sustainability can be supported by means of a economic coordination service. As a result, new mechanisms have been designed to coordinate data and value flows using Blockchain technology, required for sustainable network infrastructures that can be openly expanded by participants that can play both the role of producers and consumers of connectivity. The modelling work is described in [16]. The development of a pilot system is under development, and will be completed after the end of netCommons. The main research themes look at consensus Proof of Authority (PoA), smart contracts (Solidity), and the integration of a network monitoring service using the Prometheus Free and Open Source Software (FOSS)[10] for traffic accounting and economic compensation. We consider this cooperation between UPC and AmmbrTech an important outcome of netCommons and WP3 in particular. Our work on community clouds, and the adoption from a community of users is a relevant asset for a company that wants to develop wireless routers for large-scale mesh networks adoption. The challenges to solve are the ones known in the literature and the ones we studied in netCommons, such as the scalability of the network, the distribution of some still centralized systems (accounting, IP addressing, authentication) and the governance of a large distributed community. Blockchains

[7] TTN in Barcelona: https://www.thethingsnetwork.org/community/barcelona/

[8] UMOBILE: H2020 #645124 project: https://cordis.europa.eu/project/rcn/194285/factsheet/en, http://www.umobile-project.eu/

[9] A research collaboration agreement between UPC and AmmbrTech / Ammbr Research Labs was signed and funded by AmmbrTech for the period of February-December 2018. The results will be applied in a new contract for 2019 to run pilot experiments in different communities around the world.

[10] See https://prometheus.io/.

have been proposed to solve some of these problems by netCommons members [17, 18] and the fact that a private company invested funds to start a cooperation with one of the netCommons partners is a clear proof of the impact of the project.

We considered to evaluate and redesign the user interface of Cloudy in 2018 applying the netCommons collaborative design methodology presented in D3.6 [3]; however, this task evolved slowly in the discussions with the community and got postponed beyond the end of netCommons to be able to include the results of other activities and ideas that the community is pursuing, including the idea of a mobile application that would complement the current Cloudy servers.

On the edge computing platform side, by the end of 2018, we can observe the appearance of open source edge computing platforms, as exemplified by Cloudy since 2015, such as EdgeXFoundry[11] developed under the umbrella of the Linux Foundation, which may become very strong in the industrial domain. We observe also the need for 5G edge computing infrastructures, where the foreseen Multi-Access Edge Computing (MEC) deployments by network operators will be insufficient to cover the demands of applications. The need for an edge cloud computing layer close to the end user than the MEC infrastructure, could be addressed by Cloudy. Interoperation at the resource and service layer with diverse actors, however, will need accounting and governance services in Cloudy, for which we expect that our work on blockchain has brought us closer to integrate this technology into Cloudy.

In the following sections we elaborate and summarize details about the specific areas of development in 2018 with results already introduced above.

2.3. Collaborative informed gateway selection

Thanks also to the netCommons legal work, Wireless Community Networks (WCN) are gaining more popularity for self-provision of Internet access in underserved regions [19]. The connectivity to the Internet is arranged through several gateway nodes among the client nodes in a large-scale wireless network. One of the main challenges is to fairly distribute the available Internet bandwidth among client nodes while maintaining a good overall performance perception at the same time. In guifi.net there are about 12,000 nodes using 394 active web proxy nodes (January 2018) acting as Internet gateways [20, 21].

There are several forms of middleboxes that can act as Internet gateways, ranging from simple IP routers to devices requiring application intelligence [22], implementing firewall or network address translation services, or HTTP connection pooling and content caching in web proxies. In guifi.net web proxies (usually based on the Squid implementation) have been the preferred way for participants to share Internet access with other guifi.net participants, being able to control the maximum network throughput offered or simply share their spare Internet capacity, reuse public content through content caches, provide some degree of privacy hiding the address of the origin host, or enabling public entities to provide free Internet access without infringing telecom market competence regulations. For these reasons, our work has focused on the case of web proxies as Internet gateways.

End-users configure already known and typically nearby proxy servers, adding them manually to a web browser extension that switches to the next proxy in the list when the current fails. The choice of a proxy in each client is not based on performance, but only on the failure of the current choice, and thus can be considered considered quasi-static. In contrast to that, the overall network structure is heterogeneous and the performance of the service offered by the gateway nodes can fluctuate significantly [23]. During peak hours, the Quality of Experience (QoE) perceived by end users varies depending on the choice of the gateway node. In order to achieve better QoE, network nodes need to monitor periodically the service performance they achieve from the gateway nodes.

There is a large body of work on the gateway selection problem, not only in Wireless Networks but also due

[11] https://www.edgexfoundry.org/

to the rising number of Internet of Things devices that need efficient gateway selection algorithms. Most of these works have some important limitations: despite proposing interesting solutions, they tend to fail in considering heterogeneity, monitoring overhead, and lack practical testing in a real-world environment. We refer the interested reader to the discussion on the state of the art in [9] and [10].

The majority of the gateway selection algorithms in the literature involve explicit measurement of all the gateway nodes by each client, which is not suitable for a large-scale network setting since the cost of measurement can outweigh the benefits of gateway selection. Even though the overhead of a single measurement request is small, in a resource-constrained, large-scale network, the overall traffic generated by measurement requests from every node becomes a major contributor to network congestion. Additionally, many gateway selection algorithms test all the options to select one best suitable gateway for the node. Thus, in the end, a significant amount of measurements and message exchanges is left unused. We argue that a partial view of gateways among different small groups of nodes would be effective in terms of reducing in-network traffic, measurement overhead and balancing the client nodes over the gateways.

The main goal of our investigation was to provide an efficient and effective gateway selection by reducing the gateway monitoring overhead, yet providing recent measurements to the client nodes. We argue that the gateway selection algorithm in large heterogeneous WCN can benefit from performance estimations at a lower cost through collaborative information sharing between network nodes.

We proposed a set of algorithms for informed gateway selection that reduces the gateway monitoring overhead by random sampling and distribution of the information within a localized group of client nodes. We found a good balance between explicit measurements and random sampling while keeping the information and decisions on the clients, leaving the gateway nodes unmodified. We deployed our algorithm in an experimental heterogeneous environment and quantify its efficiency and effectiveness and the influence of collaborative performance measurements.

2.3.1. Design of the gateway selection algorithms

We propose a client-side informed gateway selection algorithm where network nodes collaborate with closely located neighbor clients to sense the performance of the gateways. Each node keeps a table, called *gateway performance table*, where it stores the results of its own gateway performance measurements for the different gateways as well as the gateway measurement results obtained from its neighbor nodes. The main objective of collaborative sensing is to reduce the in-network traffic and to increase the awareness about gateway nodes at each client node.

Parts of the algorithm can be replaced (for example, we propose two different procedures to select the best gateway). We have therefore structured our approach into different components that are organized in three layers (see Figure 2.1): The bottom layer provides the performance sensing, that is the measurement of the gateway performance and the identification of close neighbors to collaborate with; the middle layer is in charge of the actual collaboration between network nodes, i.e., the exchange of measurement results; the top layer selects one gateway from the table of measured gateways.

All components of the algorithm run on the client side, i.e. on the network nodes and their execution can be roughly divided into two phases, the *bootstrapping phase* and the *periodic sensing phase*. When a node is activated it starts with an empty gateway performance table. The goal of the bootstrapping phase is, therefore, to identify the set of close neighbors and to receive their measurement results in order to fill the node's table. With this initial version of the table, the node can do a first gateway selection. After this has been done, the node enters the periodic sensing phase where it performs its own measurements (sensing) and exchanges measurement results with neighbor nodes.

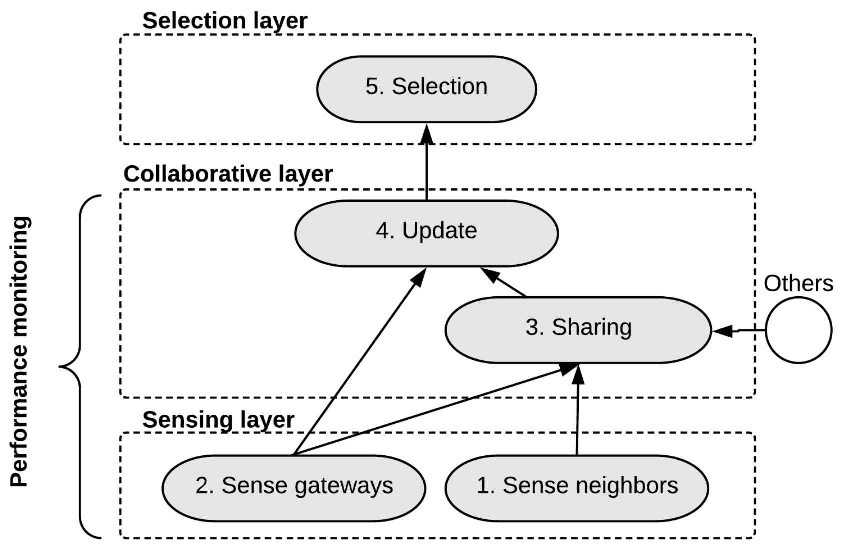


Figure 2.1: Layered structure of the proposed algorithm

2.3.2. Summary and future work

We looked at the informed gateway selection problem as a collaborative performance sensing within close neighbors, drastically reducing the information to share (a reduction factor from n to 2) to achieve good QoE at client nodes and overall fair distribution of the capacity of gateway nodes. Our selection algorithms are simple yet effective, that is infrastructure and technology agnostic and support incremental implementation (compatible with WCN networks that grow organically in the number of clients and number of gateways). Experiments show that the precision estimation of our proposed algorithm is high ($> 80\%$) throughout the experiments, which results in high-quality Internet access for clients. By utilizing the partial knowledge of the gateway performance information, the *collaborative-best* and *collaborative-fair* variants perform close to a brute force algorithm.

Our collaborative sensing algorithm is applicable to other domains such as homogeneous networks, sensor networks, IoT networks, distributed service placement, and collaborative congestion control.

Future work will explore adaptive sensing to further reduce the monitoring overhead depending on the number of close neighbor nodes, the number of gateway nodes by adjusting the measurement period accordingly. We plan to incorporate fault tolerance and capacity planning of gateway selection in an extended version of the algorithm. Further, we will test the scalability of our proposed algorithm as part of a new Cloudy release, in the real heterogeneous production network of guifi.net, with a large ratio of client nodes versus gateway nodes.

2.4. Towards IoT Service Deployments on Edge Community Network Microclouds

In a demo paper [8], we have shown that professional IoT services can be deployed on community edge microclouds. The purpose of this demonstration is to open the door for edge microclouds to serve as deployment environment that can be used to provide commercial services.

IoT services for personal devices and smart homes are typically provided by commercial solutions which are proprietary and closed. These services, however, give only limited power to the user on controlling the data and enabling services, which discourages acceptance by privacy-aware users.

Cloudy offers the opportunity to run IoT service instances as Docker containers, and there are Docker-based solutions to monitor and control network interactions with the inside and outside world[12]. These User-controlled infrastructures at the network edge will give to the user the benefit from being able to choose and control the operation of the most suitable service from different IoT service providers. Instead of running one or more proprietary hardware blackboxes, an open source platform supports different service providers and the user could choose to install only the service which satisfies her specific privacy requirements. Small and medium-sized enterprises (SMEs) could also increase the portfolio of services and offer tailored and granular IoT services to be run at customer premises on customer provided servers.

We conduct the demonstration on microclouds, which have been built with the Cloudy platform in the guifi.net community network. The demonstration is conducted from the perspective of an end user, who wishes to deploy professional IoT data management services in volunteer microclouds.

The open deployment environment is represented by the Cloudy distribution, installed by the users on their edge devices. Through the support services of Cloudy, a new edge node is integrated into the existing community microcloud. Cloudy foresees the deployment of applications by Docker containers.

To represent applications from professionally provided IoT services, we have chosen Hitachi Vantara's Pentaho open-source business intelligence (BI) suite [24], that provides comprehensive reporting, data processing, data integration, and data mining features. The Pentaho Data Integration (PDI) client is a desktop application that users can use to process and integrate data on their own machines, and design data workflows that they can upload to the repository in the Pentaho Server. Users can schedule jobs to run data workflows at regular intervals, and perform other advanced operations.

After the installation of Pentaho services in Cloudy, an end user connected to the Cloudy device finds the Docker-based Pentaho service available in the community cloud. The Pentaho service is discoverable in the microcloud because the provider has tagged it as a shared service. Therefore any participant of the community cloud will be able to find it. After logging in by accessing the service through the Cloudy Web interface, the user sees the home screen of the Pentaho service, where the user can manage stored files and scheduled jobs.

The work and demo paper [8] showed that the environment provided by Cloudy edge microclouds can also be used to deploy professional IoT applications. This is a key achievement since it supports the advancements of community networks to become an edge computing platform that can also be used by commercial service providers. The demos was conducted from a publicly available instance of Cloudy in Internet, which is also connected to the Cloudy instances within the guifi.net community network. We deployed one instance of the Pentaho Server as a publicly accessible Docker container on two Cloudy nodes. This way the view on the operation and management of a Cloudy node from a guifi.net user perspective was presented.

2.5. Lightweight service deployment in micro-clouds

In this line of work, which resulted in a journal paper [11], we detailed the initial evaluation reported in D3.3 on service placement mechanisms for Cloudy. The extended evaluation provides more precise results on the improvements and limitations of the proposed service deployment mechanism Bandwidth and Availability-aware Service Placement (BASP).

Previously, we motivated the need for bandwidth and availability-aware service placement in CN micro-cloud infrastructures. CNs provide a perfect scenario to deploy and use community services in a participatory manner. Earlier work done in CNs has focused on better ways to design the network to avoid hot spots and bottlenecks, but did not relate to schemes for network-aware placement of service instances. However, as services become more network-intensive, they can suffer from network congestion, even in well-provisioned clouds. In the case of CN micro-clouds, network awareness is even more critical due to the limited capacity of nodes and links, and an unpredictable network performance. Without a network-aware system for placing services, locations with

[12] Examples are: cAdvisor (Container Advisor) provides container users an understanding of the resource usage and performance characteristics of their running containers (https://github.com/google/cadvisor), Docker Universal Control Plane https://www.docker.com/products/docker-enterprise, among others. See for instance https://code-maze.com/top-docker-monitoring-tools/.

poor network paths may be chosen while locations with faster, more reliable paths remain unused, resulting ultimately in a poor user experience.

We proposed a low-complexity service placement heuristic called BASP to maximize the bandwidth allocation when deploying CN micro-clouds (already introduced in D3.3). We presented algorithmic details, analyzed its complexity, and performed an initial evaluation.

This year we carefully evaluated its performance with realistic settings. Our experimental results show that the BASP consistently outperforms the currently adopted random placement in guifi.net by 2x bandwidth gain. Moreover, as the number of services increases, the gain tends to increase accordingly.

Furthermore, we deployed our service placement algorithm in a real network segment of the Quick Mesh Project Sants and UPC (QMPSU) network, a CN deployed in an area of Barcelona, and quantified the performance and effects of our algorithm. We conducted our study on the case of a live video streaming service and Web 2.0 Service integrated through Cloudy distribution. Our real experimental results show that when using BASP heuristic algorithm, the video chunk loss in the peer side is decreased, worth a 37% reduction in the overall packet loss rate. When using the BASP with the Web 2.0 service, the client response times decreased up to an order of magnitude, which is a significant improvement. We see an opportunity for further improvement in looking into live service migration, i.e., the controller needs to decide which micro-cloud should perform the computation for a particular user, with the presence of user mobility and other dynamic changes in the network.

2.6. PiCasso: service deployment in information-centric networks and services

We explored an advanced mechanism for service deployment, PiCasso, initially developed iby the UMOBILE research project, in the context of information-centric networks and services. The concept of information-centric networks and services was considered by the guifi.net community useful and attractive to their needs. Further development and evaluation was done in the context of netCommons considering the software architecture of Cloudy, the results are reported in a journal paper under evaluation [12].

As participation in Community Mesh Network (CMN) networks is open, they grow organically, since new links are created every time a host is added. Because of this, the network presents a high degree of heterogeneity with respect to the devices and links used in the infrastructure and its management. This unique characteristic makes CMNs different from the conventional Internet Service Provider (ISP) networks as the topology is dynamically changing (as elaborated in D2.5 and D2.7 [25, 26]). Hence, the current software architectures and platforms are failing to capture the dynamics of the network and therefore they fail to deliver the satisfying Quality of Service (QoS) [20] [27].

The latest advances in lightweight virtualisation technologies (e.g., Docker, Unikernel), allows many developers to build local edge computing platform that could be used to deliver services within CMNs [28]. Despite delivering these lightweight services within a data centre is trivial, delivering them across intermittent connectivity of CMNs has a lot of challenges. As a matter of fact, most of the edge computing platforms like Cloudy [29] still rely on the host-centric communication that binds the connection to the fixed entity (i.e., an instance running on a single host, or several independent instances on different hosts but not coordinated). This approach could struggle for *service delivery* to bring service instances to the network edge where demand would be, as network wide connectivity might not be guaranteed. In addition to that, those platforms do not have specific strategies on the *service deployment* of a set of multiple instances that a service may require in CMN environments. This raises several questions: Which services should be delivered? When should they be delivered? What are the suitable criteria for node selection to host the service? Is network-aware placement enough to deliver satisfactory performance to CMN users? However, this is not trivial and requires an effective strategy to manage the service delivery in CMNs.

On the other hand, Information-Centric Networking (ICN) has recently emerged as a potential solution for delivering named contents. The ICN leverages in-network storage for caching, multi-party communication for replication and interaction models that decouple senders and receivers. Instead of using IP address for communication, ICN identifies a content by name and forwards a user request through name-based routing. This decouples the content from its origin address, where the content can be delivered from any host that currently has the content in its storage. Although ICN brings a lot of flexibility in terms of content delivery, the current ICN implementations are rather focused on the simple static content (e.g., short message, video file). In this regard, we argue that ICN should be extended to better support transporting at the service layer.

Our research results extend previous work [30] and [31] by focusing on two main interrelated research problems: service delivery and service deployment. The former refers to the process of delivery and instantiate a service instance (e.g., web server) from the service provider to the edge computing node. The latter is the logic that decides where and when to deploy the service instance regarding the service requirements, network status and available resources (e.g., CPU load, memory). In this context, we propose *PiCasso*, a lightweight edge computing platform that combines the lightweight virtualisation technologies and a novel Information-Centric Networking (ICN) to facilitate both service delivery and service deployment in challenging environment such as CMNs. We underpin PiCasso with Docker container-based service that can be seamlessly delivered, cached and deployed at the network edge, taking advantage of the experience with Cloudy in netCommons. The core of the PiCasso platform is the *decision* engine making a decision on where and when to deploy a service instance to satisfy the service requirements while considering the network status and available hardware resources. PiCasso introduces a new service abstraction layer using ICN to enable more flexibility in service delivery. Instead of hosting services in the fixed centralised location (e.g., service repository), PiCasso allows the edge devices obtaining service instance from the nearest caches by utilising inherent name-based routing and in-network caching capabilities of ICN. Furthermore, PiCasso is also integrated with a service controller and a full functional monitoring system to optimise the service deployment decision in CMNs.

Specifically, our key contributions are summarized as follows

- First, based on measurements in guifi.net, we describe the design of PiCasso, a multi-access edge computing platform which deploys QoS-sensitive services at the network edge. The decision engine of Picasso selects the appropriate nodes for service instantiation based on constraints observed in our guifi.net measurements (network bandwidth, available hardware resources and network topology). The HArdware and NETwork Resources (HANET) decision engine algorithm uses the state of the underlying community network to optimize the service deployment.
- Second, we utilise the ICN principles in the architecture of PiCasso in order to enable more flexibility in the delivery of named data objects.

This was the first effort in deploying an ICN-service in a running wireless CMN such as guifi.net.

2.6.1. Key Observations

Here are some observations that we have derived from the measurements in the guifi-Sants mesh network.

A lack of smart service platforms: Despite achieving the sharing of bandwidth, guifi-Sants and guifi.net in general, have not been able to widely extend the sharing of ubiquitous cloud services, such as private data storage and backup, instant messaging, media sharing, social networks etc., which is a common practice in today's Internet through cloud computing. There have been efforts to develop and promote different services and applications from within community networks through community network micro-clouds [29] but without massive adoption. Further, a growing number of micro-cloud services desire computational tasks to be located nearby users. They include needs for lower latency, a better-user experience and efficient use of network bandwidth.

Variability in topology and change in capacity load: The guifi-Sants network is highly dynamic and diverse due to many reasons, e.g., its community nature in an urban area; its decentralized organic growth with extensive diversity in the technological choices for hardware, wireless media, link protocols, channels, routing protocols etc.; its mesh topology etc. The current network deployment model is based on geographic singularities rather than QoS. The network is not scale-free. The topology is organic and different with respect to conventional ISP networks. This implies that a solution (i.e., algorithm) that works in a certain topology might not work in another one.

Non-uniform resource distribution: The resources are not uniformly distributed in the network. Wireless links are with asymmetric quality for services (25% of the links have a deviation higher than 40%). There is a highly skewed bandwidth, traffic and latency distribution. The symmetry of the links, an assumption often used in the literature of wireless mesh networks, is not very realistic for our case and algorithms (heuristics) unquestionably need to take this into account.

We built on these insights, and designed PiCasso, a low-cost edge computing platform that sits at the "extreme" edge of the wireless network. The system was developed in the UMOBILE project, and netCommons applied it to the Cloudy architecture and evaluated it the guifi.net network. We presented a detailed evaluation assuming services are run in Service Execution Gateways, that are equivalent to Cloudy nodes. We show how PiCasso achieves more efficient use of network bandwidth at low network cost in guifi-Sants network using its ICN capabilities.

2.6.2. PiCasso: Multi-Access Lightweight Edge Computing Platform

PiCasso is implemented based on the service and access abstraction where lightweight virtualisation services are delivered through ICN. There are several ICN implementations [32, 33, 34] that have been proposed during the past decade. Among those implementations, Named Data Networking (NDN) is the most suitable candidate for PiCasso as it uses a simple stateful forwarding plane to utilise the distributed in-network caching without any control entity. Thus, we developed PiCasso that extends NDN protocol stack to support service delivery and service deployment in CMNs.

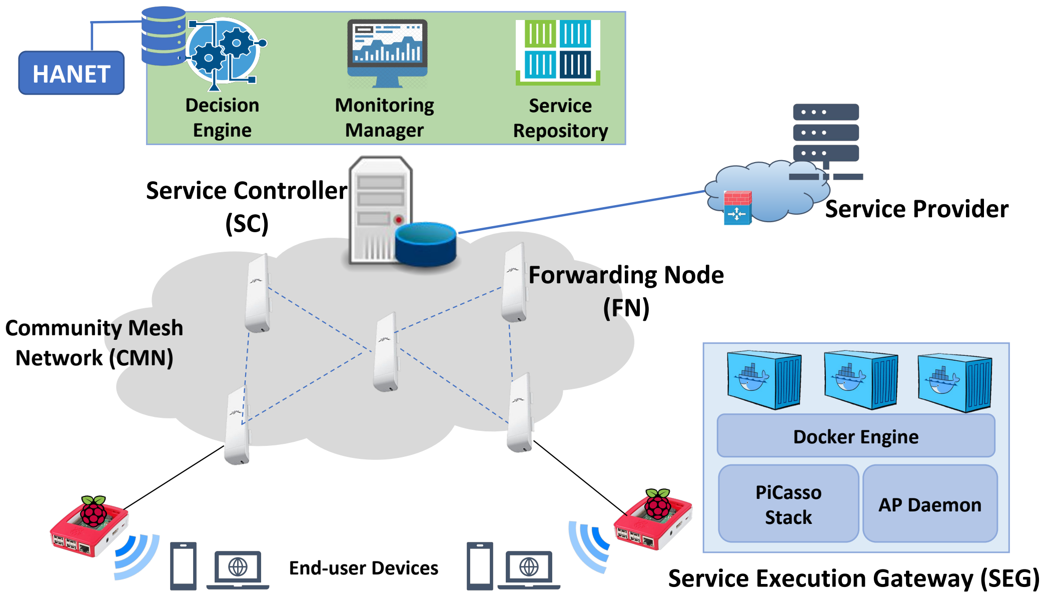


Figure 2.2: The overview of the PiCasso platform. Main components of Picasso platforms are shown: Service Controller (SC), Service Execution Gateway (SEG) and Forwarding Node (FN).

PiCasso is a lightweight edge platform that can rapidly deliver services to end users at the edge even though the network connectivity is intermittent. PiCasso relies on service and access abstraction where lightweight virtualisation services are delivered through the ICN. This approach that does not rely on underlying host-centric networking model, decouples the service from it's physical location by taking advantage of the naming and content caching that could be used to make intelligent forwarding decisions and publish/subscribe communication primitives that allow asynchronous communications, etc.

The overview of PiCasso platform is presented in Fig. 2.2. The key entity of PiCasso is referred to as Service Controller (SC) that periodically observes (i.e., monitors) the network topology and resource consumption of potential nodes for the service deployment. In our model, we assume that the service providers upload their services to a service repository inside the SC before distributing to the network edge. To maintain a good QoS and overcome the network connectivity problems, SC augments the monitoring data along with service deployment algorithms to decide where and when to place the services. The Service Execution Gateway (SEG) provides a virtualisation capability to run a service instance at the network edge (e.g., users home). In PiCasso, we use Docker, a container-based virtualisation to build lightweight services and deploy across the SEGs. Each SEG is also equipped with the access point daemon (e.g., hostapd[13]) to act as the point of attachment for the end-users to access the services via WiFi connection. The *Forwarding Node (FN)* Fig. 2.2 is responsible for forwarding the requests towards the original content source or nearby caches. Each FN is equipped with a storage while dynamically caching the content chunks that flow through it. Notice that, FN does not necessary need to execute the services.

2.6.3. Architecture of PiCasso

Current implementations of edge computing still rely on a centralized approach with large amount of traffic spread out over the network. Streams of requests are sent from edge devices to the data center instead of fetching the contents and services from the nearest nodes. NDN (Named Data Networking) is one of the ICN projects which instead of using IP address for communication, directly addresses the contents by name regardless of physical location. As a matter of fact in NDN, a piece of content or service can be stored or cached at multiple locations. NDN uses Name based Forwarding (NFD) where the routing should be done dynamically and effectively to fetch the desired contents of services from the best location. NDN naturally fits with the nature of CMNs allowing nodes located far away and have intermittent connectivity to retrieve the content or services directly from the nearest caches.

Currently, PiCasso is written in Python and implemented on top of NDN protocol stack [32] and Docker technology [35]. The main function blocks of PiCasso's architecture are presented in Fig. 2.3 and are the following ones.

NFD Forwarding plane sits between application and transport layer while looking at the content names and opportunistically forwarding the requests to an appropriate network interface. It creates an ICN overlay to support name-based routing over the network. In PiCasso, we have also extended the NDN protocol stack by introducing a Delay Tolerant Network (DTN) face to facilitate operation in challenge network environment like post-disaster scenario. This new face communicates with an underlying DTN implementation that handles intermittence by encapsulating Interest and Data packets into a DTN bundle. The details of implementation and evaluation can be found in [36]. We integrate the NFD forwarding plane to PiCasso architecture through a Python wrapper of NDN APIs, called PyNDN[14].

Service Execution runs on the SEG and has major functionality as follows: registers the SEG to the service controller, receives push command to instantiate and terminate services dynamically regarding the decision of service deployment. This module uses docker-py[15], a Python wrapper for Docker to expose the controlling messages to Docker engine.

Monitoring Agent is responsible for measuring the current status of underlying hardware resources such as current memory usage, CPU utilisation, CPU load etc, and reporting this data to SC. Further, it associates with Docker engine to report the status of running containers (e.g., container names, number of running containers) and resource consumption inside each container (e.g., CPU and memory usage).

[13] https://wiki.gentoo.org/wiki/Hostapd
[14] https://github.com/named-data/PyNDN2
[15] https://github.com/docker/docker-py

Decision Engine (DE) or orchestrator is responsible for selecting an appropriate SEG node for service instantiation based on constraints such as available hardware resources, QoS and network topology. DE has access to algorithm repository that can execute to make decisions on deployment of service instances. The service deployment algorithms can be dynamically updated regarding different deployment scenarios and service requirements.

Service Repository is a repository for storing dockerized compressed service images. Images of the services are stored augmented with specification about service deployment in the form of JavaScript Object Notation (JSON) format. Our implementation allows the third party service providers to upload their service along with a deployment description augmented with specifications and QoS requirements.

Monitoring Manager periodically collects the monitoring data from each SEG and stores in the database (Monitoring DB). It is implemented based on a time series database, called InfluxDB[16]. We also implemented the dashboard for monitoring system using Grafana[17] to visualise time series data for SEG's measurements and application analytics. Fig. 2.4 shows the user interface of PiCasso monitoring dashboard.

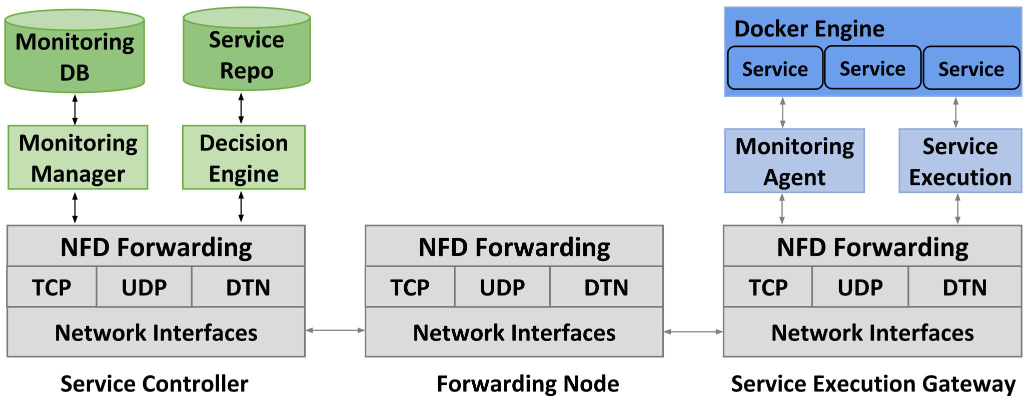


Figure 2.3: PiCasso's function blocks.

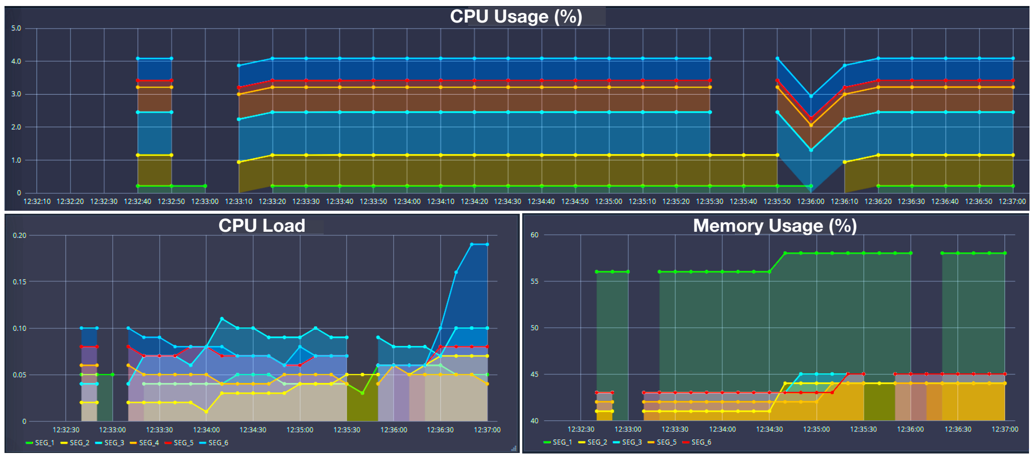


Figure 2.4: PiCasso monitoring dashboard

[16]https://github.com/influxdata/influxdb-python
[17]https://grafana.com/

2.6.4. Discussion and implications for Cloudy

Local Service Ecosystem: The PiCasso edge computing platform, combines a set of NDN tools that simplify and optimize the delivery of content and services to clients, a kind of local Content Distribution Network (CDN), ideally with presence of PiCasso support in the first hop, the access point. The result is that the indirection infrastructure offloads a majority of requests, decoupling content and service from the volume of demand. This is encouraging to CMN users to participate as active contributors, ultimately creating an ecosystem of local services, that do not result in high load in servers and nearby network links. PiCasso packages together different cloud services and content at near minimal network and server cost to end users. This would allow Cloudy instances to coordinate and distribute services across an overlay of Cloudy servers. However, the challenge for Picasso remains to analyze and optimize the delivery of different kinds of services when using the ICN paradigm. For instance, one of the services to consider in our future work is live video streaming, where content names/identifiers could be used to route requests to the right streaming servers, and take advantage of the application semantics (streaming content) to optimize the distribution and delivery of this particular content.

Deployment benefits (transparency): The Picasso platform is easy to deploy thanks to the plug-and-play feature of nodes. The adoption of the Picasso platform requires minimal changes in the Cloudy architecture or network configuration since nodes are added via plug-and-play. Moreover, PiCasso nodes are able to discover the closest node and dynamically retrieve the service image from the nearest cache. Hence, content can be transparently delivered, cached and deployed at the network edge, at just one network hop from the client. For that, the Serf service in Cloudy can provide that function, as it is able to sort service announcements in terms of latency.

Traffic reduction benefits (Operator gain): Network bandwidth is crucial in guifi-Sants network since it highly fluctuates. The use of PiCasso results in significant traffic reduction in CMNs from the benefits of in-network caching and name-based routing. These functions assists PiCasso to reduce the service delivery cost as well as the network traffic during the service deployment (42% reduction in terms of traffic comparing to a host-centric solution). However, regarding the results from our deployment trial, the traffic consumption of PiCasso is not yet optimized. As a matter of fact, NDN strictly requires the collaborative effort in order to achieve the maximum bandwidth reduction from in-network caching capability. To summarize, in a mid-size network (e.g., 80 nodes) as guifi-Sants it is, if we can deploy the PiCasso in critical nodes of the network, the expected traffic reduction will be similar in percentage, but given the larger size of the network, having content still one hop away from users results in a bigger difference in terms of intra-network traffic that reduces network congestion, in addition to an increase of controller (server) load that grows more slowly than the size of the network.

2.7. Exploring blockchain for economically sustainable wireless mesh networks

In this line of work we looked at mechanisms to coordinate data and value flows towards sustainability using Blockchain technology. We look at consensus (Proof of Authority (PoA)), smart contracts, and we evaluate blockchain platforms. A scientific paper derived from this work [14] is currently under revision for a journal publication. This work is in collaboration with industry (AmmbrTech SRL). Decentralization, in the form mesh networking and blockchain, two promising technologies both at the root of Community Networks (CNs) and netCommons, are starting to be considered also in tht mainstream telecommunications industry, specially for 5G management and cost reduction. Mesh networking allows wider low cost Internet access while blockchain enables complete transparency and accountability for investments and revenue or other forms of economic compensations from sharing of network traffic, content and services. Crowdsourcing network coverage combined with crowdfunding costs can create sustainable yet decentralized Internet access infrastructures, where every participant can invest in resources and Internet access, to make the network economically sustainable. While mesh networks and mesh routing protocols enable self-organized networks that expand organically, cryptocurrencies and smart contracts enable the economic coordination among network providers and consumers. We explore and evaluate two existing blockchain software stacks, Hyper-Ledger Fabric (HLF) and Ethereum 'geth' with PoA, deployed in a real city-wide production mesh network, and in a centralized laboratory network. We quantify the performance, bottlenecks and identify the current limitations and opportunities for improvement to serve the needs of wireless mesh networks.

2.7.1. The context and analysis

One of the open challenges of peer-to-peer socio-technical structures are is trust between peers and how to ensure the economic sustainability of this collective effort and the balance between contribution and consumption [37].

As an example scenario and mechanism for economic sustainability is the economic *compensation system* used in guifi.net [37]. An answer to the lack of incentives to invest in network infrastructure, it was introduced in 2011 as a cost sharing mechanism. The idea of the compensation system is to balance between total resource contribution and its consumption. The economic cost of any contribution and consumption of network resources by each participant in a given locality are recorded. The overall result is a zero-sum computed periodically, from monthly to quarterly, where the participants with over-consumption or negative balances have to compensate those with over-contribution or positive balances.

Currently the above described economic compensation system is done manually: each participant declares its costs and consumption and then the guifi.net foundation[18] validates this claim by cross checking it with their own network traffic measurement data and network inventory, according to the agreed list of standard costs. Any disparities between these two records are flagged, clarified or raised to a conflict resolution mechanism. There is, however, room for error or intentional false or exaggerated claims put forward by a participant, the recorded data being tampered with, or simply mistrust among the parties. The correct application of the compensation system is critical for the economical sustainability of the network, ensuring its proper operation, as well as future investments. Therefore, we argue that there is a need for an automated system where participants can trust that the consumption of resources is being accounted in a fair manner, and that these calculations and money transfers are automated to avoid the cost, delays, errors and potential mistrust from manual accounting and external payments.

Blockchain technologies seem to be apt to make the peer-to-peer nature of access networks trusted and economically sustainable. A Blockchain is an immutable and distributed data storage without the provision of retrospective mutation in data records. However, most blockchain networks are open and public (permissionless) that encourage the users to be anonymous [38]. This implies that anyone, without revealing their true identity, can be part of such a network and make transactions with another similarly anonymous peer of the network.

In the perspective of community networks such as guifi.net, however, every participant who joins the network to contribute and benefit from the infrastructure must first register its identity and the identity of the resources that it contributes to the wider pool. This is particularly needed so that any malicious entity, such as hidden nodes in guifi.net used by other ISPs, can be filtered out. Because of such registration process one also needs an efficient identity mechanism on top of blockchain's immutable record keeping. *Permissioned blockhains* are part of such solutions, mostly envisioned for business networks where there is often a stringent requirement of *know your customer* in addition to keeping the intra- and inter-business transactions confidential.

In this study, we extend our previous work [17] by exploring the plausibility of combining decentralized access networks with a permissioned blockchain running on servers inside the access network, that would result in a model for economically self-sustainable decentralized mesh access networks, guaranteeing trust among participants, allowing economic profitability, and enabling at the same time easier Internet connectivity. We

[18] https://fundacio.guifi.net/Foundation

study the viability of such an approach, by evaluating two of the most prominent platforms for building local blockchain applications. These platforms are Hyperledger Fabric (HLF)[19], an industry-oriented modular, and permissioned distributed ledger and Ethereum[20], a general-purpose, business-oriented nonetheless, platform.

We deploy the Hyperledger Fabric and Ethereum platform in a centralized network in our laboratory, as well as in a decentralized production wireless mesh network that is part of guifi.net. Our key contributions towards the adoption of these technologies in CNs can be summarized as follows:

- First, we analyze the performance of both platforms in terms of metrics such as transaction latency, Central Processing Unit (CPU) and memory utilization of Hyperledger Fabric and Ethereum components. To the best of our knowledge, this is the first Hyperledger Fabric and Ethereum deployment made in a production wireless mesh network. Our results show that both Hyperledger Fabric and geth Ethereum network can be deployed on even resource constrained devices like Raspberry Pi 3 boards or router boards with limited computational capability. Both the blockchain software stacks perform well without saturation and much delays for a moderate load of up to 100 transactions fired in the network at a time. In Hyperledger Fabric, our measurements reveal that endorsers are the bottleneck and care has to be taken in designing endorsement policy for scaling the network. In case of Ethereum, our results show that a there is a limit on the number of requests a node can support and can only be scaled vertically i.e., by increasing computational capability of serving node
- Second, driven by the findings in a mesh network, we propose a placement scheme for Hyperledger Fabric and Ethereum components that optimizes the performance of the blockchain components in mesh networks.

2.7.2. Blockchain: The underpinning Technology

A blockchain is an *append-only* immutable data structure. Its most famous example is the Bitcoin cryptocurrency network [38]. In Bitcoin the blockchain is used to enable trust in financial transactions among different non-trusting parties in a pure peer-to-peer fashion without the need for going through a third financial party like e.g., a bank. Such trust is provided in terms of immutability of blockchain's data structure. Each *block* in blockchain contains information that is immutable. The immutability aspect is rendered true by including the hash of all the contents of a block into the next block which also chains the blocks together. Tampering with one block disturbs the contents of all the following blocks in the chain. Each block in the chain is appended after a *consensus* is reached among all the peers of the network. The same version of a blockchain is stored in a distributed manner at all the peers of the network. That is why it is sometimes referred to as *distributed ledger* as well.

Two blockchain platforms are chosen for evaluation in wireless mesh networks namely Hyperledger Fabric and Ethereum, due to their popularity and potential to be used in different applications.

2.7.3. Permissionless vs Permissioned, Public vs Private

Bitcoin [38] and Ethereum[21] [39], as various other blockchains, are considered as *permissionless*, meaning that anyone has "write" access to the blockchain. As a result anyone can be a part of the network, mining and performing transactions with other parties. The consensus in such an open environment is tackled with algorithms like the Proof of Work (PoW) protocol. Some degree of anonymity is also at the heart of such platforms. A user (or in general an entity) usually uses the hash of its public key as its identifier as opposed to using its real-world credentials.

[19] https://www.hyperledger.org/projects/fabric
[20] https://www.ethereum.org/
[21] https://ethereum.org/

In the aspect of "write" openness, *permissioned blockchains* are in sharp contrast with public blockchains which we discuss next. Permissioned blockchains, a concept particularly popularized by the Linux Foundation's Hyperledger, are usually considered for business applications. In such applications the identity of users, in addition to trusted and immutable data storage, is also important such as the stringent requirement of *know your customers* for many businesses. Hyperledger tries to leverage the best of both worlds by implementing a cryptographic *membership service* on top of blockchain's trusted, immutable, and distributed record keeping.

Another categorization can be done based on the openness of reading from the blockchain. In the case where a blockchain exposes its data publicly it is characterized as *public*. On the other hand, blockchains that prohibit access to its data are called as *private*.

In our study, the requirement of both users' identity and trusted record keeping is of paramount importance and that is why we decided to conduct our study using private permissioned blockchains. Hyperledger Fabric fulfills by default these properties. On the other hand, while Ethereum is not primarily destined to serve this purposes, it can also be used as private permissioned blockchain. Nevertheless, executing resource-full consensus algorithms in a permissioned environment where the participants are known has no application except experimentation with the protocols themselves. On the other hand, some protocols, like Ethereum, offer inexpensive consensus algorithms like PoA that are ideal for a private permissioned instances.

The submitted paper [14] provides a detailed description of the evaluation setup and the evaluation results. Here we jump to the lessons learned.

2.7.4. Discussion

Hyperledger Fabric: As we observed in our experiments, in terms of resource consumption, the endorser nodes can prove to be a bottleneck. We believe that this bottleneck is because of the execution of an additional chaincode container at each endorsing node. In our current study we only considered one endorser node to study the resource utilization with a simple endorsement policy encoded in the corresponding chaincode. It might get more complicated when we consider more than one endorser, and more sophisticated endorsement policies. However, as discussed in the paper, if done right it can actually improve performance. In addition to this, the actual distribution of endorsing peers in a production network, such as QMPSU, might also affect the network performance (both in terms of CPU utilization and transaction latency). Therefore we advise caution when designing an endorsing policy that is also cognizant of the underlying network infrastructure (i.e, topology, capacity, performance, etc), especially in the resource constrained nature of CMNs. A deployment strategy and an apt endorsement policy balancing the load on various endorsers in the network can improve the performance of the blockchain network and allow scaling of the blockchain network without forming a bottleneck. As far as the orderers are concerned, horizontal scaling by adding more nodes is possible, nevertheless, this would need some sort of mechanism for syncing between instances. For instance, it is possible to have multiple instances of ordering service nodes all connected to a single fault tolerant service (such as the Open source Kafka application) that would do the ordering (crash fault tolerant).

Ethereum: The results concerning Ethereum show that it can be used successfully as private permissioned blockchain in a mesh environment, using PoA consensus. Nonetheless, there are various parameters to be adjusted and bottlenecks that need to be discussed. Unlike HLF, in Ethereum there is no clear horizontal scaling pattern. While having a lot of sealers could balance the incoming transactions, the transaction throughput is largely affected by the hardware resources like CPU and memory of the nodes who accept the transactions and less affected by the number of nodes. This, depending on the frequency of transactions generated, can be a significant issue for mesh like environments, since the hardware used is usually low-power/low-cost devices. Moreover, the broadcasting of the pending transactions between the sealers can become problematic over non-stable mesh connections, especially between remote nodes, or nodes connected with lossy links. This situation could also deteriorate by an increased number of nodes and small blocktimes, leading to higher frequency and higher number of message exchanges between the sealers. On the other hand, these effects could be moderated by utilising smart placement algorithms like Bandwidth and Availability-aware Service Placement (BASP), which would play a significant role in avoiding network saturation, by placing the sealers in locations that would minimise the overhead of the blockchain. Finally, while we deploy multiple clones of one sealer, other approaches are possible, like having multiple sealer accounts, considering that a minimum of $N/2 + 1/$ instances of them are always available [40].

2.7.5. Conclusion

The missing ingredient for widespread adoption of decentralized access networks (such as community mesh access networks) has always been the issue of economic sustainability. In this work, as explained in [14], we tackle the issue of addressing trustworthy economic sustainability by proposing the need for an economic substrate built using blockchain that can keep a record of the transactions related to the contributions (of nodes, links, Internet gateways, maintenance), consumption of communication network's resources as its economic compensation in a decentralized and trusted manner. The evaluation of the Hyperledger Fabric and Ethereum blockchain deployment in a centralized network, i.e., laboratory, and a decentralized network, i.e., in a real production mesh network, gives an understanding of the performance, overhead, influence of the underlying network, and limitations of the two platforms. The results show critical aspects that can be optimized in a Hyperledger Fabric and an Ethereum deployment, in the perspective of decentralized networks, where several components can prove to be bottlenecks and therefore put a limiting effect on the rate of economic transactions in a mesh network. Future work will expand the evaluation to a wider range of hardware and network configurations considering real and synthetic transaction traces. We will also consider the influence of the execution of non-trivial smart contracts, with a more realistic design of an endorsement policy (chaincode).

2.8. Exploring the Collaborative Governance of Decentralized Edge Microclouds with Blockchain-based Distributed Ledgers

We explored blockchains as support service in an approach to extend microclouds with an economic model. This work was published in [15] and is summarized in the following.

Since current attempts to build microclouds lack a collaborative governance system to operate successfully, we discuss the opportunity to use blockchain technologies to implement key services to enable the decentralized collaborative governance of microclouds. A multi-agent approach could further contribute to improve the efficiency in the decision making in the collaborative governance service.

2.8.1. Architectural design and implementation options

The architectural design we present builds upon the design of Cloudy. For this we refine the available support services and aim to extend Cloudy with collaborative governance services.

The architecture is organized into the three layers front end, governance services and support services (Figure 2.5).

The front end layer provide a user-friendly Web interface to perform operations on the community cloud node. The Application Programming Interface (API) aims to support automated node-to-node operation. The Web interface available in Cloudy is the main tool for the user to search and deploy services.

The governance service layer contains components for accounting of usage, resources and participation, as well as operations which are fed from the processing of this information. For the implementation, distributed ledgers seem to be a suitable option specifically for the following two components:

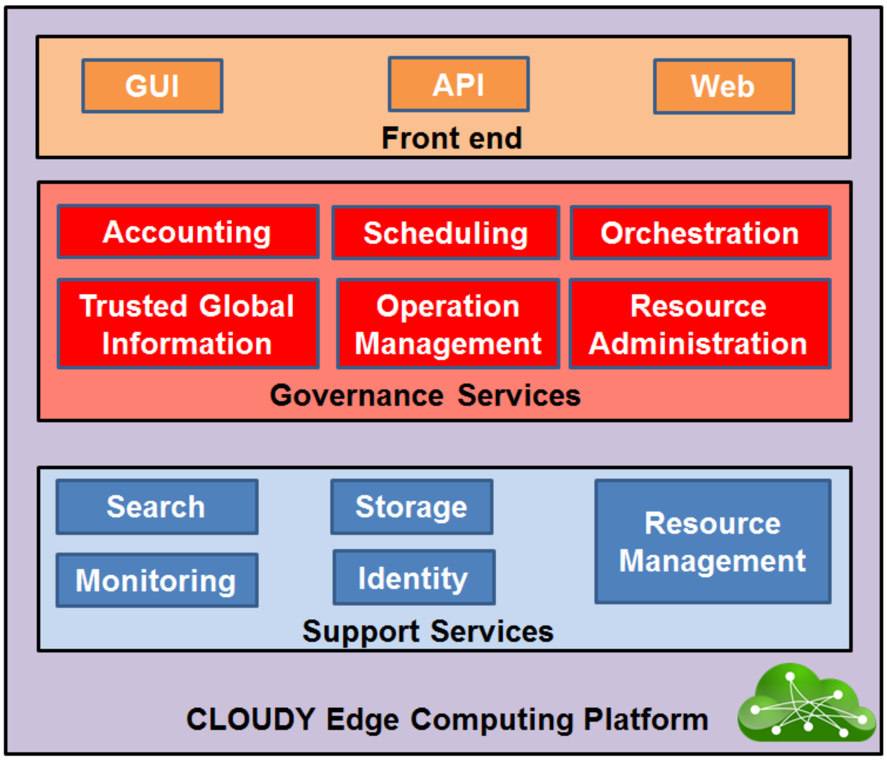


Figure 2.5: Technical layers for the collaborative governance of edge microclouds.

Accounting: Service usage and service contributions of community cloud members should be registered by the system in order for the community to be able to provide feedback and potentially rewards. The requirements for this component include that the information provided to take decisions is trusted by all participants. The component receives input from the monitoring component (support service layer). The workload of the current accounting data can be considered as low, since changes in the service offers in the Cloudy community clouds are not very frequent. The computational requirements to perform this trusted accounting on a device should be low in order for the service execution to be transparent for the owner of a node. A distributed ledger offered by a permissioned blockchain platform could be considered as a solution to implement this component.

Trusted Global Information: The overall system status (e.g., computational resource usage, service usage and offer) should be registered in a distributed ledger in order to provide information to the participants. This feedback could serve for the community to take informed decisions. It could also provide information to feed into social network channels to document usage and benefits.

With regards to the support service layer, the components are available in Cloudy, but developments in blockchain technology may require adapting them for better integration. For instance, the identity service could leverage ideas of Sovrin[22] to improve the identity management in guifi. The storage service could be extended by the InterPlanetary File System (IPFS)[23]. An IPFS storage layer combined with a blockchain platform could reduce the cost of smart contract execution on the blockchain.

[22] Sovrin: Identity For All https://sovrin.org/
[23] https://ipfs.io/

2.8.2. The multi-agent approach to enhance the governance of Cloudy microclouds

A community microcloud is a decentralized system with multiple owners of individual nodes which donate resources to a common resource pool that forms the microcloud infrastructure. The collaborative governance system aims to implement a set of services to achieve a higher attractiveness of microclouds by improved performance. We can identify several options for contributions by a Multi-Agent System (MAS):

1. Autonomous agents acting in behalf of the Cloudy providers and users: Currently, the preferences of a Cloudy node owner on the node operation are not delegated to a software agent. The need for multi-tenancy of a Cloudy node, however, is already recognized and individual profiles to represent the specific preferences of a user could be created, based on which a software agent could interact with other agents. Research on user models and how they are integrated in multi-agent negotiation was done for instance in [41]. It was shown that agents contributed to a higher fulfillment of the user preferences.
2. Multi-agent system to support the cloud: In [42] the potential of agents to improve cloud performance was indicated. In microclouds, more intelligent decisions to determine resource allocations by MAS could be very relevant. Resources in a microcloud are lightweight and heterogeneous. Therefore, appropriate decisions are needed to achieve an efficient and performing system.
3. End-user friendliness: There are several roles in community microclouds. The actors participating in microclouds can take roles which are similar to those of data center clouds. Node owners can act as service providers, e.g. offer an application as SaaS. At the same time, node owners can be consumers of services offered by other nodes. The microcloud infrastructure itself, however, does not have a single owner and consist of those resources and services which are donated to the community. In [43] the understanding of QoE as a multiple dimensional construct was presented. This view to manage QoE could also be considered if applicable to the conditions of microclouds.

2.8.3. Concluding remarks on microclouds

Community microclouds currently lack a collaborative governance layer. Several services of this governance layer could be implemented by a trusted and immutable distributed ledger, which blockchain technology can offer. As such, blockchain technology is suggested as technical enabler for building the collaborative governance services.

Operating the collaborative governance service for a distributed and decentralized computing infrastructure at the network edge provides specific challenges, such as fulfilling the multi-tenancy purpose of the microcloud nodes, to offer suitable performance on lightweight computing devices, and to be end user friendly. A multi-agent system has the potential to improve the efficiency of the microcloud governance service by determining through interactions among agents more appropriate decisions with regards to individual and global performance goals.

2.9. Comments from members of the Cloudy community

As a final close to this chapter we report some feedback we obtained from the community using and developing Cloudy as volunteers and active members of the CN. These are the outcome of rather informal interaction with users fundamentally from guiifi.net, where we shared the following questions obtaining normally an e-mail answer.

Q1: *(My role, participation)* Were you, or someone in your community, involved in the development or use of open source software realized in the netCommons project? Please briefly describe your experience.

Q2: *(Adde value)* What is the added value that such software brought to your community?

Q3: *(Future)* Are you planning to keep using the netCommons software after the end of the project?

Q4: *(Methodology)* Where you involved in the use of the participatory methodology developed in the netCommons project? If yes, was it useful to the co-creation and use of applications in your community network? If not, briefly describe why.

Q5: *(Personal data)* Can we include your comments to a project report? If so, nominal or anonymous?

The questions were posed both in Catalan and English, leaving people the choice of the language they prefer. The answers we obtained were either in Catalan or in English. In case of Catalan we report the original text in italic, and the English translation for easy of access. We received four answers, we label **A** to **D**, indicating only the CN they come from and not the name, even if they did not object to the use of a nominative answer, but we don't think this adds anything to the result.

Answers

A: Ribaguifi, Santiago

Q1: *Lo hemos estado utilizando en la red de guifi.net de la Ribagorza. Es la máquina que aloja nuestro servidor de gráficas, está ubicado en el local social de Eresué.*

We have been using it in the guifi.net network of the Ribagorza. It is the machine that hosts our graphics server, it is located in the social premises in Eresué.

Además del servidor de gráficas, estuvimos probando a instalar un servicio de streaming de audio (CherryMusic). Funcionó bien, aunque lo usamos sólo un par de meses.

In addition to the graphics server, we were trying to install an audio streaming service (CherryMusic). It worked well, although we only used it for a couple of months.

Q2: *Nos ha permitido tener una máquina monitorizando la red, de bajo consumo y que gracias al Cloudy la configuramos con unos pocos clicks.*

It has allowed us to have a machine monitoring the network, with low consumption and thanks to the Cloudy we configure it with a few clicks.

Q3: *Piensas seguir utilizándolo? Sí. Tuvimos problemas al actualizar a Debian 9, así que entendemos que habría que revisar la compatibilidad con nuevas versiones. También vemos útil añadir un servicio de VPN (tinc, openvpn...) que permita acceder de manera remota (y segura) a la máquina cuando estás fuera de la isla de guifi.*

Do you plan to continue using it? Yes. We had problems updating to Debian 9, so we understand that compatibility with new versions should be checked. We also find it useful to add a VPN service (tinc, openvpn ...) that allows remote (and secure) access to the machine when you are away from the guifi island.

Q4: This group is remote, not involved in the participatory methodology.

Q5: *Sí, puede ser de forma nominal. Como os resulte más útil/práctico.*

Yes, it can be nominal. As you find it more useful/practical.

B: guifi.net, Barcelona

Q1: *Per la meva part he usat els serveis de guifi i el tema de containers per poder aixecar serveis.*

For my part I have used the guifi services and the topic of containers to be able to launch services.

Q2: *M'ha anat ve una interficie senzilla i facil d'usar. Ha estat un plug & play. Molt senzill configurar els serveis associats a guifi.*

It was useful to have an easy to use interface. It has been a plug & play. Very easy to configure the services associated with guifi.

Q3: *Penso seguir usant-lo.*

I intend to continue using it.

Q4: Semi-remote participant, not involved in the participatory methodology.

Q5: No reply.

C: guifi.net, Barcelona

Q1: I am part of a community in which the Cloudy software is being used to fulfill different technical networking needs and I'm a contributor to the Cloudy codebase.

Q2: Ease of use, by means of a simple graphical user interface, for instance to automate network monitoring tasks, and the possibility to share different tools and services within the community by means of containerized applications.

Q3: Yes, and possibly extending it by contributing new features as new needs emerge or novel tools are available.

Q4: Briefly, but yes. Communities of practice often fall into routine operation and end up approaching new challenges with old methodologies. Assistance from external agents (e.g., academia) helps breaking up with old, inefficient habits and dynamics, and opens new spaces for decision making and collaboration inside the community itself.

Q5: No reply.

D: guifi.net, Barcelona

Q1: I've participated in the Cloudy project both as developer and user.

Q2:

1. easiness to learn about services offered by others (up-to-date services lists)
2. simplicity of installation, configuration and operation of critical applications for the operation of our community network (guifi.net)
3. possibility to explore new applications using a single integrated framework

Q3: Yes. Cloudy fulfils many of may needs nicely, so no need to stop either using it or supporting it.

Q4: Yes. Having the guidance of experts in participatory research has helped a lot to clearly define the objectives and the steps to take to achieve them. I very much appreciate the collaboration established between the academia and the community and I hope it has been a win-win relationship. At least it has been very positive for the community.

Q5: No reply.

3. Developments and Use of the PeerStreamer Application

This chapter describes the development of the PeerStreamer application and its video conferencing version (which we named PartyHub), which was explicitly part of the DoA, together with "side" activities that emerged as requests from the ninux community or from interactions with the "global" community of people involved in CNs, as it happened also in Task 2.4 relative to monitoring instruments for CNs. Together with the development choices we report on the dissemination process, on the efforts we did to introduce its use in the ninux community and on the feedback we received from early adopters.

All released source code can be found in the `PeerStreamer-ng`, `psng-pyserf` and `PeerStreamer-Docker` repositories in the project Github account[1].

Since M24 we further extended PeerStreamer-ng to support video-conferencing in community networks. Besides this implementation task, the work has focused on disseminating the platform functionalities, provide installation and usage documentation and get in contact with community network users. The communication with community users has exploited several means; we created an IRC channel (#peerstreamer), we gave demo presentation during meetings ("Wireless Battle of the Mesh v11") and the Sarantaporo public event "Building the community in Community Networks") and we joined community-centered Telegram groups.

Communication was not only dissemination; we have promoted and sustained a two-way communication enriching the development process of precious insights and feedback. Feedback have consisted of bug reports, new feature requests and use-case suggestions.

One of the most demanded feature was the availability of a packaged version of PeerStreamer-ng for Ubuntu/Debian operating systems, which is now available on PeerStreamer-ng website. Debian-based systems are quite common in community network devices and we have decided to support this packaging to further ease the installation on those systems. With Ubuntu/Debian packaging and the Cloudy bundling community network users can easily install and run our software.

Along with user-driven development, implementation effort between M24 and M36 has manly focused on extending PeerStreamer-ng streaming capabilities to support video conferencing and deploying a web-based many-to-many streaming platform, called *PartyHub*. While PeerStreamer-ng focuses on webcasting of events with a one-to-many approach, PartyHub instead focuses on small groups (2-4) or people interacting with each other. WebRTC has become the ultimate choice for our project as it has proven to be widely supported, stable and nicely integrating with our codebase. PartyHub leverages PeerStreamer-ng capabilities and WebRTC support to provide a video conferencing tool designed and implemented for community networks.

The rest of this chapter covers our dissemination efforts and feedback received (Sec. 3.1), the development of PartyHub (Sec. 3.2) and the use and adoption in the ninux network (Sec. 3.3).

As already mentioned we also expanded the activity beyond PeerStreamer, and Sec. 3.4 reports on two more software development activities that were carried on in this year. Even if they were not part of the specific objective of the work packages they are integral part of our development efforts together with the ninux CN: the first deals with a hardware / software project promoted by one of the members of the ninux network which we helped supervise thanks to the multi-disciplinary methodology (Sec. 3.4.1), and that is now having an interesting development with the foundation of a start-up; the second is the beginning of a new, quite large, software project, which tries to bridge the work in WP2 and WP3 on the scalability of CNs (Sec. 3.4.2) and

[1] Please see https://github.com/netCommonsEU/

offer communities a tool to predict the evolution of the network as it grows, helpint to take the correct decisions in investing and setting-up new nodes.

The activity with the Cosenza ninux community and the supervising activity in Sec. 3.4.1, were particularly appreciated, because the teams felt they really got a contribution from netCommons, and they presented us with two appreciation letters that we report and comment D6.3 [44] discussing impact and interaction with the ninux community.

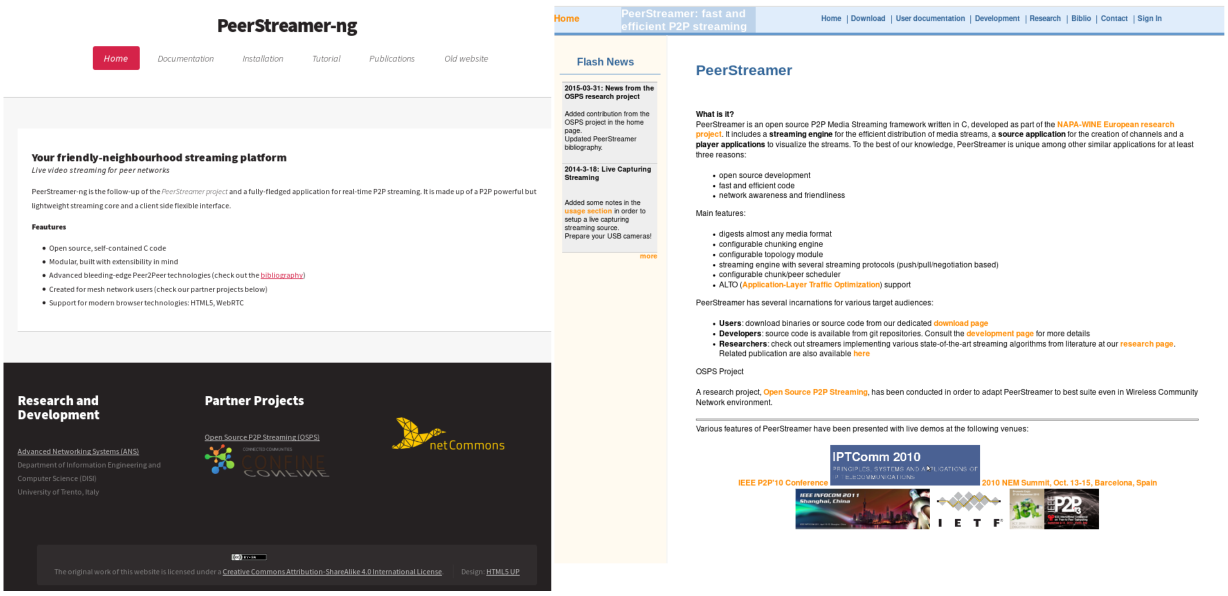
3.1. Dissemination and Feedback

3.1.1. PS-ng Communication and Diffusion

While our communication spanned across different media, our written documents have been of three kinds:

- PeerStreamer-ng website;
- PeerStreamer-ng wiki;
- end-user tailored guides in local language.

In an effort to make PeerStreamer-ng more usable by the community we completely reshaped its website, which was the product of previous research projects, and was mainly devoted to explain the technical advances that these projects achieved in the past. Now `www.peerstreamer.org`[2] is designed to clearly advise end-users on the early steps in our platform adoption, and runs smoothly on mobile platforms. In line with the nature of the netCommons project all our website content is released under a Creative Commons license.



Your friendly-neighbourhood streaming platform

Live video streaming for peer networks

PeerStreamer-ng is the follow-up of the *PeerStreamer project* and a fully-fledged application for real-time P2P streaming. It is made up of a P2P powerful but lightweight streaming core and a client side flexible interface.

Feautures

- Open source, self-contained C code
- Modular, built with extensibility in mind
- Advanced bleeding-edge Peer2Peer technologies (check out the bibliography)
- Created for mesh network users (check our partner projects below)
- Support for modern browser technologies: HTML5, WebRTC

Research and Development

Advanced Networking Systems (ANS)
Department of Information Engineering and Computer Science (DISI)
University of Trento, Italy

Partner Projects

Open Source P2P Streaming (OSPS)

The original work of this website is licensed under a Creative Commons Attribution-ShareAlike 4.0 International License. Design: HTML5 UP

Home
PeerStreamer: fast and efficient P2P streaming
Home | Download | User documentation | Development | Research | Biblio | Contact | Sign In

Flash News

2015-03-31: News from the OSPS research project

Added contribution from the OSPS project in the home page.
Updated PeerStreamer bibliography.

2014-3-18: Live Capturing Streaming

Added some notes in the usage section in order to setup a live capturing streaming source.
Prepare your USB cameras!

more

PeerStreamer

What is it?
PeerStreamer is an open source P2P Media Streaming framework written in C, developed as part of the NAPA-WINE European research project. It includes a **streaming engine** for the efficient distribution of media streams, a **source application** for the creation of channels and a **player applications** to visualize the streams. To the best of our knowledge, PeerStreamer is unique among other similar applications for at least three reasons:

- open source development
- fast and efficient code
- network awareness and friendliness

Main features:

- digests almost any media format
- configurable chunking engine
- configurable topology module
- streaming engine with several streaming protocols (push/pull/negotiation based)
- configurable chunk/peer scheduler
- ALTO (Application-Layer Traffic Optimization) support

PeerStreamer has several incarnations for various target audiences:

- **Users**: download binaries or source code from our dedicated download page
- **Developers**: source code is available from git repositories. Consult the development page for more details
- **Researchers**: check out streamers implementing various state-of-the-art streaming algorithms from literature at our research page. Related publication are also available here

OSPS Project

A research project, Open Source P2P Streaming, has been conducted in order to adapt PeerStreamer to best suite even in Wireless Community Network environment.

Various features of PeerStreamer have been presented with live demos at the following venues:

IPTComm 2010
PRINCIPLES, SYSTEMS AND APPLICATIONS OF IP TELECOMMUNICATIONS
IEEE P2P'10 Conference
2010 NEM Summit, Oct. 13-15, Barcelona, Spain
IEEE INFOCOM 2011 Shanghai, China
IETF
IEEE P2P'13

Figure 3.1: The new website (left) together with the old, non scalable website (left).

For technical, advanced usage, such as re-building the Cloudy container, our website points directly to our wiki[3]. A wiki can be very useful to engage power users and invite them in the active development of the platform.

Keeping in mind that not all community users are fluent in English, and the closer collaboration with ninux community network we have provided PeerStreamer-ng installation and early-steps documentation in Italian[4].

[2]Also reachable as http://peerstreamer.eu
[3]See https://ans.disi.unitn.it/redmine/projects/peerstreamer-ng/wiki
[4]Available at https://ans.disi.unitn.it/papers/guida_ninux_peerstreamer.pdf

3.1.2. Tests and Feedback Received

In the third year of the project we maintained contacts and we involved in the test and development several community networks, from which we received feedback to improve PeerStreamer-ng (more details will be given in Sec. 3.1.2.3). The interest in the platform has always been high, and we were able to install PeerStreamer-ng instances in two temporary occasions and two stable installations. In the former two cases it was a good chance to gain insights on the user's perception of PeerStreamer-ng, while in the latter cases we guided users to install and configure PeerStreamer-ng in their own network.

With the purpose of further engaging users and encourage feedbacks, we set up a demo instance of PeerStreamer-ng in our labs in Trento showcasing a TV channel (BBC) transcoded from satellite broadcast and a radio channel taken from the Internet[5]. This idea, albeit intuitive, provides a way to showcase our platform, it was a useful reference for our contacts in CNs and gave the possibility to have a taste of our work before installation. This improved the collaborative development and encouraged deployments.

ninux community users, in particular, used our demo server, commenting about PeerStreamer-ng adoption, possible enhancements and appreciated the very short delay PeerStreamer-ng introduces compared to other platforms.

3.1.2.1. The Battle of The Mesh Event

Our first insights came from Community Network experts at "Battle of the mesh v11"[6], a conference focused on community network technologies where we had the chance to give a demo of our platform capabilities. In that venue a mixed international community of people running community networks around the world gathers and discusses many technical and non-technical details about community networks. Every year a mesh network is created to test and debug routing protocols, and this year, we participated with two people from the Trento university that spent the whole week setting the network up and installing PeerStreamer-ng in a set of 10 Raspberry Pi that we brought to the conference. A webcam was constantly capturing the conference, and people in the event were introduced to the software and invited to try it. During the event we also gave a presentation on PeerStreamer-ng. PeerStreamer-ng was installed on top of the Cloudy platform. Itsdeployment, hence, was straightforwardly operated by simply connecting the mesh network nodes to the Raspberry Pis with an Ethernet cable. In this conference we had two main goals, the first was to introduce PeerStreamer-ng to experts and activits, the second was to take measurements on the live video distribution considering both the loss and delay of the resulting P2P overlay network.

Fig. 3.2 reports one of the network configuration we tested during the conference, with 15 mesh nodes and 7 PeerStreamer-ng nodes. We were able to run performance tests on a network that was extremely challenging: it had a very high density, it was running in a crowded room and it was interfered by several other wireless networks in the area.

Results shown in Figs. 3.3a and 3.3b summarize one of the tests we carried on, with one video source and seven video sinks, and show that the average delay was limited to less than one second, while the loss was limited (on average) to a value close to 6%, which is tolerable for live streaming. We also tested PeerStreamer using various routing protocols, and we confirmed that the change of protocol does not affect the compatibility with PeerStreamer.

For the development of PeerStreamer-ng Battle Mesh was a key moment because it was a further confirmation that outside our lab, in a real world challenging scenario PeerStreamer-ng could provide video streaming with a satisfactory quality. The tests we perfomed confirmed the advantages of PeerStreamer-ng compared to other approaches, which we better document when describing the deployment we did in the ninux network (see Sec. 3.3).

[5]The WebRTC server is on a public IP and can be reached at any time at http://napasource.disi.unitn.it:3000
[6]See https://www.wireless-meshup.org/doku.php

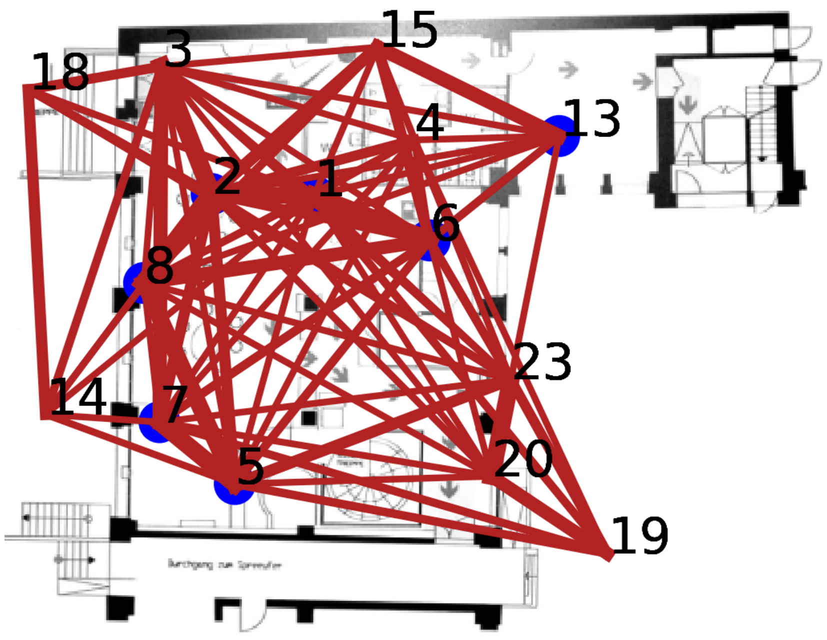
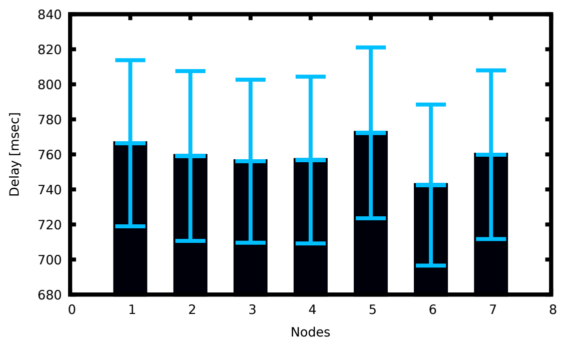
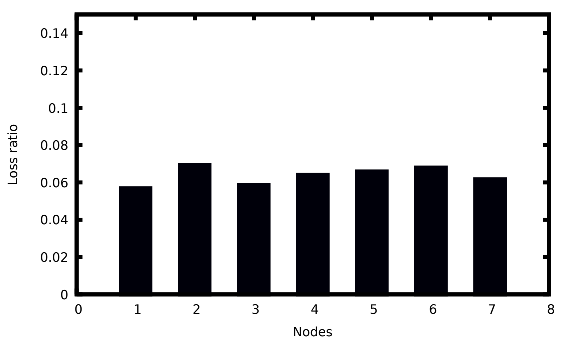


Figure 3.2: The network topology tested at the Battle of The Mesh Conference.



(a) The delay metric.



(b) The packet loss metric.

Our audience, composed mainly of power-users, gave us valuable feedback on the interface, the installation process and other desired features. Three of them are the possibility of encrypting traffic through HTTPS, sharing other media than webcam (like in screen sharing sessions), and the availability of a Debian/Ubuntu pre-packaged version for several platforms, which we document later on.

3.1.2.2. Sarantaporo.gr

We gave a second demo at the Sarantaporo public event “Building the community in Community Networks,” which was organized by the netCommons Project and was participated by the Sarantaporo.gr CN. In that situation the audience was completely different, and the goal was to share the live video of the conference that was happening in one of the villages to another village, in a Taverna (a local restaurant).

Albeit the scenario was simpler than in the Battle of The Mesh (we had only one source and a receiving end of the video) Sarantaporo community network users offered us a different user perspective. We had non tech-savvy users in a remote village wishing to follow the conference on CNs that we were performing in the main hall of a remote village. The set-up needed to be completely plug-and-play so that the PeerStreamer devices (again two Raspberry Pis) needed to be attached to a TV screen and the conference shown to people dining in the Taverna. We kept a textual chat open as a side-channel to receive feedback from the one of the CN member watching the video, and we helped locals to set-up the video source.

We collected several comments which focused mainly on usability, like resizable playout screen and the possibility to choose the streaming bitrate upon creation of the video source.

3.1.2.3. Developments as a Consequence of User Feedback

Given the feedback we received in the two meetings we described, several developments were carried on in the last year of the project in order to improve the usability of the platform.

Supported Architectures. While power users (like many CN users are) have no issues in compiling the PeerStreamer binary from scratch, having our software bundled, packaged and ready to be installed in various architectures makes the software adoption easier in community networks. To address the typical software installation scenarios, we decided to provide our software binaries through:

- Cloudy containers;
- Debian packages.

In the first case this was useful to maintain the integration with the Cloudy platform, whose adoption was described in Chapter 2. In the second, it made it possible to run PeerStreamer in a number of existing devices in the houses of CNs users.

Bundling PeerStreamer-ng in those package formats is not trivial and we came out we code repositories automating the process. Binaries target by nature a very specific hardware architecture so we decided to fully support both packaging formats for both the AMD64 bit and ARM7 32 bit (a.k.a., ARMHF) architectures. The former is the most widespread architecture for laptops and servers in general while the latter is common in small network devices like routers and antennas and it is compatible with Raspberry Pi models 2 and 3. We found Raspberry Pi devices very suitable for out deployment activity as they are:

- cheap;
- generally popular among network passionates;
- coming with a wide variety of well-supported software.

As we described in previous deliverables, PeerStreamer-ng will run on a dedicated device, which can be simply connected to the home router of a user (via Ethernet or Wi-Fi), and each user will connect to it with a web browser (we support the Firefox Browser). This simplifies the management of the software and improves usability compared to a software installed in the user's PC.

In the initial phase we supported the Raspberry Pi 3+ devices, the latest version of the hardware, which we bought to test and develop our code. After contacts with the ninux community we tested and refined our code to work also on previous versions of the hardware, that several ninux users already owned at the time.

Despite supporting two architectures for two different packaging systems requires a relevant effort in terms of coding automatic build-chains and testing the different package releases, we are persuaded it can maximize the impact of PeerStreamer-ng in community networks. At each new PeerStreamer-ng release, we hence proceed with the four different packaging of our software.

Debian Packaging. In order to meet the Debian/Ubuntu packaging requirement we received from community power-users, we released a building tool chain for PeerStreamer-ng producing a Debian package (a .deb file).

We release our tool chain through the usual netCommons account on github. The tool chain is designed to download and compile the latest version of our software and to include in the package description the needed information to handle the software dependencies upon installation. The tool chain can be used to produce packages for different computer architectures and we are currently providing on our website both an AMD64 and an ARM version[7].

After the package installation, which is standard and straightforward with Debian package managers, PeerStreamer-ng is available as a systemd service. A systemd service is part of the Debian daemons subsystem called *systemd*, and it can take advantage from well-mantained software architecture with a uniform interface. After the installation, PeerStreamer-ng starting and stopping operations can be performed by these simple systemd commands:

```
systemctl start peerstreamer
systemctl stop peerstreamer
```

And setting PeerStreamer-ng as a service launched at system start-up can be done with:

```
systemctl enable peerstreamer
```

Debian packaging is also essential to support virtualized installations of PeerStreamer-ng. In a couple of occasions in fact, ninux users asked to install PeerStreamer-ng not in a physical platform (a Raspberry Pi) but in a Docker Container in a virtual machine they already owned.

HTTPS and screen sharing. After the Battle of the Mesh we were asked to support screen sharing, also called *screencasting*. It is reasonably easy to modify PeerStreamer-ng libraries to support this further functionality, however, popular browsers support it only through HTTPS connections. We implemented HTTPS and screen sharing support for PeerStreamer-ng on a different development branch of our official code repository. However, we have not merged this modifications in the main code base as HTTPS brings some extra complexity complicating user experience. In particular,

1. certificates can only be self-signed (hence, they appear as not trustworthy to the browsers);
2. Firefox browser does not allow to associate a self-signed certificate to multiple TCP ports at once.

Certificates associate a public key to an identity, which is the website hostname in the case of web applications. Generally, hostnames are stored in Domain Name System (DNS) databases, and, their binding to servers (or server clusters) is unique. PeerStreamer-ng instances are installed on commodity devices run by community users, hence they typically do not have a unique hostname registered on some public DNS registry. Even if they had, users are not likely to pay for the release from a certification authority of the certificate for their hostnames. For all these reasons, certificates do not contain data that browsers can validate in the usual way and browsers prompt the users about the un-trustworthyness of the HTTPS connection.

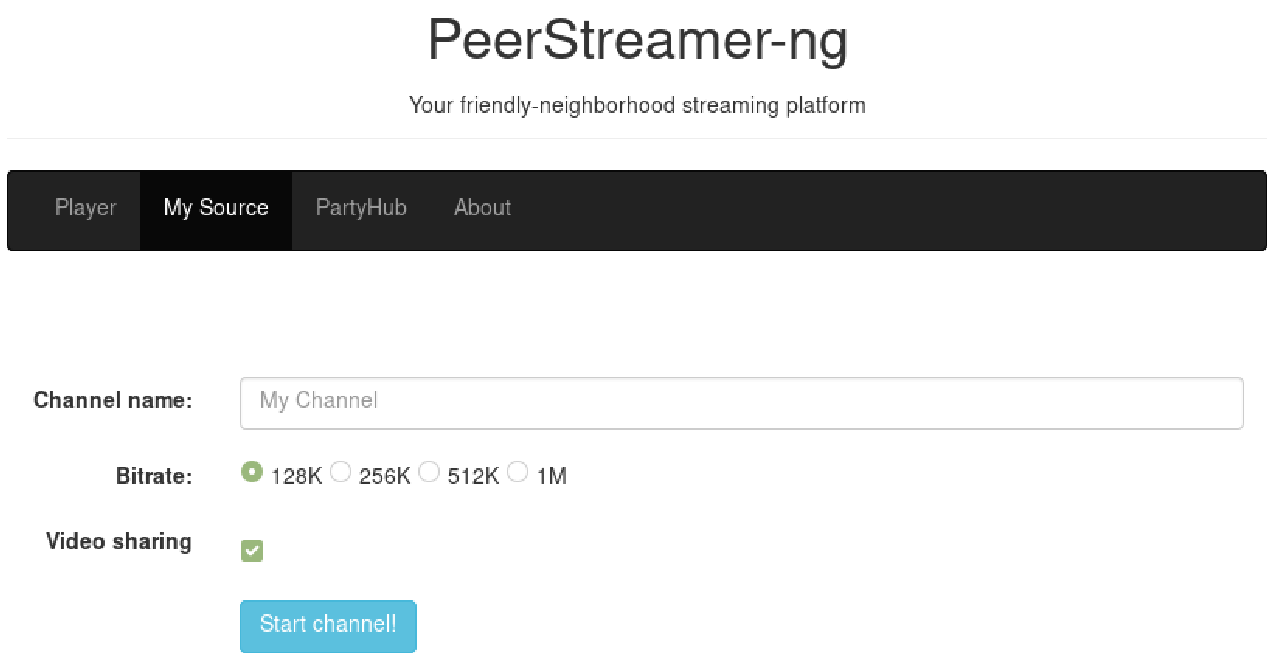
Firefox, one of the most wide-spread browsers in the world, when used with our infrastructure asks users for accepting the untrustworthy certificate. Moreover, even if the user accepts the certificate, Firefox does not allow the communications of the Javascript library with both the PeerStreamer-ng back-end and the Janus instance as they are instantiated on different TCP ports.

To solve this issue we would need to set-up our own Certification Authority (CA) and distribute, together with PeerStreamer-ng the valid certificate for the CA, to be installed in the user's browser. This would be highly impractical, and for this reasons we have thus decided not to merge HTTPS/screencasting features in the main code base yet.

Note that this is basically the same unsolved problem that any wireless router has when the user connects to it, since the router has a private IP address which the owner can change, it is impossible to associate a valid certificate to it. In the future, we can imagine to adopt a strategy similar to the one used by some router vendors, that use the popular Let's Encrypt service[8] to generate on-the-fly a valid certificate[9]. Due to the complexity of

[7]Both of them are available at http://peerstreamer.eu/getit.html
[8]See https://letsencrypt.org/.
[9]See https://www.asus.com/support/FAQ/1034294/

this solution, and the fact that indeed the local wireless network of an user is generally encrypted with a local key, we did not implement HTTPS in PeerStreamer-ng.


PeerStreamer-ng

Your friendly-neighborhood streaming platform

Channel name: My Channel

Bitrate: [✓ 128K] 256K 512K 1M

Video sharing ✓

Figure 3.4: The channel creation interface with the optional video quality and video sharing radio button.ad

Platform Development. Web Real Time Communications (WebRTC) technology has proven to best suit our streaming platform. It supports real-time communication, providing Real Time Protocol (RTP) streams, both for uploading and downloading. Its implementation is almost pervasive in nowadays devices through the latest browsers, like Mozilla Firefox, Chrome and Safari.

We have hence extended the integration of PeerStreamer-ng with Janus[10], the RTP-WebRTC bridge and ported our web interface based on HTML and javascript code to work entirely and only with WebRTC. Our current library, called *psng.js*, wraps the communication with Janus and it exports a handy and simple interface, easy to maintain and robust.

In particular, besides initialization functions, these are the key javascript methods provided:

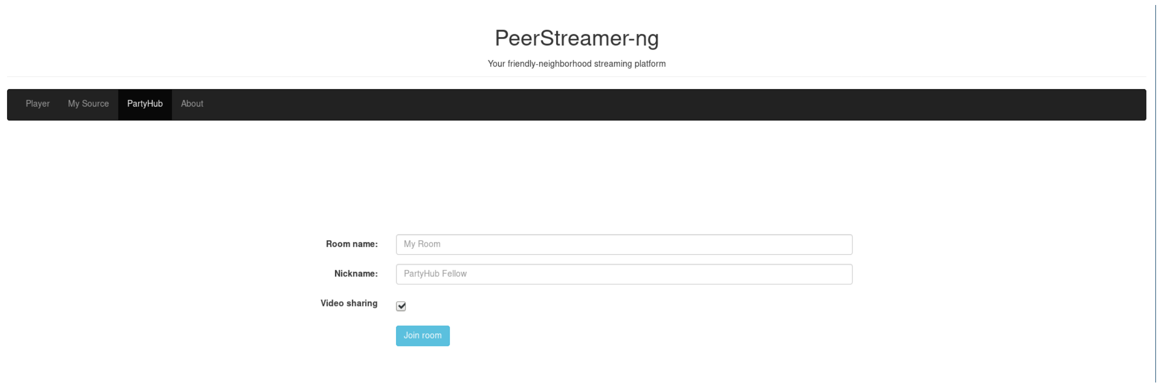
```
add_streamer(nickname, ip_address, port);
del_streamer(nickname);
```

The former creates and starts a new video stream for a given IP address and port source port and the latter destroys it.

Thanks to the insights and early adoptions of community users we could also start a never-ending process of fixes and debugging to the user interface. A process whose inputs we are very thankful for and whose nature is inherent of any deployed software. Among the outcomes of this process, our web interface is now completely designed and implemented to be suitable for devices with different screen sizes. With broad adoption in mind, we tested PeerStreamer-ng it using laptops, smartphones and tablets.

Finally, following the feedback we received from two communities (ninux and Sarantaporo) we added two features, the tunable video quality and the possibility to share only audio, which now appear in the PeerStreamer-ng interface Fig. 3.4.

[10]See https://janus.conf.meetecho.com/



Room name: My Room

Nickname: PartyHub Fellow

Video sharing ☑

Figure 3.5: PartyHub landing page. Users are required to pick a "conference room" and a nickname.

3.2. PartyHub

PartyHub is PeerStreamer-ng component enabling video conferencing. It is provided through the web interface along with the other functions.

In PartyHub there are two entities, participants and rooms, both identified through nicknames. Rooms are a logical grouping of participants, they exist as long as at least one participant is active.

When a user initializes PartyHub, she is asked to pick a room name and a nickname. All the users that pick the same room name join the same conference page with an incoming streaming window for each of the other participants.

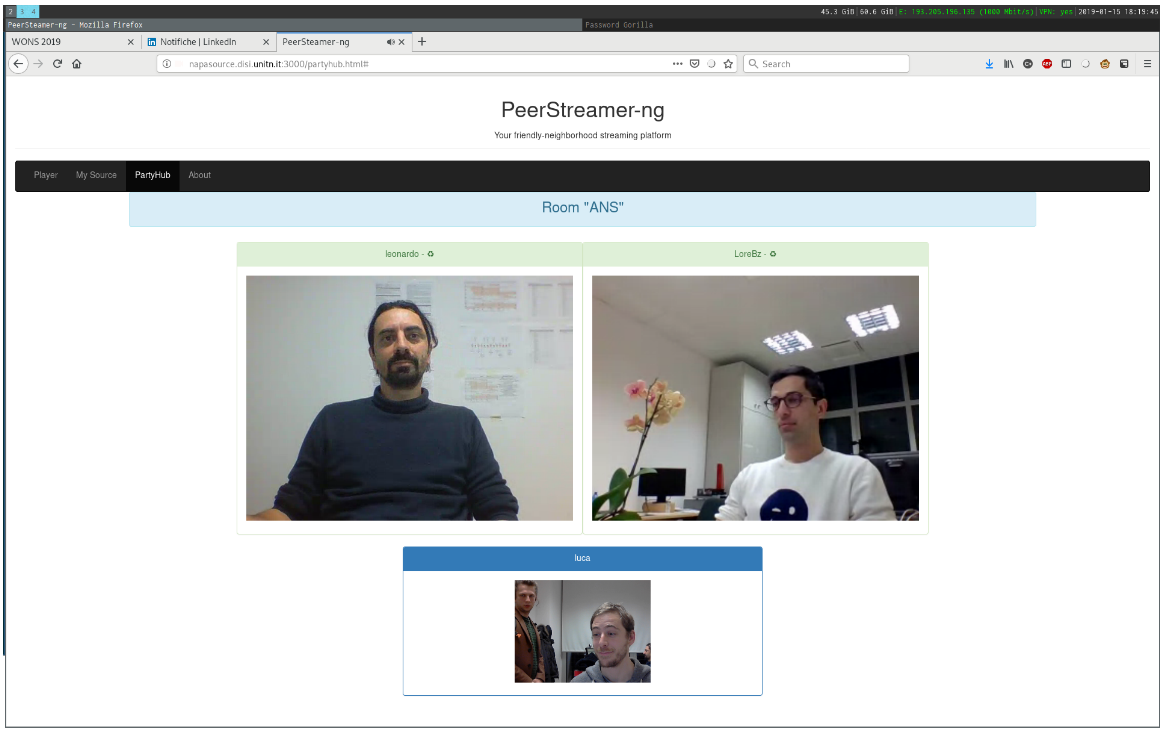


Figure 3.6: PartyHub conferencing room "netCommons" with three active users.

When a user joins or leaves a room, her action is signalled to the other instances which react by updating the PartyHub interface accordingly. A user joins a room by simply entering the room name in the Graphical User Interface (GUI) and she leaves it by closing the browser tab. Fig. 3.5 shows the landing page asking for user log-in, while Fig. 3.6 presents a sample video conference page with three participants.

Again at the moment of joining, we provide the users with the possibility of streaming audio only. This feature is especially useful in situations in which the user upload bandwidth may not to support a minimum video streaming quality.

PeerStreamer-ng, and hence PartyHub, is composed of three main entities:

- the front-end, namely the web graphical user interface;
- the back-end;
- Janus, the RTP-WebRTC bridge.

This architecture is explained in more technical details in D3.4 [2], here we just recall it to introduce the developments on PartyHub. All the three entities communicate with each other and they perform different roles. The front-end is responsible of just issuing ReST HTTP requests and dealing with the web page elements; Janus (named after the two-faced mythological god), from one side accepts or serves an RTP stream and from the other serves or accepts a WebRTC stream. Janus uses RTP for streaming to and from the back-end and WebRTC for streaming to and from the front-end. The back-end is responsible of orchestrating the resources, dealing with the front-end requests and managing Janus bridges. All these components run in the same host, for instance a Raspberry Pi, and the user simply uses his/her laptop or mobile browser to access to the service (both to create and receive the stream). In practice, the user experience is similar to on-line streaming providers, but there is no server, all the interactions between PeerStreamer-ng nodes happens in a peer-to-peer way.

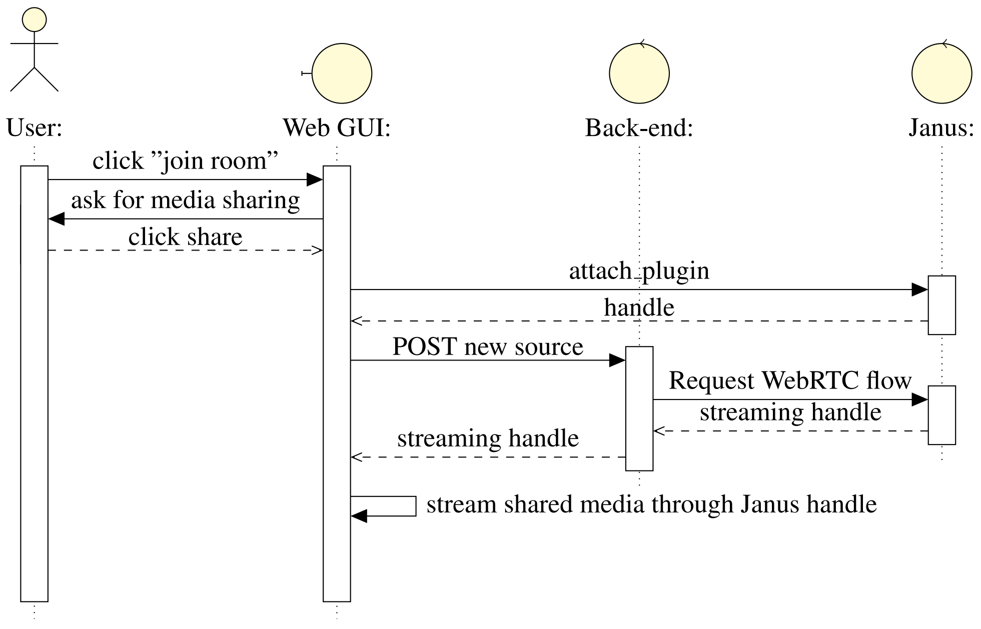


Figure 3.7: PartyHub videoconference join call diagram.

Fig. 3.7 depicts the interaction flow following the entrance of a user in a videoconference room. All the components excluding the user browser are placed in the PeerStreamer-ng/PartyHub installation. The communication between the web GUI and Janus happens only through handlers exchanged through the back-end which is the central orchestrator of the resources. With the Janus handlers, the web GUI can receive and stream the media as they were served directly through WebRTC from the P2P overlay. During the *join* process, a new streaming resource is created and, consequently, a new Janus handler is issued. The back-end dictates which RTP ports the handler is bound to so to start a new P2P streamer and advertise it.

Fig. 3.8 presents the PartyHub periodic tasks performed by the web GUI. The list of the room participants is updated and, if there are new participants the GUI had not been aware about, new streaming handlers are requested and the HTML element are consequently updated. The web GUI periodically sends an *UPDATE*

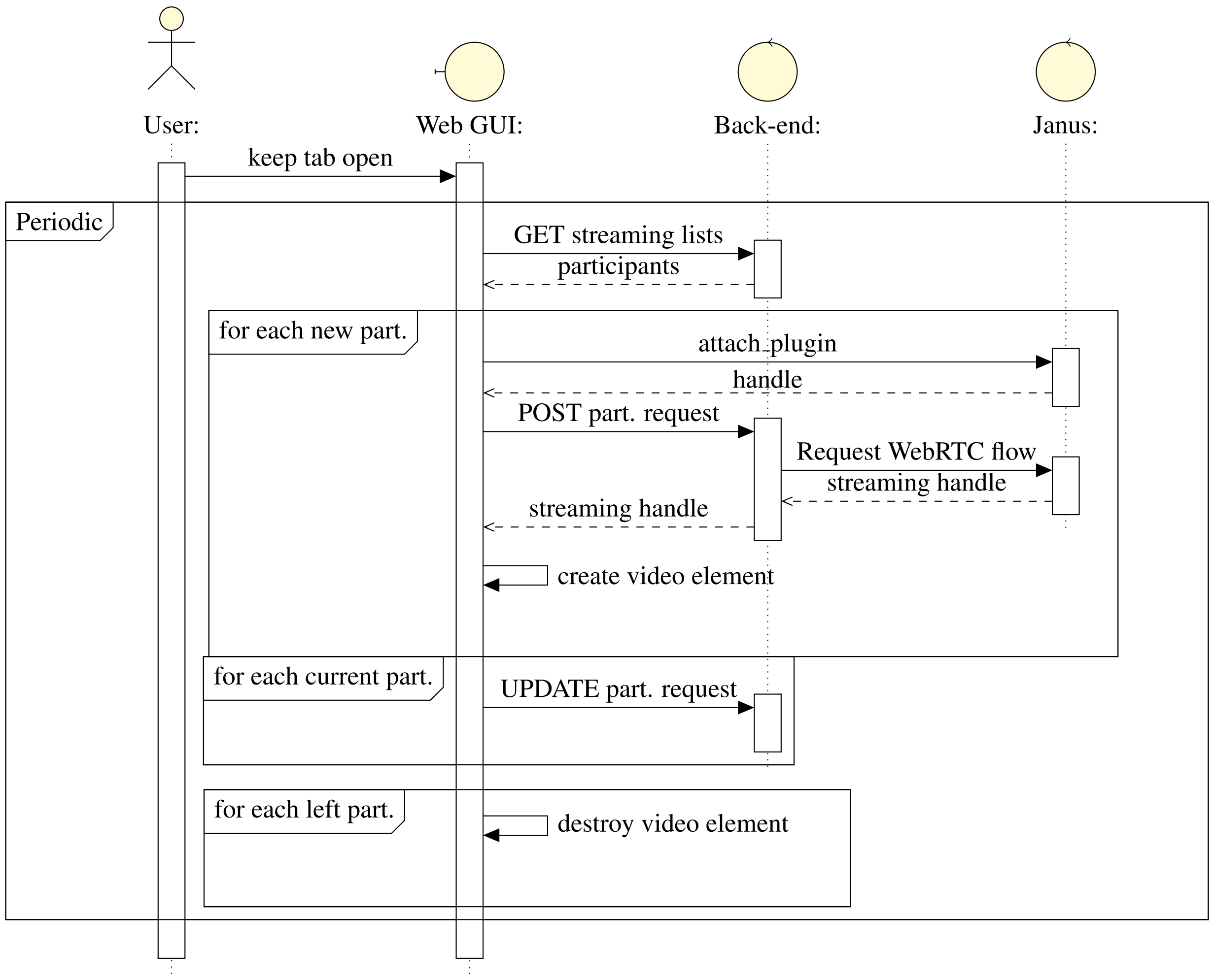


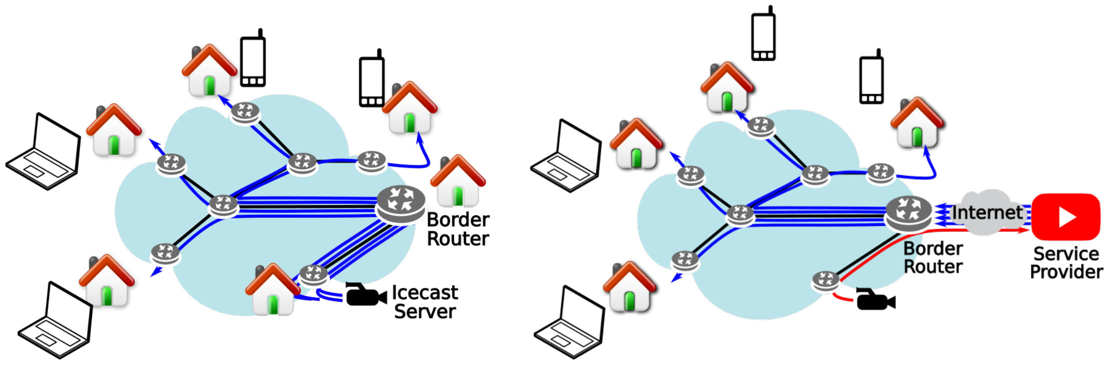
Figure 3.8: PartyHub periodic task call diagram.

request for each of the participant streams to signal it is still interested in receiving them to the back-end. That prevents the handlers to expire and grants resource de-allocation whenever the user leaves the page even without notifying it explicitly with the GUI. Finally, the GUI has to clean up the HTML elements associated to participants whose streaming have expired.

3.3. Experimenting On Ninux

In 2018 we directly approached the ninux community to install, test and adopt PeerStreamer. We contacted two ninux communities, the one from Florence and another from Cosenza (Calabria, Southern Italy). ninux is one of the olddest CNs in Europe, but also one that is under a period of internal discussion about its current organization and its perspectives. Part of this discussion is described in a chapter dedicated to Italy of the Global Information Society Watch (GISWatch) [45], and has to deal with two factors. The first is that the size of the community decreased, as an effect of both people leaving the community, but also of people leaving their cities and the country. Many people in the ninux island of Rome left the city, as well as those in the ninux island of Florence. In Florence, since the beginning of 2016 when the project started to interact with them, 5 key people (over the about 10 the community had initially) left the city for various reasons, and the network size started to decrease. The second reason, which is well explained in the GISWatch chapter is that ninux has always been a community with a very hacker-oriented mindset. Offering Internet Connectivity was never the primary goal of ninux (see also Deliverable D1.2 [46]), it was the side effect of a community primarily interested in the technical and ethical-political values related to community networks. Nowadays, the WiFi technology is well established and some of the technical curiosity that initially moved people to join ninux has faded. Yet there are some of the ninux islands that are growing, thanks to the effort of a core or people working on the infrastructure and to a real attachment to user's needs, which require Internet access but not only. Among these there is the ninux island of Cosenza, which has been active in the last period both in renewing the technological base of the ninux nodes, but also in re-discussing the legal organization of the community (see for instance the interactions documented in D4.3 [47]). With these two communities, Florence and Cosenza we were able to set-up tests and fixed installations in their network nodes.

3.3.1. The PeerStreamer-ng Architecture for ninux



(a) The streaming architecture using IceCast. **(b)** The streaming architecture using YouTube.

Figure 3.9: Two centralized streaming models.

PeerStreamer-ng resulted as a valid tool for ninux users because it offers two services that are unique in the context of video streaming in CNs: live video streaming (one-to-many) and video conferencing (many-to-many, PartyHub). The alternatives to PeerStreamer-ng are essentially two, the first is to use an internal open source video streaming service like Icecast[11]. With Icecast, represented in Fig. 3.9a the source of the video generates one single stream for every recipient of the video, irrespectively if the users are inside or outside the network. This actually generates an excessive load on the link that connects the video source to the rest of the network, and potentially to the gateway link if the video is served also outside the CN. Moreover, Icecast is not optimized for live video streaming, thus it introduces a very high overhead due to the use of an application layer not tailored to real time communications like WebRTC (see the discussion in [2] for the tests we performed with various protocols before converging to WebRTC). The second option is to use an external video service like YouTube, and let all the users receive the stream from the YouTube platform. While technically feasible this is not welcome by the community, which feels that forcing people to use a cloud-based proprietary solution is not aligned with the values of a CN. Moreover, this only partly solves the problem of the bandwidth erosion, as there is only one single video stream exiting from the source, but there is still one video stream per user, which will still produce a relevant load on the CN gateway as shown in Fig. 3.9b.

PeerStreamer-ng solves this problem with a balanced distribution of the load on the network links, as there is only one video stream to each PeerStreamer installation, which potentially redistributes the stream to more than one client. The presence of an instance of PeerStremaer-ng installed in a public server at the University of Trento also made it possible for people outside the network to watch the video without installing PeerStreamer-ng. The architecture described in Fig. 3.10 was the product of a series of interactions with the ninux community members (and not only, also with other people involved in the streamed events) which led us to this solution.

[11] See http://icecast.org

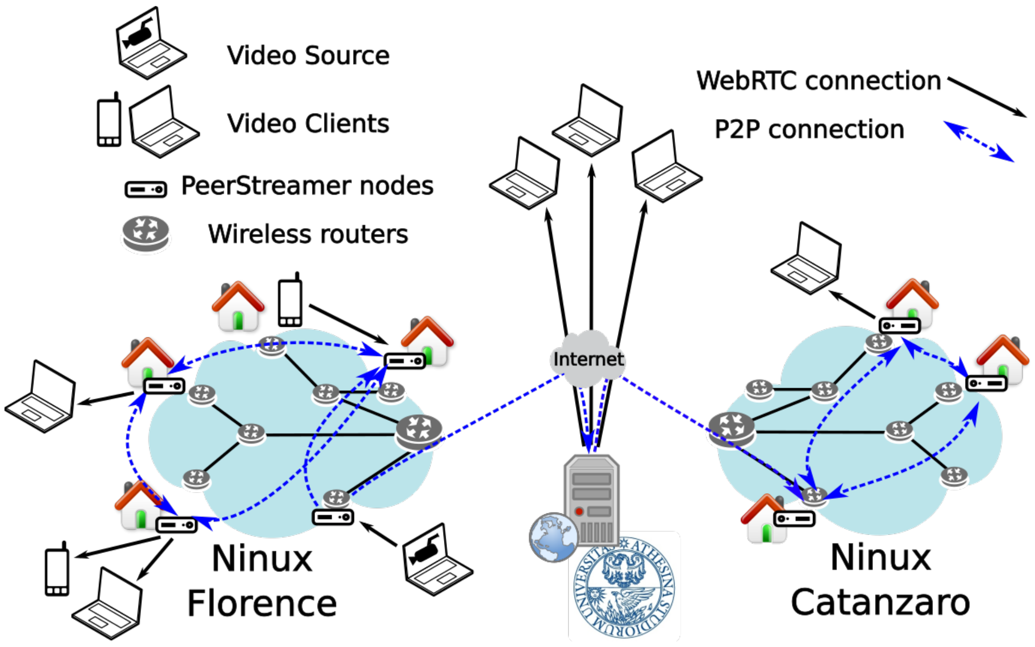


Figure 3.10: The PeerStreamer-ng configuration used in ninux; the presence of a public server allows two islands to easily share the same session with minimal resources.

The key elements to observe in Fig. 3.10 are the following ones.

- The load in the network links is evenly distributed, avoiding overloading of some specific link.
- People outside the CN can watch the video from the public instance of PeerStreamer-ng installed in the premises of the University of Trento.
- Every user inside the CN can watch the video stream from his/her own PeerStreamer-ng node without the need of a dedicated stream from the source.
- More than one user can watch the same video from the same PeerStreamer-ng node. After the first user, this does not create additional load on the network.
- Even if a user does not own a PeerStreamer-ng node he/she can watch the video from another PeerStreamer-ng node (in ninux or the public one).
- PeerStreamer-ng is fully protocol-agnostic, it does not depend on the support of the underlaying routing protocol. This is a key difference with any IP multicast-based solution, for which there is no stable support in the most used routing protocols for mesh networks.

The PeerStreamer-ng nodes receive an IP address from the local router of the user. This IP address must be routable in the CN in order for other nodes to exchange packets (CNs do not use Network Address Translation (NAT) to segment local networks). Yet the IP addresses used are private IP addresses and can not be routed on the Internet. To bridge the public instance with the ninux ones, we configured the PeerStreamer-ng nodes to acquire an address not only from the local network but also from a private class which belongs to a Virtual Private Network. This class can be routed both in the CN and in the Internet (via the Virtual Private Network (VPN)). In practice, a node in ninux may or may not have a local gateway (a standard ADSL connection) and the routing protocol (in this case OLSRv2) will decide which is the most convenient path (through ninux or through the VPN via the ADSL). This also solves another problem, that is the uplink bandwidth scarcity on the source when the video needs to be distributed to the Internet. Let's say that the source node has an ADSL connection. With the YouTube solution, the node will use its own ADSL connection to upload the video to YouTube. If the connection is not fast enough, the quality of the streaming will be affected. In PeerStreamer-ng, from the source the video is shared among more ninux nodes, and from each ninux node, potentially, a fraction of the chunks can be sent to the public Internet instance, thus obtaining an effect of connection bundling. In fact, PeerStreamer-ng nodes (even without anybody watching the video) contribute to the distribution of the video chunks inside the ninux network, but also towards the publi Internet instance. This is possible only thanks to the P2P algorithms used in PeerStrreamer-ng, and would not be possible with any other option.

In the current deployment, the two networks (Florence and Cosenza) are not bridged due to difficulties in configuring both networks to work this way at the same time. In the next events programmed for 2019 we plan to bridge the two networks.

Streaming Tecnolokia in Ninux Florence. As described in [3] the ninux CN of Florence started to realize a set of skill sharing talks together with other organizations in Florence. Before the summer a first series of talks were organized, and in fall 2018 four more events were organized[12]. Before the last two talks, three PeerStreamer-ng nodes were installed in the houses of people in Ninux Florence, and also the link to the public instance was publicized in the ninux telegram chats (and also other chats of interested people).

The key "selling point" of PeerStreamer-ng was the simplicity of its use and the excellent trade-off between the video quality and the low resources needed. The speaker could simply connect to the ninux network (in the place where the talks took place there is a ninux node called "ex-fila", described in D3.6 [3]) and from there connect to any PeerStreamer-ng instance in the network (or the public one) via his browser. No configuration, log-in, or software installation was needed, just a Mozilla Firefox browser. Moreover, in the ex-fila ninux node the available bandwidth of the Internet connectivity is extremely low, so that with a one-to-many transmission model there would have been no practical way of achieving the same result using some on-line service.

Since PeerStreamer-ng is a fully P2P software there is no practical way of counting the effective number of users watching the streaming sessions. For this reasons we publicized the talks in ninux chats (and other non-ninux specific chats) and then we asked people to give us feedback on these chats. In both events we could gather feedback from 3-7 viewers, which, considering that the live audience was about 15-20 people was a positive response. Interestingly enough, among the remote audience there were at least two ninux users that could follow the talk even if they do not reside in Italy anymore.

As a conclusion, setting-up this testbed was challenging due to several reasons, both in the technological and in the social domain. At the very beginning we had to support ninux Florence to recover part of their network that, due to decrease of the participation in the community, was non-functional. A large amount of work was needed to design and implement the architecture depicted in Fig. 3.10, to help people setting up PeerStreamer nodes, and to help them to organize the talks (one of the talks was held by Leonardo Maccari). It was an iterative process which helped the community to acquire a new internal service which solves some practical issues. The ninux community has shown interest, the PeerStreamer-ng nodes are still running and for 2019 the new skillsharing nights will also be streamed using PeerStreamer-ng.

PartyHub in Ninux Cosenza. Another Ninux island has shown interest in using PeerStreamer-ng in their network is Ninux Cosenza. In this case, we just proposed the installation and use of PeerSteamer-ng to the community, and gave them feedback and support for the installation phase. As a result, there are now 3 instances of PeerStreamer-ng running in the ninux island of Cosenza, on the nodes of three users that dedicated one Raspberry Pi to PeerStreamer-ng. PeerStreamer-ng is mostly used for one-to-one video communication, thus using PartyHub as a video conferencing tool. Again, the "selling point" of PeerStreamer-ng is its easyness, and the excellent performance on a local network. In fact, the use of the UDP transport protocol together with a really lightweight signaling overhead reduces the latency compared to a web-based, TCP-based solution. An appreciation letter from the community of Cosenza is reported in D6.3 [44] describing their experience with PeerStreamer-ng and netCommons in general.

[12] See http://www.firenze.ninux.org/2018/10/21/continua-skill-sharing/

3.4. Additional activities: the Turnantenna, and the CN Graph Generator

3.4.1. The Turnantenna

During the third year of netCommons, one member of ninux Florence, Marco Musumeci, started to develop a project that was previously bootstrapped by another member, Salvatore Moretti. The project is made of the hardware and software design of a device named “Turnantenna”, which is a two-axes motor plus a 3D-printed structure to be able to point a ninux device from remote. At the very beginning the project was imagined and realized by Salvatore in his spare time, and never reached enough maturity to be deployed. Later on, Marco took over the project and decided to develop it using a different approach.

Marco, being a Mechanical Engineer, was fascinated by the idea and contacted Leonardo Maccari to imagine a path that could make the project come to life. Leonardo Maccari agreed to support the project, and invited Marco to review the Participatory Methodology developed in WP3. At the time the methodology was not yet fully defined, and it seemed to be too complex to be applied to a one-man project. Later on instead Marco understood that the project was more complex that he could handle and he needed to frame it in a broader context, there he re-evaluated the possibility to use the participatory methodology. He decided that to make it successful the project would need to be realized with the support from members and non-members of the community, to find sources of funding and to publicize it with other people inside and outside ninux. In short, the methodology motivated him to modify the spirit of the initial project to make it sustainable.

From May 2018, Marco was able to receive a scholarship from the Google Summer of Code to develop the Turnantenna[13], he presented his work to ninux meetings and to the audience of the Battle of the Mesh. After the summer he submitted it to the open call of the Maker Faire in Rome (the largest makers faire in Italy), there it was accepted, presented in a dedicated stand and received an award. It was featured on a French engineering magazine, and as Fig. 3.11 show the Turnantenna is on its way to completion.

Leonardo Maccari followed and guided his work, and recently Marco, together with other ninux (and non-ninux people) participated to an accelerator programme to create a start-up on the concept of the Turnantenna, a small contribution, maybe, but an interesting industrial evolution on netCommons. A letter of appreciation signed by Marco is reported in n D6.3 [44], where he describes how the participatory methodology and other netCommons results were instrumental to carry on his work.

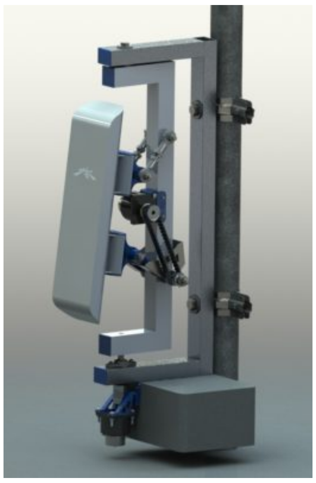
After we received the letter the Turnantenna project was chosen among the best three projects of the programme and will receive a seed funding to start a social enterprise.

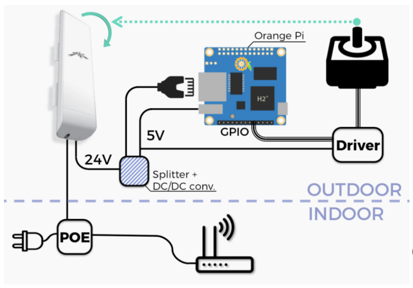
3.4.2. The Graph Generator

As a result of the work carried on in various WPs in netCommons, for instance WP2 dealing with the scalability of community networks, we started a completely new line of research. UniTn led this effort to realize a new realistic topology generator for community networks, with the contribution of AEUB-RC and UPC. The goal of this research effort is to characterize the technical scalability of a mesh-based community network, and, on the long run, to estimate its social impact in various real-world scenarios. Exploiting open source resources, such as Open Street Map and very detailed (less than 1 m resolution) Laser Imaging Detection and Ranging (LIDAR) data on buildings, we introduced a framework for the stochastic forecast of the growth of a Community Network given the area where the community starts building it. This base methodology, implemented into an automated tool, takes into account the technical and economic feasibility of adding nodes to the network, as well as guaranteed limits on the per-node performance of the network in saturation predicting the topology of the network and its limit size given the above constraints.

The methodology is coupled with simple economic incentive schemes to explore if proper incentives mechanisms can influence (and improve) the growth of the network. We selected four different scenarios: Urban,

[13] See https://blog.freifunk.net/tag/turnantenna/.


(a) The initial renderings of the Turnantenna.



(b) The electrical scheme.

(c) The Turnantenna shown at the Maker Faire, with a blue ribbon award.

ENGINEERINGNET MAGAZINE
MAGAZINE POUR LES INDUSTRIES DE FABRICATION ET DE PROCESSUS ANNÉE 25 – NO 143 – NOVEMBRE 2018
Superordinateurs : à quoi faire ?
Topingénieur Yves Van Ingelgem 14 Impression 3D avec un plus 20 Le décollage des hélicoptères belges 23 Electromobilité de chez nous 28

Dans l'actualité

Europlasma
Europlasma, le nanocoater d'Oudenaarde, qui utilise le plasma pour appliquer des films de protection ultra-minces sur des formes en 3D, a été intégré cet été au groupe d'investissement Andlinger & Company, qui avait déjà auparavant pris sous son aile le français Coating Plasma Innovation (machines de laminage à plasma atmosphérique). Europlasma est principalement connu pour ses technologies plasma basse pression Nanofics, par lesquelles les couches de protection sont appliquées sur les filtres et les composants électroniques et PlasmaGuard, qui applique des revêtements hydrofuges sur les textiles. Brigitte Speleers est directrice générale depuis septembre.

The Bright World of Metals
Du 25 au 29 juin 2019, GIFA, le salon international des technologies de fonderie, aura lieu à Düsseldorf. La foire se déroule simultanément avec trois salons associés : METEC, THERMPROCESS et NEWCAST sous le slogan : « The Bright World of Metals ». Sont abordés, la technologie et les produits de fonderie, la métallurgie et la technologie des traitements thermiques. Les nouvelles technologies et Industrie 4.0 restent toujours en mémoire. En 2015, 2 214 exposants ont attiré quelques 78 000 visiteurs venus de 120 pays. Encore une fois, de nombreux séminaires et colloques, des forums technologiques et un congrès international seront les pôles d'attention. Metec présente l'European Metallurgical Conference (EMC), la plus importante conférence européenne sur la métallurgie des non ferreux et les European Steel Technology and Application Days (ESTAD), consacrés à l'acier, aux matières premières, à l'industrie manufacturière, aux applications et à l'acier aux niveaux environnemental, ingénierie et énergie. THERMPROCESS inspire avec le spectacle spécial FOGI et le symposium THERMPROCESS de VDMA Metallurgy. NEWCAST se distingue par le Foundry Meeting, le forum et le trophée NEWCAST et le forum GIFA ferme la boucle.

Open source
La sixième édition de la Maker Faire de trois jours à Rome a attiré plus de 100 000 visiteurs dans sept pavillons remplis à craquer. Don Eyles, autrefois programmeur de l'atterrissage d'Apollo 14 sur la lune, a souligné l'initiative « maker » de la British Interplanetary Society. Elle souhaite envoyer des masses de satellites dans l'espace à un prix coûtant, non de 100 000 dollars, comme le CubeSats (10 x 10 x 10cm) mais à une fraction de ce montant avec les quatre fois plus petits Q CubeSats. Q signifie « quarter sized ». Marco Musumeci, ingénieur en mécanique, a montré son antenne à deux axes avec laquelle un signal WiFi peut être envoyé très précisément jusqu'à 8 km. « Vous pouvez facilement l'utiliser pour ajuster la topologie d'un réseau. Vous définissez l'azimut et cela commande les moteurs. » Son travail s'inscrit dans l'initiative open source Ninux (Neighborhood Internet, Network Under eXperiment). (Photo: LDS)

CONNEXION DE RÉSEAUX DE BUS DE TERRAIN
avec le coupleur PROFIBUS/PROFINET

- Modèle très compact pour montage sur rail
- Échange jusqu'à 244 octets de données d'E/S
- Aucun impact du côté opposé en cas de défaillance du système bus
- Réglage des adresses PROFIBUS via le commutateur DIL ou le logiciel
- Configuration uniquement par le biais du fichier GSD/GSDML, aucun autre logiciel requis

Helmholz

novembre 2018 **Engineeringnet Magazine** | 9

(d) Marco Musumeci Featured on the French Engineeringnet magazine.

Figure 3.11: Various phases of development of the Turnantenna project.

Suburban, Intermediate, and Rural targeting spots in Tuscany, Italy. Results in all these scenarios highlight the characteristics of the topology that spontaneously emerge from the natural growth of the network, and the advantages that properly crafted incentives bring to this process, improving the size, the performance, and the resilience of the network emerging from this spontaneous process.

We realized an open source network generator which is already available on-line[14]. This software is made of a back-end and a front-end. The former is a PostgreSQL database where we loaded the map, the building

[14]See https://github.com/AdvancedNetworkingSystems/TerrainAnalysis.

positions and a layer representing the altitude measured with LIDAR. The database can be queried to know if there is line-of-sight between two buildings. Moreover, we included data coming from the data sheets of commercial outdoor Wi-Fi devices in order to be able estimate the available bandwidth of the wireless links. The former is a software able to generate random topologies with queries to the database, and estimate how much a network can grow before it breaks some quality threshold we set as a stop condition. For each network we also output an estimation of its cost in terms of capital expenditure needed to set-up all the network nodes, based on the average prices of the wireless device.

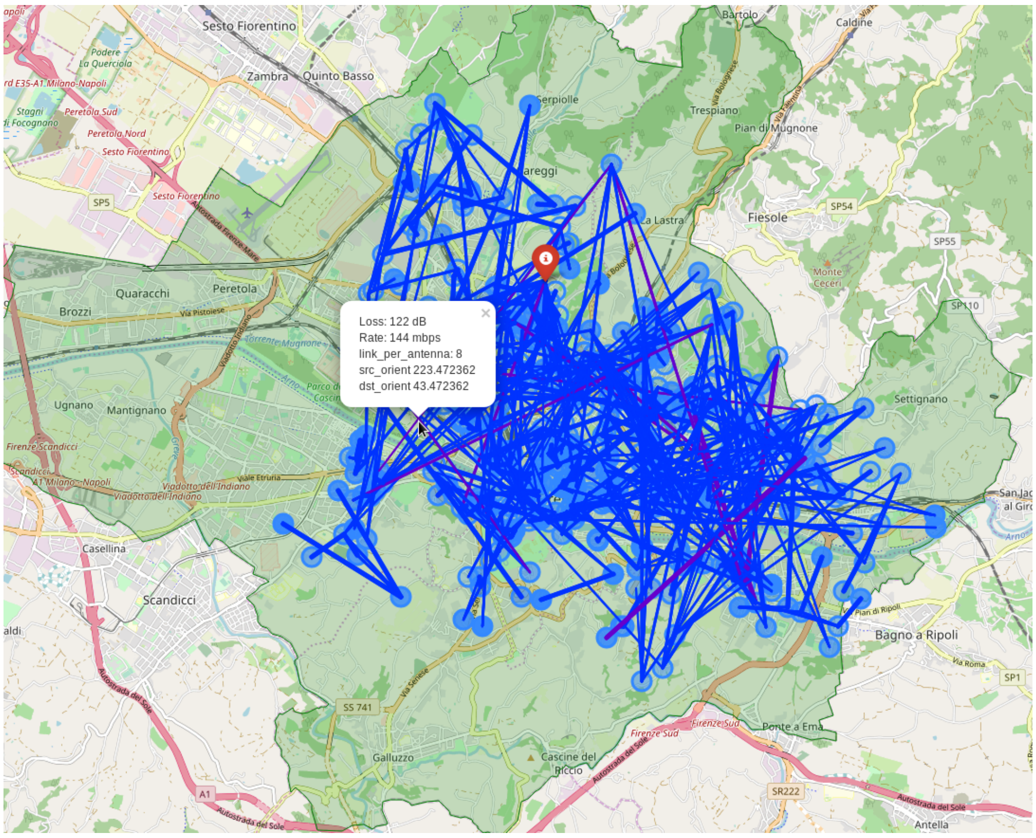

Loss: 122 dB
Rate: 144 mbps
link_per_antenna: 8
src_orient 223.472362
dst_orient 43.472362

Figure 3.12: A potential network generated in the area of Florence (Italy), with details on a specific link.

This line of research has endless possibilities of future extension, among which we mention:

- Characterizing the sustainability of mesh community networks in different scenarios: urban, rural, sub-urban. We will answer the question: is a mesh network a competitive option for a specific geographical area?
- Detailing the technical good practice that make a mesh network scale. We will answer questions on what kind of device, what number of device and what specific configuration members have to use to achieve long-term scalability, which is generally in competition to short-term targets (like reducing the cost or maximising the local available bandwidth). As studied in WP2, robustness and absence of single points of failure can be one of the outcome of this process.
- Studying the various economic approaches that can be adopted to make the network sustainable. We will answer the question: what is the best cost-sharing model for a network growing in a specific geographical area?
- Producing open APIs to let members know in advance if there is line of sight between one building and another, and what is the maximum bandwidth they can expect with certain Wi-Fi devices. Community networks members today have no tool to perform this task, and need to physically go on roofs to check

the presence of line-of-sight.

The results of this activities are included in a submitted paper, currently under review, with authors from UniTN, AUEB and UPC [48]. The paper has been invited to a special issue in Elsevier Ad Hoc networks, the most important journal in the area of distributed wireless networks.

4. From CommonTasker to AppLea: experimentation and development activities

For the cooperative mobile app fro smart farming, the third year of the project marked the beginning of the actual experimentation with the app under actual conditions of intended use. Decisive to this end, much as throughout the app participatory design and development process, was the engagement of the local community of the Sarantaporo.gr CN. The feedback from this field experimentation has been the main driver for the development activities around the app this year.

4.1. From laboratory experimentation to field experimentation

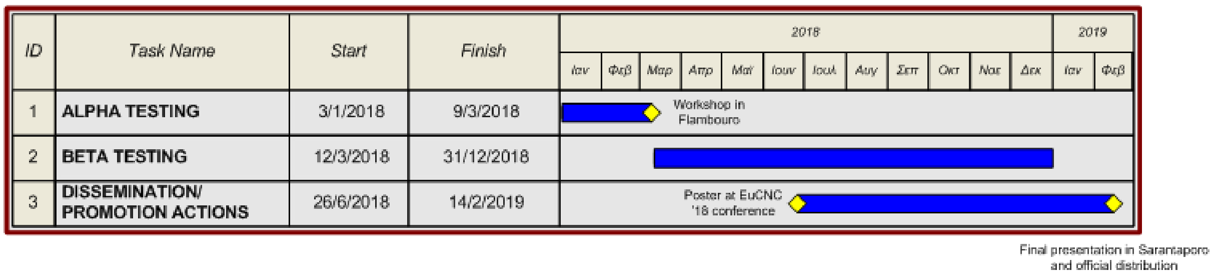
The development of the Android mobile app was conceived from the beginning of the project as a participatory design process, which would involve the local community in all its design and development steps. During the first two years, the engagement of the community was mainly achieved through physical and virtual meetings. In these meetings, we (the development team) would present the current status of the app and the community would give us feedback as to what could or should be added/removed/modified. The workshop in Milea village, in the Sarantaporo area in November 2016 and the subsequent physical meetings with the Sarantaporo.gr team in February, May, and November 2017 were key moments in this process [49, 6]. In between, several phone conferences and exchanges with the community gave further inputs to the app development process.

Within 2018, the engagement of the community in the design process both escalated and became more systematic. The first step to this end, in the beginning of the year, was to set up an extended *alpha testing* team that comprised the AUEB development team (A. Pilichos, M. Karaliopoulos), the Nethood colleagues that coordinated the participatory design process (P. Antoniadis, A. Papageorgiou) and three members of the Sarantaporo.gr CN (G. Klissiaris, V. Chryssos, A. Vaitsis). These seven people formed a tight group that tested the first "internal" releases of the app. Its short-term mission was to ensure a stable version of the app, which could be presented and distributed to selected community members in the Sarantaporo area for field testing.

This indeed happened in the weekend of March 10-11 in the context of a workshop that took place in Flambouro, one of the 14 villages served by the Sarantaporo.gr CN. Besides presenting the first open release of the app and explaining how this could be downloaded and bootstrapped (user profile creation, credentials etc), we set up what would serve as the *beta testing* team for the app. This team included three farmers from the Sarantaporo area, who are members of the Sarantaporo.gr CN (S. Katis, D. Dallas, T. Minas). They come from different villages in the area and have different skills and experience with mobile apps (from elementary to more advanced). The latter was deemed critical for making this team representative of the potential users of the app and give more credibility to the process.

One of the first outcomes from the March workshop in Flambouro was the official renaming of the app, from CommonTasker to AppLea. CommonTasker was the original name given to the app by the time of the netCommons project proposal, given the aspirations about it by that time [1]. AppLea is the name chosen by the beta testing team, essentially a pun on the name of a village in the area.

From the workshop on, the two groups, the alpha testing and beta testing teams, have coexisted and experimented with the app in parallel. Most of their exchanges have been through Telegram groups, letting all questions and requests be shared among the whole group. From March till December, there have been five updates of the app (releases) for experimentation by the beta testing team, and more internal releases for in lab testing by the alpha testing team. Each new release incorporated the responses of the development team to the



ID	Task Name	Start	Finish
1	ALPHA TESTING	3/1/2018	9/3/2018
2	BETA TESTING	12/3/2018	31/12/2018
3	DISSEMINATION/ PROMOTION ACTIONS	26/6/2018	14/2/2019

Figure 4.1: Plan for experimentation with the app, in-lab (alpha testing) and field (beta testing).

feedback received by the alpha and beta testers on the previous release, closing one round of exchanges with the beta testing team and initiating a new one. We describe the changes that were made in each component of the mobile app in Sec. 4.2.

Fig. 4.1 captures the roadmap of experimentation with the app during the last year of the project. A final meeting in the Sarantaporo area is planned for early February to further promote its use to the local community.

4.2. Evolution of the AppLea functionality through community feedback

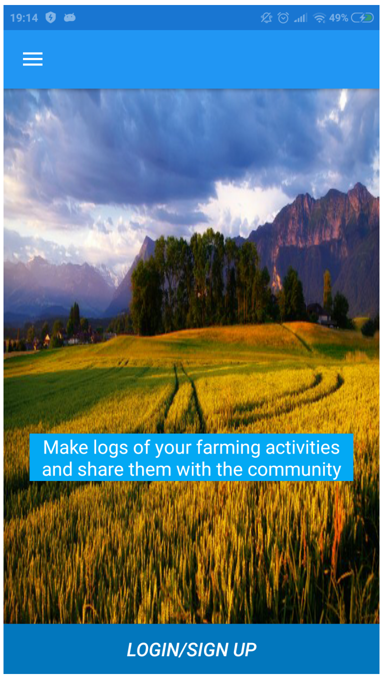
In what follows, we describe the main development activities that took place around the app in Y3 of netCommons. This description is organized largely in line with the structuring of the app modules in D3.4 ([2], Section 4.3 "Application architecture and design"). We report here changes that have been made with respect to the status of the modules reported therein. Screenshots are presented from the mobile front end, the part of the app that is visible to the end users and the one the feedback from beta testing concerned. Several of the requested changes were addressed by minor additions and modifications in the way information is organized and presented to them and could be resolved fast. In most cases, however, the requested changes demanded more extensive modifications, with implications for the backend of the application as well.

In summary, what came up during the testing/experimentation phase is that the users expressed far higher interest for the logging functionality of the app. Namely, they insisted on finer details regarding how information about their farming activities can be most efficiently and easily logged down and a posteriori retrieved. On the contrary, they showed less enthusiasm about the capability to share data with other users. In fact, several of those expressed concerns about the kind of information that could be exchanged since they anticipated some of their practices and how-to knowledge in the farm as a comparative advantage that they did not want to dismiss.

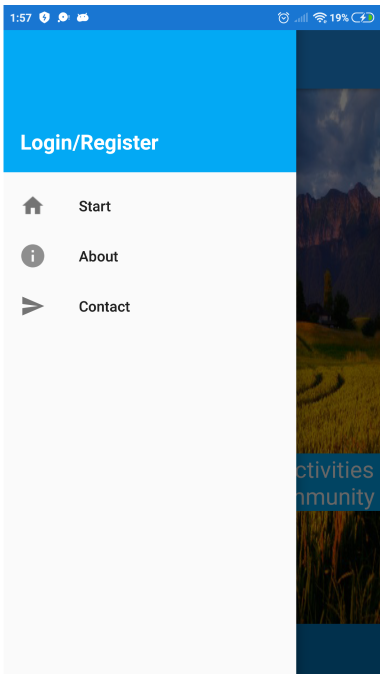
The changes that were made to the app reflect the user preferences, as expressed in the feedback received primarily from beta testers, and to a lesser extent, by alpha testers. In user interface terms, compared to the app status in [2], the calendar module was promoted to a major app module, while the gamification module was demoted to a less important one. Commensurate with this bias in users' preferences was the effort we invested in modifying and enhancing each app module. Yet, we made sure that the app supports data sharing, at least as an option for those who want to practice it.

4.2.1. The splash page

The original splash page was fully replaced in the course of the beta testing process. First, we changed the main graphic to reflect better the revised focus of the application on farming activities, rather than as a more generic tool for crowdsourcing activities and sharing [2]. The new version of the AppLea splash page is shown in Fig. 4.2a. It no longer consists of two frames, this was deemed an unnecessary overhead and an out-of-date design practice. There is one graphic instead and an enter button that guides the user to the login page of the app.

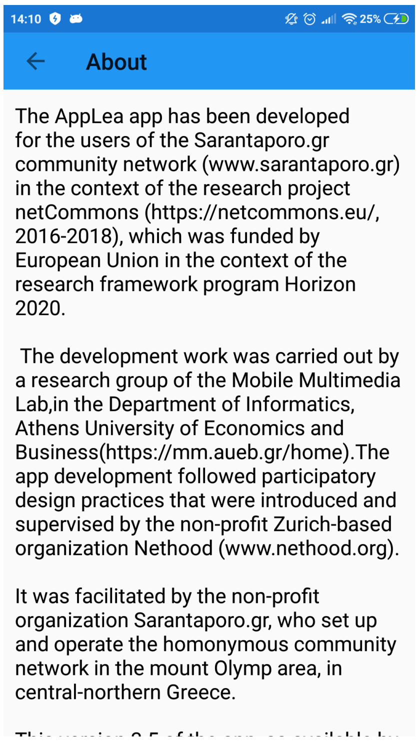

Make logs of your farming activities and share them with the community
LOGIN/SIGN UP

(a) Menu button at the splash page.



(b) Menu added to the top-left of the splash page.

Figure 4.2: The revised AppLea splash page.



About

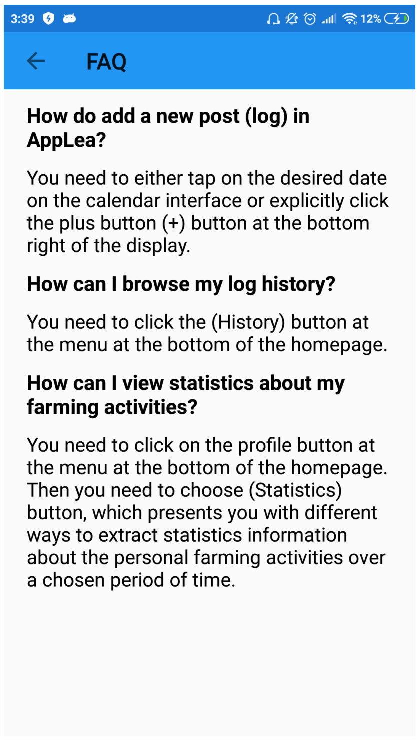
The AppLea app has been developed for the users of the Sarantaporo.gr community network (www.sarantaporo.gr) in the context of the research project netCommons (https://netcommons.eu/, 2016-2018), which was funded by European Union in the context of the research framework program Horizon 2020.

The development work was carried out by a research group of the Mobile Multimedia Lab,in the Department of Informatics, Athens University of Economics and Business(https://mm.aueb.gr/home).The app development followed participatory design practices that were introduced and supervised by the non-profit Zurich-based organization Nethood (www.nethood.org).

It was facilitated by the non-profit organization Sarantaporo.gr, who set up and operate the homonymous community network in the mount Olymp area, in central-northern Greece.

This version 2.5 of the app, as available by

(a) About the app.



FAQ

How do add a new post (log) in AppLea?

You need to either tap on the desired date on the calendar interface or explicitly click the plus button (+) button at the bottom right of the display.

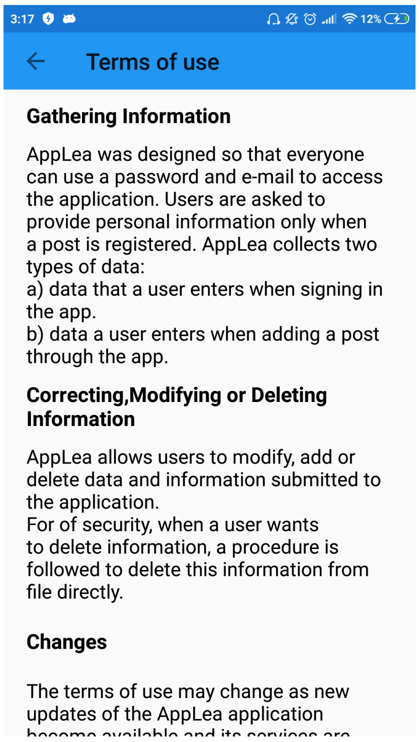
How can I browse my log history?

You need to click the (History) button at the menu at the bottom of the homepage.

How can I view statistics about my farming activities?

You need to click on the profile button at the menu at the bottom of the homepage. Then you need to choose (Statistics) button, which presents you with different ways to extract statistics information about the personal farming activities over a chosen period of time.

(b) FAQ.



Terms of use

Gathering Information

AppLea was designed so that everyone can use a password and e-mail to access the application. Users are asked to provide personal information only when a post is registered. AppLea collects two types of data:
a) data that a user enters when signing in the app.
b) data a user enters when adding a post through the app.

Correcting,Modifying or Deleting Information

AppLea allows users to modify, add or delete data and information submitted to the application.
For of security, when a user wants to delete information, a procedure is followed to delete this information from file directly.

Changes

The terms of use may change as new updates of the AppLea application become available and its services are

(c) Terms of use.

Figure 4.3: Additional pages with information about the app, attached to a menu button on the splash page.

Second, and more important, in response to requests by Sarantoporo.gr members in the course of the alpha testing process, we have added a menu button at the top-left part of the splash page (Fig. 4.2b). The menu links to a set of pages providing information that users typically want to have about the app: an about page, summarizing who developed the app and for which purpose (Fig. 4.3a), a summary of its capabilities, an FAQ (Frequently Asked Questions) page (Fig. 4.3b), and, last but not least, pages concerning the Privacy policy and the Terms of Use of AppLea (Fig. 4.3c).

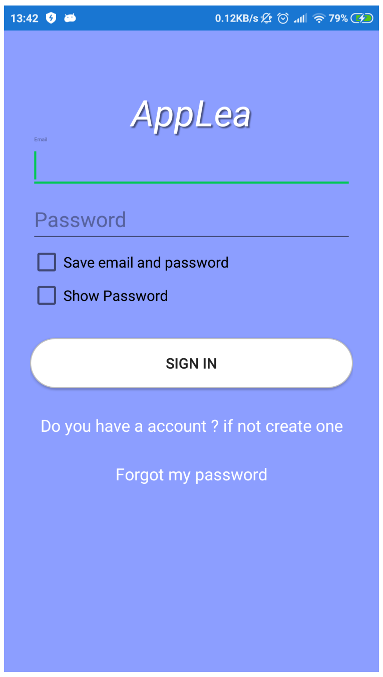


Figure 4.4: Login page – authentication options.

4.2.2. The Login Page

We have added the possibility to sign in a user through his/her mobile phone number. The mobile number rather than an email address serves as username (Fig. 4.4) and the user gets an SMS message to her phone with a one-time code she can use to sign in.

This functionality was requested because many users in the Sarantaporo area, in particular the elderly ones, do not have an email account.

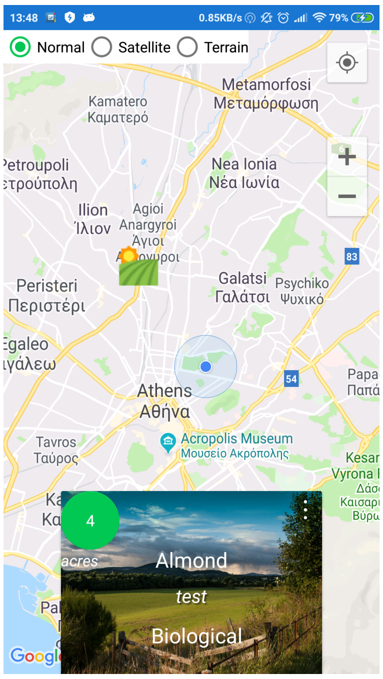
4.2.3. The user profile pages

The user profile pages have been extended, both in terms of information items and detail. The enriched information is organized under three different blocks, which are accessed by respective buttons on the main profile page (see Fig. 4.6a).

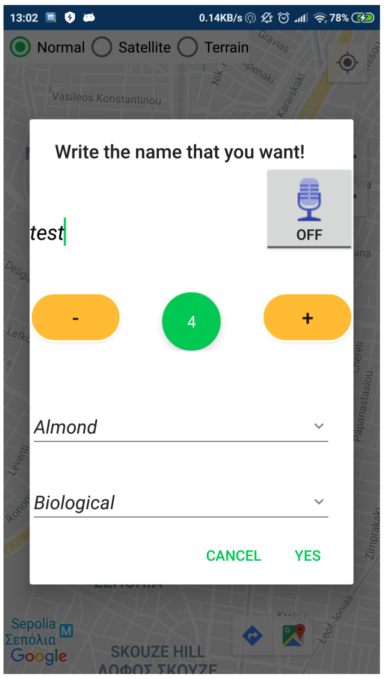
We defer the discussion of the statistics' block for section 4.2.8 and focus on the changes in the other two blocks, the one related to the Field Browser module and the personal information. Three main additions were made in these two blocks, in response to requests from the beta testing team.

- One set of changes related to the Field Browser tool (see Fig. 4.5). On the one hand, we have extended the information that is stored as attributes for the fields/farms a user possesses (or hires). We now distinguish between cultivations that are characterized biological and those that are not since the former demand way more detailed reporting of the farming activities than the latter.

  On the other hand, we have put significant effort to ease the way information is added to the field browser. The field entries have edit menus that let users alter the information relating to them; the user can spell



(a) Field info embedded in a map interface.



Write the name that you want!

test

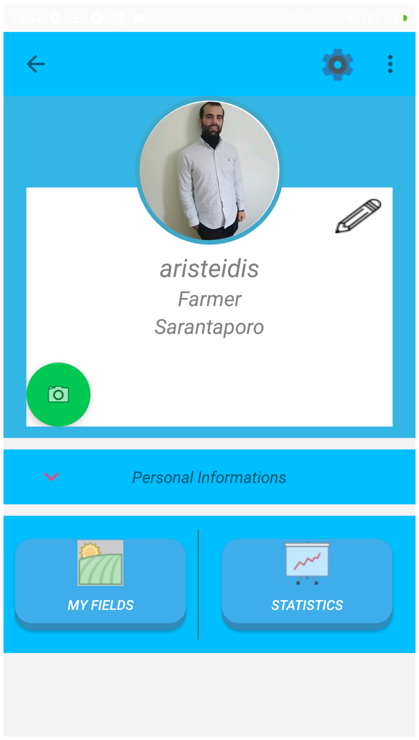
Almond

Biological

CANCEL YES

(b) Editing field information with speech recognition.

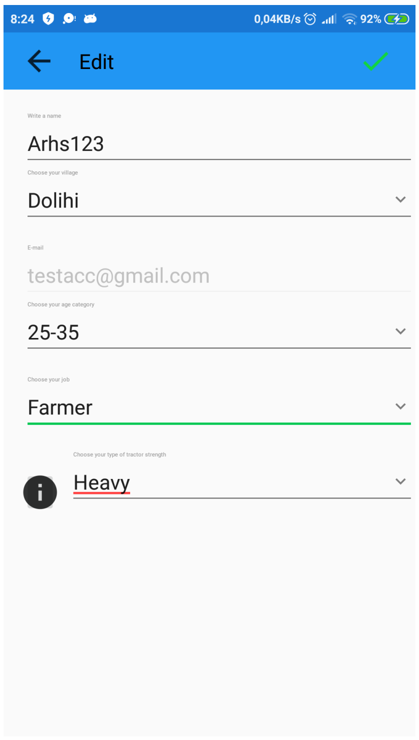
Figure 4.5: The revised interface for adding fields to a user's profile.



aristeidis
Farmer
Sarantaporo

Personal Informations

(a) Main profile page.



Write a name
Arhs123

Choose your village
Dolihi

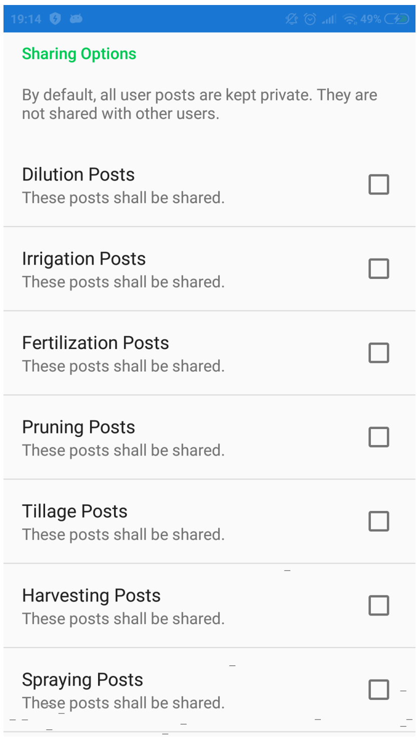
E-mail
testacc@gmail.com

Choose your age category
25-35

Choose your job
Farmer

Choose your type of tractor strength
Heavy

(b) Editing profile info.



Sharing Options

By default, all user posts are kept private. They are not shared with other users.

Dilution Posts ☐
These posts shall be shared.

Irrigation Posts ☐
These posts shall be shared.

Fertilization Posts ☐
These posts shall be shared.

Pruning Posts ☐
These posts shall be shared.

Tillage Posts ☐
These posts shall be shared.

Harvesting Posts ☐
These posts shall be shared.

Spraying Posts ☐
These posts shall be shared.

(c) Privacy settings.

Figure 4.6: Profile page – intensity of tractor use and sharing configuration per type of farming activity log.

out the name of the field, using voice recognition capabilities; and we let all other attributes be chosen out of spinners (drop-down lists) and increase/decrease buttons.

- Our beta testers explicitly requested to be able to get timely periodic reminders about the need to lubricate their tractors. In response, we added a field under the personal information block characterizing the tractor use (heavy, intermediate, light), as shown in Fig. 4.6b.
- Finally, we enhanced the user profile with privacy control options. More specifically, as shown in Fig. 4.6c, we let the user specify which log entries can be shared with other app users and which are not. At the profile pages, the categorization is at the level of activity, the default value being set to non-sharing. There, a user may choose (opt-in) to share *all* log entries relevant to harvesting, turning sharing to the default practice. Yet, (s)he may override this default setting online for a particular post (s)he may want to share or keep private with other users.

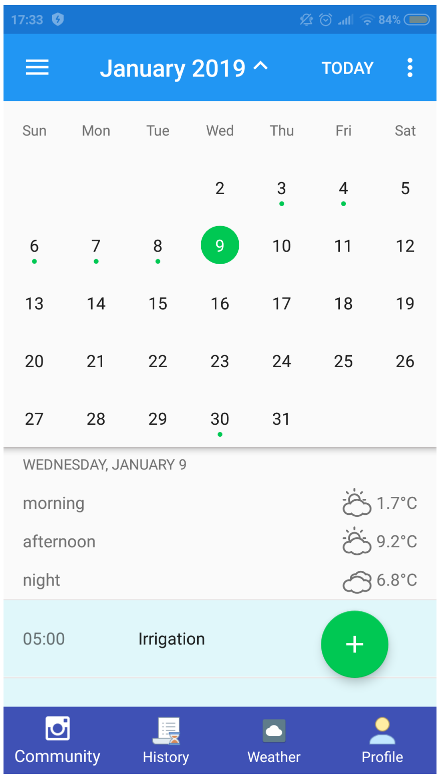
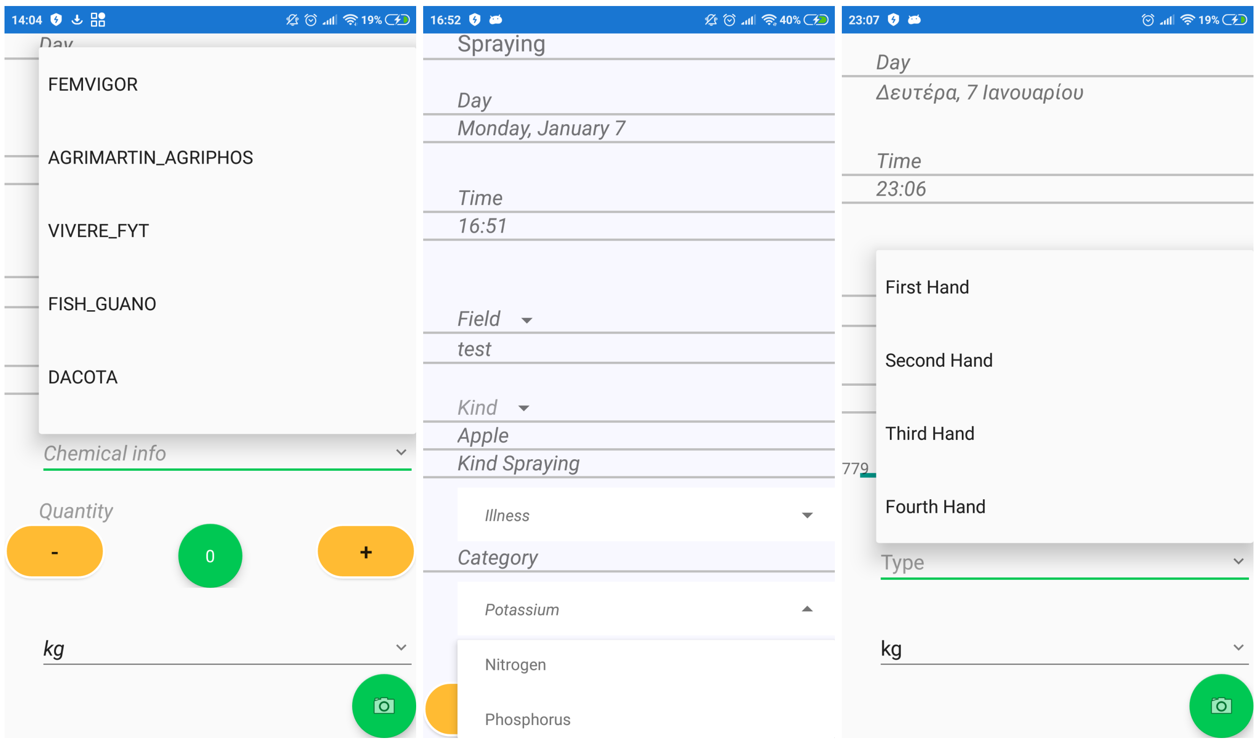


Figure 4.7: The revised AppLea app homepage, featuring a large calendar as asked by beta testers.

4.2.4. The user homepage

This is the page where the mobile app user lands, after signing in the application, and now it display basically the calendar as shown in Fig. 4.7. The changes that were made here are the following.

- The calendar module is promoted to the main element in the page, covering a larger part of its area and displacing the scoreboard extract that appeared in the app version reported in [2]. This change was made in response to an explicit request from the Sarantaporo.gr members who are part of the alpha testing team. We discuss the calendar module in more detail in Sec. 4.2.5.
- We have retained the menu bar at the bottom of the page, as fast entry points to different modules but changed these modules. The revised bar includes links to the user profile, the log history ("history" button), the social layer of the app ("community" button), which lets users socialize and exchange content, even beyond the strict farming context, and the weather module.
- We have added an action button linking to weather information. Although there are various sites around



(a) Fertilization. **(b)** Spraying. **(c)** Harvesting.

Figure 4.8: Log entry forms for different farming activities included in the calendar module.

providing weather information, the inclusion of such information as part of the app, was requested emphatically by the beta testers. We elaborate on the implementation of the weather module in Sec. 4.2.6.

4.2.5. The calendar module

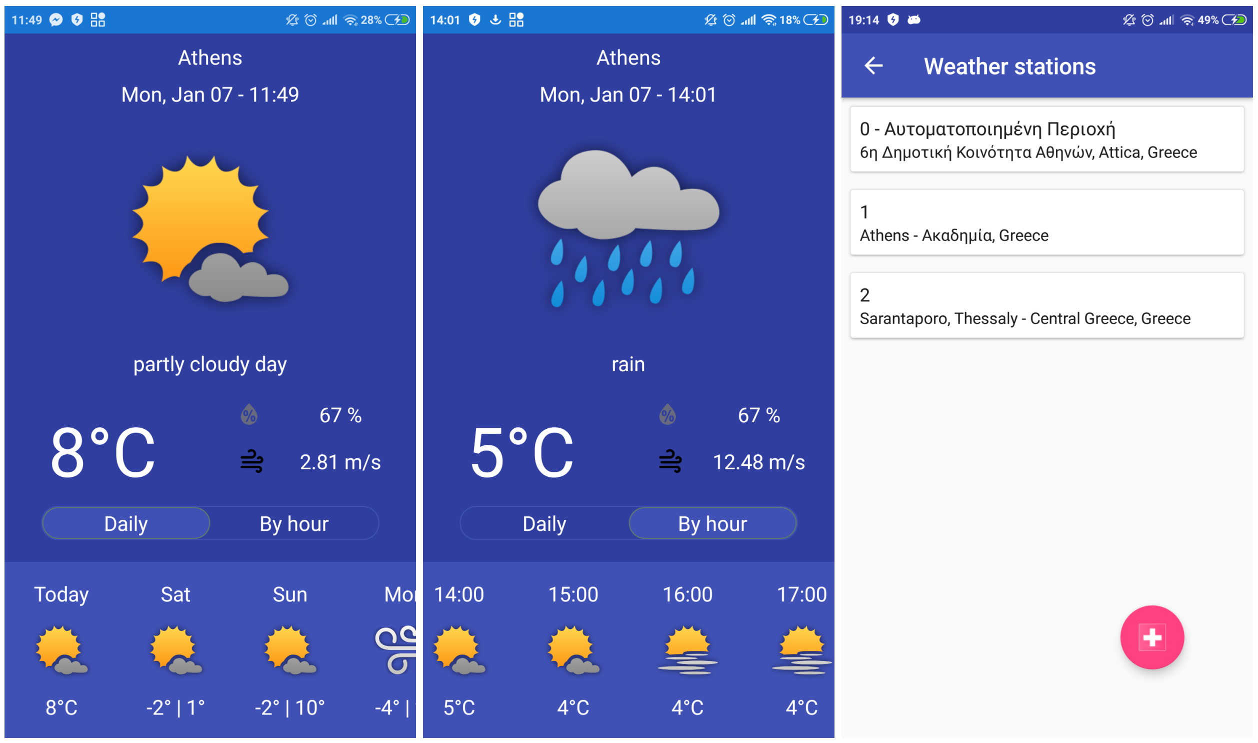
As already mentioned, the importance of this module increased during the testing phase. Many of the comments we received from beta testers concerned the finer details of this module.

Most of those related to the precise format of the forms used for logging the different types of farming activities. In this context, the generic recommendation of the beta testing team was to "minimize user text insertion". This implied, for instance, that drop-down lists should be used instead of text fields, wherever possible. As a result, we got in touch with agronomists and got hold of lists with all brands of fertilizers, pesticides, and weed killers that circulate in the Greek market, so that we could enter them as choices in these forms.

Likewise, we got much advice from beta testers as to what matters most for each activity (for example, in the case of irrigation, what matters is the volume of water consumed rather than time) as well as which measurement units they are most familiar with (e.g., bags of fertilizer). On a similar note, we detailed the harvesting form allowing the user to specify the round of harvesting ("hand" of a harvesting, as the users say, typically ranging from one to four). The new layout of the module is shown in Fig. 4.8.

4.2.6. The weather module

The request from the community members and beta testers is to have weather information as part of the app and, if possible, predictions in the hourly scale. They also informed us that there were two weather stations in the area, set up by independent entities and urged us to use their data.



(a) Weather per day. **(b)** Weather per hour. **(c)** Adding weather stations.

Figure 4.9: Weather module in AppLea.

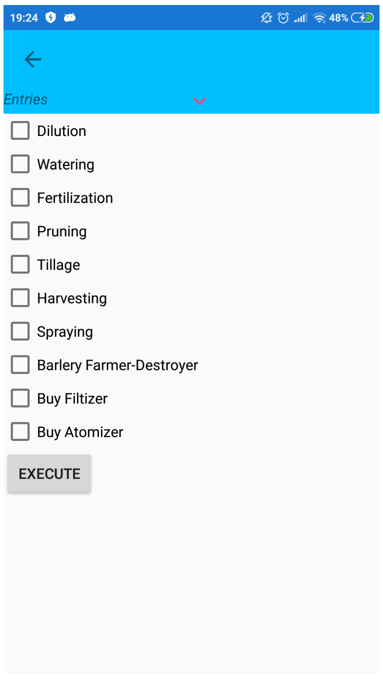
In response to this request, we invested more effort and code in the weather module to present weather information on an hourly basis, as shown in Fig. 4.9. To our surprise, the weather module proved very popular among the beta testers who came back with several suggestions as to how to format the presentation of the weather information.

4.2.7. Log history – processing capabilities

After the calendar module and the activity logging forms, this is the third part of the UI that saw many important changes in response to the suggestions and requests made by the beta testing users.

The main addition here relates to filtering capabilities. The log history presents the user with all entries he/she has made to the log, in reverse chronological order. The user can filter the full list of entries based on different information fields including time, name of the field, type of cultivation, farming activity carried out, as shown in Fig. 4.10. This lets him/her get a quick summary of all activities related to a specific farm, or time interval or iterate on a specific farming activity across all farms. The filtering action can be composite, i.e., the log entries can be filtered simultaneously along more than one filtering criteria.

The main user capability that filtering facilitates is the binding of log entries into reports that can be exported as .pdf files. Such reports can then be printed and be submitted to agronomists to get advice about best practices or identify possible reasons for unfavorable events (e.g., excess weed growth in a cultivation or unexpectedly small crop). Such reports, on per farm basis, can be also delivered to cooperatives, in particular when the cultivation is a biological one (Fig. 4.11).



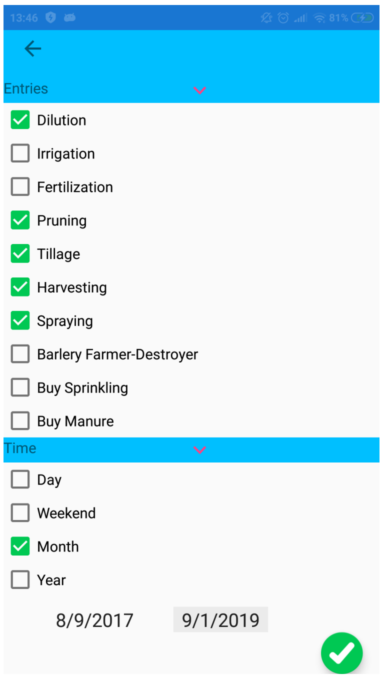
19:24 48%

Entries

- [ ] Dilution
- [ ] Watering
- [ ] Fertilization
- [ ] Pruning
- [ ] Tillage
- [ ] Harvesting
- [ ] Spraying
- [ ] Barlery Farmer-Destroyer
- [ ] Buy Filtizer
- [ ] Buy Atomizer

EXECUTE

(a) Filter by activity.



13:46 81%

Entries

- [x] Dilution
- [ ] Irrigation
- [ ] Fertilization
- [x] Pruning
- [x] Tillage
- [x] Harvesting
- [x] Spraying
- [ ] Barlery Farmer-Destroyer
- [ ] Buy Sprinkling
- [ ] Buy Manure

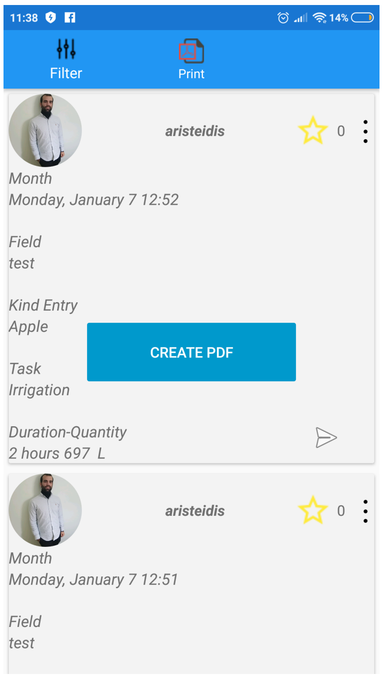
Time

- [ ] Day
- [ ] Weekend
- [x] Month
- [ ] Year

8/9/2017 9/1/2019

(b) Filter by time interval and activity.

Figure 4.10: Applying filters to the log history.



11:38 14%

Filter Print

aristeidis 0

Month
Monday, January 7 12:52

Field
test

Kind Entry
Apple

Task
Irrigation

Duration-Quantity
2 hours 697 L

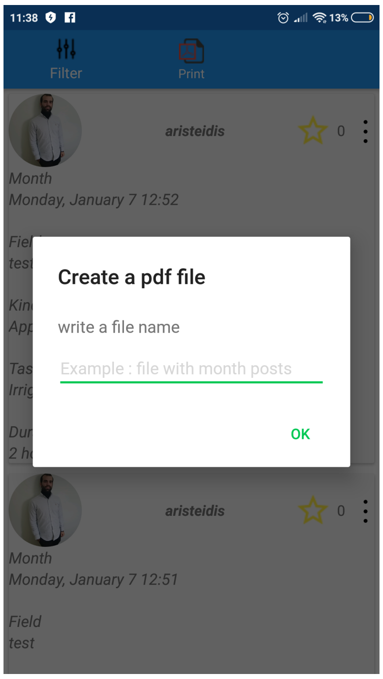
CREATE PDF

aristeidis 0

Month
Monday, January 7 12:51

Field
test

(a) Generate a pdf file.



11:38 13%

Filter Print

Create a pdf file

write a file name

Example : file with month posts

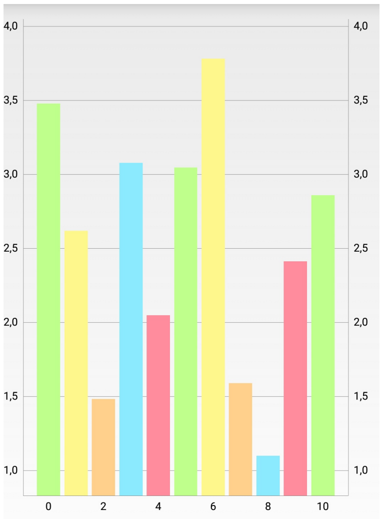
OK

(b) Name and save the generated file.

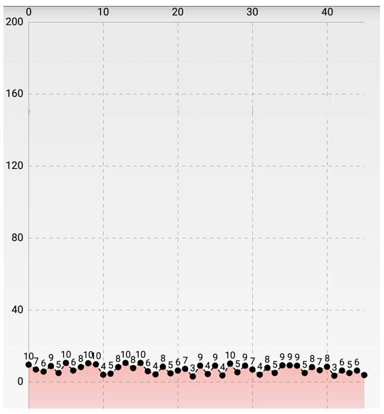
Figure 4.11: Exporting a subset of posts, after filtering, as pdf files.

4.2.8. Farming activity statistics

This is the second important outcome that can emerge out of processing the log history, besides the reports. The produced statistics come in two main forms: text reports and plots. In the first case, the app can return to the



(a) Crop size across fields.



(b) Weather time series across an interval of forty days.

Figure 4.12: Example illustrations produced by the app.

user a specific statistic (s)he may request about its activities; for instance, the aggregate amount of water used in irrigation over a specific time interval or the average consumption of water through the year over a specific field.

In the second case, different types of visualizations are supported: line charts, bar plots, and pie charts are the main ones, the first two shown in Fig. 4.12. This is by far one of the most dynamic parts of the application, which is being updated as more suggestions are made by the alpha and beta testing team as to what kind of statistic might be of value for the users.

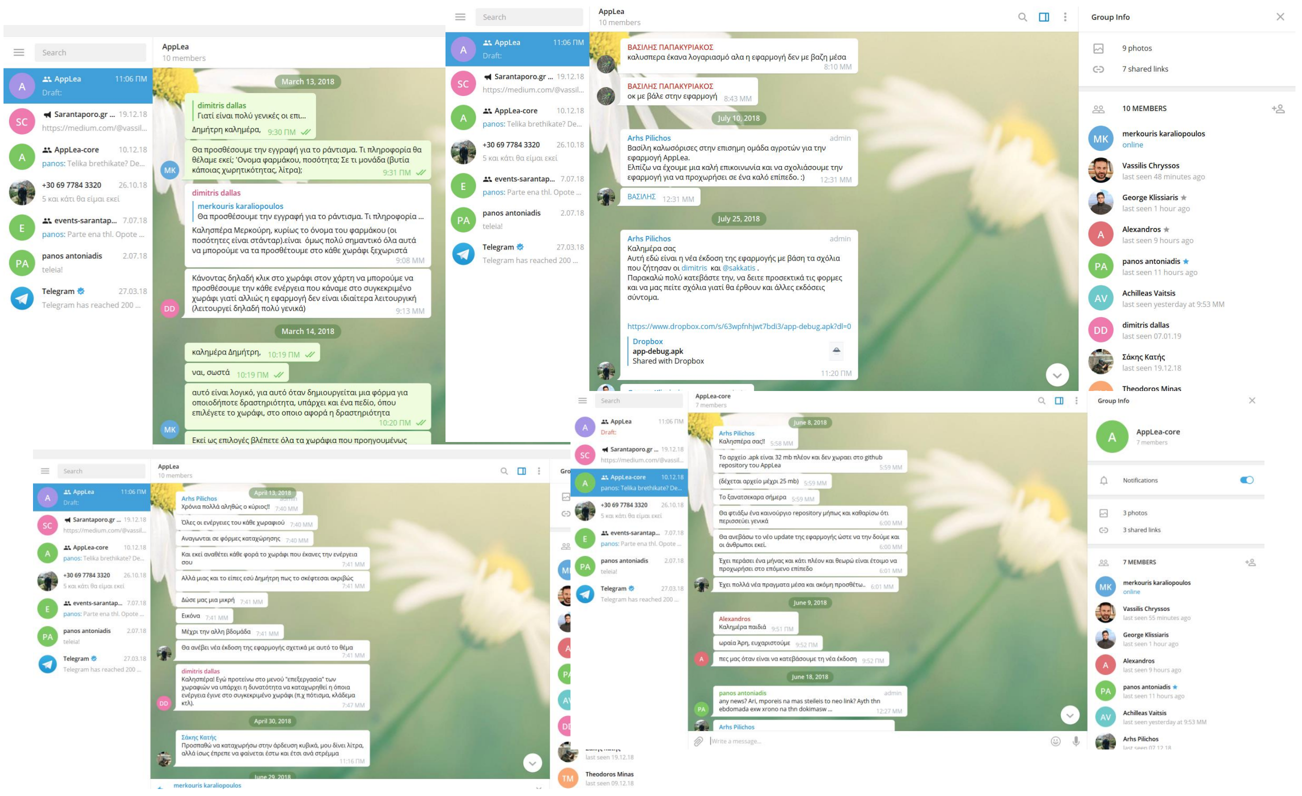
4.3. Iterating on the app impact

Since the original conception of the app and throughput its development, the bar has been set high regarding its potential impact on the CN and the local community. Recalling the original plans for the app and their evolution through the interactions with the community during the first year of the project, the app could ideally serve two purposes:

1. Become itself a reason to join the CN and engage in the community activities.
2. Catalyze synergies with external actors who could contribute to the sustainability of the CN.

The first purpose was identified already during the project preparation phase and projects the role that added-value applications, tailored to the needs of the local community, could play in the evolution of a community network such as the Sarantaporo.gr CN.

The second prospect, which emerged during the November '16 workshop in the Sarantaporo area (see [49] for a detailed report out of that workshop) was to turn the app into an integral component of a broader smart farming system currently under deployment throughout Greece (www.gaiasense.gr) and a catalyzator for a win-win synergy between the non-profit Sarantaporo.gr entity and the for-profit entity that deploys this system (in [2] we describe in detail the motivation and the potential of such a synergy). This was a more ambitious objective, going far beyond the Task 3.4 objectives. In what follows, we iterate on the project outcomes with respect to both directions.


AppLea
10 members

March 13, 2018

dimitris dallas
Γιατί είναι πολύ γενικές οι επι...
Δημήτρη καλημέρα, 9:30 ΠΜ

Θα προσθέσουμε την εγγραφή για το ράντισμα. Τι πληροφορία θα θέλαμε εκεί; 'Ονομα φαρμάκου, ποσότητα; Σε τι μονάδα (βυτία κάποιας χωρητικότητας, λίτρα); 9:31 ΠΜ

dimitris dallas
merkouris karaliopoulos
Θα προσθέσουμε την εγγραφή για το ράντισμα. Τι πληροφορία ...
Καλησπέρα Μερκούρη, κυρίως το όνομα του φαρμάκου (οι ποσότητες είναι στάνταρ).είναι όμως πολύ σημαντικό όλα αυτά να μπορούμε να τα προσθέτουμε στο κάθε χωράφι ξεχωριστά 9:08 ΜΜ

Κάνοντας δηλαδή κλικ στο χωράφι στον χάρτη να μπορούμε να προσθέσουμε την κάθε ενέργεια που κάναμε στο συγκεκριμένο χωράφι γιατί αλλιώς η εφαρμογή δεν είναι ιδιαίτερα λειτουργική (λειτουργεί δηλαδή πολύ γενικά) 9:13 ΜΜ

March 14, 2018

καλημέρα Δημήτρη, 10:19 ΠΜ
ναι, σωστά 10:19 ΠΜ
αυτό είναι λογικό, για αυτό όταν δημιουργείται μια φόρμα για οποιοδήποτε δραστηριότητα, υπάρχει και ένα πεδίο, όπου επιλέγετε το χωράφι, στο οποιο αφορά η δραστηριότητα 10:20 ΠΜ
Εκεί ως επιλογές βλέπετε όλα τα χωράφια που προηγουμένως

AppLea
10 members

ΒΑΣΙΛΗΣ ΠΑΠΑΚΥΡΙΑΚΟΣ
καλυσπερα έκανα λογαριασμό αλα η εφαρμογή δεν με βαζη μέσα 8:10 ΜΜ

ΒΑΣΙΛΗΣ ΠΑΠΑΚΥΡΙΑΚΟΣ
οκ με βάλε στην εφαρμογή 8:43 ΜΜ

July 10, 2018

Arhs Pilichos admin
Βασίλη καλωσόρισες στην επισημη ομάδα αγροτών για την εφαρμογή AppLea.
Ελπίζω να έχουμε μια καλή επικοινωνία και να σχολιάσουμε την εφαρμογή για να προχωρήσει σε ένα καλό επίπεδο. :) 12:31 ΜΜ

ΒΑΣΙΛΗΣ 12:31 ΜΜ

July 25, 2018

Arhs Pilichos admin
Καλημέρα σας
Αυτή εδώ είναι η νέα έκδοση της εφαρμογής με βάση τα σχόλια που ζήτησαν οι dimitris και @sakkatis .
Παρακαλώ πολύ κατεβάστε την, να δειτε προσεκτικά τις φορμες και να μας πείτε σχόλια γιατί θα έρθουν και άλλες εκδόσεις σύντομα.

https://www.dropbox.com/s/63wpfnhjwt7bdi3/app-debug.apk?dl=0
Dropbox
app-debug.apk
Shared with Dropbox
11:20 ΠΜ

Group Info
9 photos
7 shared links
10 MEMBERS
merkouris karaliopoulos — online
Vassilis Chryssos — last seen 48 minutes ago
George Klissiaris — last seen 1 hour ago
Alexandros — last seen 9 hours ago
panos antoniadis — last seen 11 hours ago
Achilleas Vaitsis — last seen yesterday at 9:53 MM
dimitris dallas — last seen 07.01.19
Σάκης Κατής — last seen 19.12.18
Theodoros Minas — last seen 09.12.18

AppLea
10 members

April 13, 2018

Arhs Pilichos admin
Χρόνια πολλά αληθώς ο κύριος!! 7:40 ΜΜ
Όλες οι ενέργειες του κάθε χωραφιού 7:40 ΜΜ
Αναγωνται σε φόρμες καταχώρησης 7:40 ΜΜ
Και εκεί αναθέτει κάθε φορά το χωράφι που έκανες την ενέργεια σου 7:41 ΜΜ
Αλλά μιας και το είπες εσύ Δημήτρη πως το σκέφτεσαι ακριβώς 7:41 ΜΜ
Δώσε μας μια μικρή 7:41 ΜΜ
Εικόνα 7:41 ΜΜ
Μέχρι την αλλη βδομάδα 7:41 ΜΜ
Θα ανέβει νέα έκδοση της εφαρμογής σχετικά με αυτό το θέμα 7:41 ΜΜ

dimitris dallas
Καλησπέρα! Εγώ προτείνω στο μενού "επεξεργασία" των χωραφιών να υπάρχει η δυνατότητα να καταχωρηθεί η όποια ενέργεια έγινε στο συγκεκριμένο χωράφι (π.χ πότισμα, κλάδεμα κτλ). 7:47 ΜΜ

April 30, 2018

Σάκης Κατής
Προσπαθώ να καταχωρήσω στην άρδευση κυβικά, μου δίνει λίτρα, αλλά ίσως έπρεπε να φαίνεται έστω και έτσι ανά στρέμμα 11:16 ΠΜ

June 29, 2018
merkouris karaliopoulos

AppLea-core
7 members

June 8, 2018

Arhs Pilichos
Καλησπέρα σας!! 5:58 ΜΜ
Το αρχείο .apk είναι 32 mb πλέον και δεν χωραει στο github repository του AppLea 5:59 ΜΜ
(δέχεται αρχείο μέχρι 25 mb) 5:59 ΜΜ
Το ξανατσεκαρα σήμερα 5:59 ΜΜ
Θα φτιάξω ένα καινούργιο repository μήπως και καθαρίσω ότι περισσεύει γενικά 6:00 ΜΜ
Θα ανεβάσω το νέο update της εφαρμογής ώστε να την δούμε και οι άνθρωποι εκεί. 6:00 ΜΜ
Έχει περάσει ένα μήνας και κάτι πλέον και θεωρώ είναι έτοιμο να προχωρήσει στο επόμενο επίπεδο 6:01 ΜΜ
Έχει πολλά νέα πραγματα μέσα και ακόμη προσθέτω.. 6:01 ΜΜ

June 9, 2018

Alexandros
Καλημέρα παιδιά 9:51 ΠΜ
ωραία Άρη, ευχαριστούμε 9:52 ΠΜ
πες μας όταν είναι να κατεβάσουμε τη νέα έκδοση 9:52 ΠΜ

June 18, 2018

panos antoniadis admin
any news? Ari, mporeis na mas steileis to neo link? Ayth thn ebdomada exw xrono na thn dokimasw ... 12:27 ΜΜ

Arhs Pilichos

Group Info
AppLea-core
7 members
Notifications
3 photos
3 shared links
7 MEMBERS
merkouris karaliopoulos — online
Vassilis Chryssos — last seen 55 minutes ago
George Klissiaris — last seen 1 hour ago
Alexandros — last seen 9 hours ago
panos antoniadis — last seen 11 hours ago
Achilleas Vaitsis — last seen yesterday at 9:53 MM
Arhs Pilichos — last seen 07.12.18

Figure 4.13: Telegram groups used as channels of remote interaction with the alpha- and beta-testing team.

4.3.1. Interest of the local community in the app

There are both positive and negative signals in this context, fortunately with the positive one outnumbering the negative. The following list discusses the positive points and signals we got from the community.

- The active engagement of community members throughout the application design and development. From their early comments and remarks, which led us to reshape and focus the app (see [2]), our exchanges with them in physical and remote virtual meetings, till their more regular and structural feedback in the context of the alpha- and beta-testing process, they have embraced the app and given very useful feedback to it. The original beta testing team was formed in the March workshop in Flabouro. During the July workshop it was expanded with a new member, increasing the number of those sending feedback about the app. Telegram groups (Fig. 4.13) were used for the exchanges with them and a link thereby pointed to the latest update of the app for easy download.
- The beta testers showed that they appreciated the presentation of the app in the EuCNC conference in Ljubljana [50] and the positive comments we had received there. These were communicated to the people through posts of pictures and text in Telegram groups and appeared to give them further incentives since their feedback was intensified the couple of weeks that followed.
- Statistics that can be drawn from the backend of the app (see Fig. 4.14 and Fig. 4.15 suggest that the cycle of app users beyond those who are "officially" members of the beta testing team has opened early. Fig. 4.14 shows the distribution of users per country, that is obviously concentrated in Greece, apart from netCommons members logging for testing or curiosity. Fig. 4.15a plots the number of different active users of the app on a daily, weekly and monthly basis. The rise of the curves is aligned with the March workshop in Flabouro and the set-up of the beta testing team. Fig. 4.15b shows the activity, which is obviously related to the number of users, but allows based on the number of events logged at the backend regarding the app use.

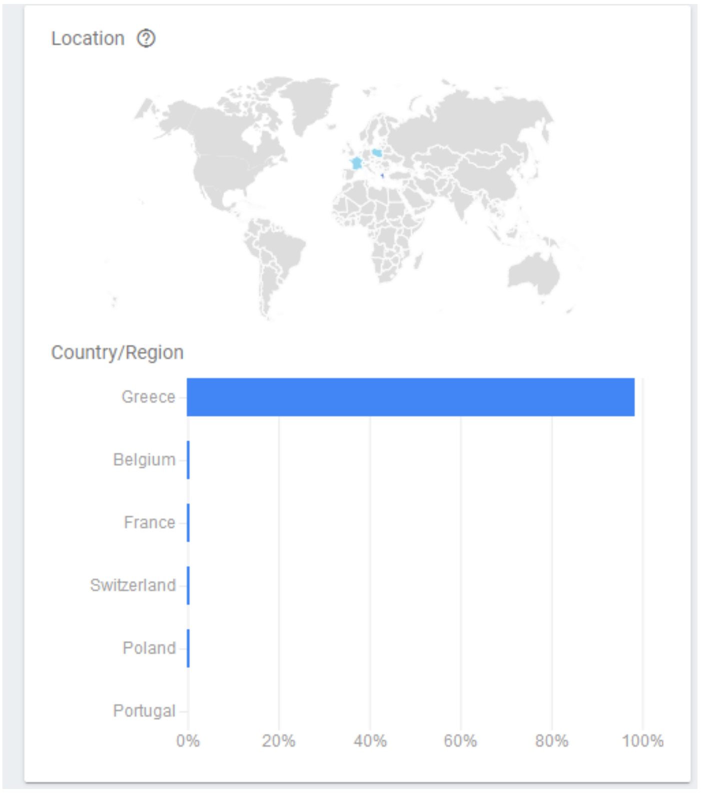
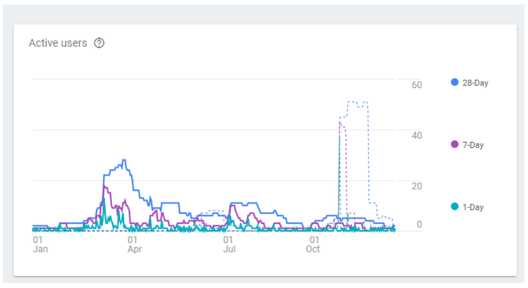
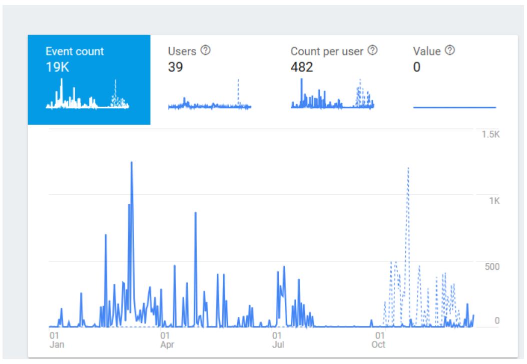


Figure 4.14: Distributions of user per country.

These statistics imply that the app has been promoted and spread by the word of mouth, before any promotional action on our side.



(a) Active users time series.



(b) Dashboard card with a more detailed view.

Figure 4.15: Information about use of the AppLea, as extracted from the analytics of the Firebase backend: active app users, activity, and geographic distribution of users.

On the negative side, the effort to generalize the use of the app has had to cope with a couple of facts that, although easily predicted, represent generic stumbling blocks that slow down innovation and adoption of tools like AppLea.

- First, the use of electronics means to log down farming activities is anything but common practice among the Greek agricultural population. Recall, that this counts as the main bottleneck towards the wide spread of smart farming processes in Greece, e.g., in the case of the gaiasense system, which we discussed in D3.4 [2].

- Second, the understanding about Internet and what can someone do in (with) it is partial and biased, an alternative symptom of what we refer to as "digital divide". Most of the CN users, including the younger members understand Internet as a synonym of social media; hence, a means for entertainment and socializing. Few members appreciate the opportunity of comparing prices for agricultural products and having the chance to save money when making purchases. But this is far from anticipating that the network and the mobile device can replace the pen and the notepad, as business tools.

4.3.2. The app as an enabler of CN synergies

During the first two years of the project lifetime, some good steps in this direction were taken. First, in the workshop of November '16 in the Sarantaporo area, the prospects from a synergy of the CN with the commercial gaiasense system were identified, noted down, and analyzed at first hand. Then, in the February '17 meeting in the Neuropublic premises (a main stakeholder in the gaiasense system), we discussed some experimentation steps that could validate the operation of the system over the CN and the app use as a means to collect the data on farming activities. By summer 2017, an innovation proposal was submitted with the participation of Sarantaporo.gr, Neuropublic and Athens University of Economics and Business to a national research and innovation program funding synergies between academic/research institutions and private sector.

Unfortunately, these efforts did not result in tangible outcomes, at least within the time frame of the netCommons project. Two were the main reasons for this, in our opinion.

First, it was quite hard to align the agendas of the two actors, i.e., Neuropublic S.A., having the main responsibility for the gaiasense system, and Sarantaporo.gr CN. For gaiasense, the main effort during the project period was devoted to pilot deployments of the system in cooperation with agricultural cooperatives. For the Sarantaporo.gr CN, the primary concern was the upgrade of their network equipment and the transition to a new subscription scheme. Although a synergy between a non-profit entity (Sarantaporo.gr) and a for-profit one (gaiasense system stakeholders) was innovative and promised mutual benefits for both parties, it was not a priority for either of them. The rejection of the submitted innovation proposal marked a missed opportunity to align the two agendas and make progress on this front.

Second, there has been a sort of *trust gap* between the two main entities, mainly on the side of the Sarantaporo.gr CN. Being a non-profit entity, with political values and concerns about privacy and data management policies, it faced with scepticism the data management practices of the gaiasense system. In this respect, there were thoughts about treating the farming data from users in the area as a community asset and catering for local storage, but these stood at odds with the cloud-based architecture of the gaiasense system.

For sure, these two reasons are interrelated, i.e., the trust gap is most probably one reason why the synergy was not pursued more actively by the two parties. In any case, the AppLea app became the vehicle to explore new opportunities, earlier unforeseen, for the Sarantaporo.gr CN. As the CN will continue seeking paths towards a long-term sustainable operation, and the gaiasense will need ways to engage farmers in logging their farming activities, the seeds that were planted in the project may have more chances to bear fruit.

4.4. A final note: open source backend alternatives to Firebase

As discussed in more detail in D3.4 [2], the backend server in AppLea has been implemented on Firebase, a powerful mobile and web application development platform led by Google. There are several reasons for this choice. First, Firebase is a powerful tool with many capabilities, including functionality leveraged from other Google services, which have been key to the AppLea functionality, such as the real-time synchronization of mobile devices. Second, by the time this report is written, it is an active project that continues to innovate, ranking first in the preferences of the app developers' community. Third, it features an attractive pricing policy (free up to 1 GB storage space and 10 GB downloads monthly, 25 USD per month for more intensive usage) summarized in Table 4.1.

Backend platform	Pricing options
Firebase	Free Plan, Flame Plan for 25 USD per month, Blaze Plan (Pay as you go)
Back4app	Free Plan, Starter/Basic/Intermediate/Standard/Advanced Plans for 4.99/14.99/34.99/49.99/99.99 USD per month, resp.
Kuzzle	Free Plan, Starter/Business for 2.000/5000 USD per month resp. custom pricing for Enterprise plans

Table 4.1: Pricing options offered by competing backend platforms.

On the other hand, Firebase raises some concerns with respect to the management of the application data. These concerns were brought up by members of the Sarantaporo.gr in the alpha testing team during the experimentation phase this year. First of all, it is not an open source platform. The open source feature is more compatible with the spirit of community networks but also reduces the risk of vendor lock-in practices. Second, it does not cater for self hosting, which would allow a deployment of the backend service locally at the CN premises. Instead, the platform services reside, operate and are being updated in the cloud. One of the advantages of self-hosting is higher transparency and control over the data residence and use. On a related note, it prevents possible conflicts with the EU General Data Protection Regulation (GDPR) mandates; for instance, storing personally identifiable information of European citizens in servers based in the United States.

In what follows, we address the question of whether there are backend platforms that satisfy both the open source and self-hosting requirements without sacrificing considerably the desirable Firebase performance features. Out of more than ten backend alternatives, four appear to satisfy those two requirements: Parse, Back4app, Kuzzle and Hoodie. Here are our findings and considerations on them.

Parse is a mobile and web application development platform, with a strong developer community behind it that supports the technology. It supports push notifications and role-based access control, data security and analytics. It provides an easy interface to create, share and filter entries but does not provide the real-time synchronization features of Firebase. Parse comes as software modules that can be downloaded on local premises; apps making use of Parse will have to be hosted in some platform.

Back4app is exactly such a Parse hosting platform. On the positive side, the platform features automated provisioning and scaling of Parse Server applications, backup and recovery features, web-based management tools and a user-friendly dashboard interface. On the negative side, it is weak with respect to documentation.

Kuzzle is designed for web, mobile, and even, IoT applications. It supports real-time pub/sub, fast search, and geospatial queries through a scalable server, a multi-protocol API and a set of plugin engines. On the downside, it is a young project and there is not much production experience with it (as the case with Firebase is).

Hoodie, our last option has the main advantage of simplicity. It is an offline JavaScript backend, which is based on a so-called "no-backend" technology. It has several attractive features but, as with Kuzzle, it has not yet gained the trust of a critical mass of developers.

In terms of pricing options, the three platforms (Parse and Hoodie do not charge as such, they are downloaded for free but apps drawing on them have to pay for hosting) compare according to Table 4.1.

Overall, the open source tools are either relatively new, without a large app base (as in the case of Hoodie or Kuzzle), or they lag (mainly) in terms of device synchronization features with respect to Firebase. In general, an option like Back4app might turn out to be the favorite compromise if the requirement for retaining user data locally to the CN premises become too strict and outweighs the performance advantages of Firebase. For the needs of the project, however, Firebase provided a "safe" firm reference for the backend, letting us focus on the requirements of the already demanding front end functionality, which had to account for the limited experience of most of the intended users with mobile apps like AppLea.

5. The Participatory Methodology Booklet

As a prosecution of the work of T3.1, and as a consequence of continued work and activity beyond the sheer timeframe of the project, we report here a "scaled down," updated version of the Multidisciplinary methodology for the participatory design of applications for Community Networks. The methodology was introduced in D3.3 [6], and first evaluated in D3.6 [3]. In these last months it was used with and by various communities.

In order to increase the visibility of this result and to support the future diffusion of this methodology beyond the end of netCommons, NetHood decided to re-publish the methodology into a stand-alone booklet. The booklet simplifies the concepts, makes them more readable than the description we published in the deliverables and is the result of the continuous improving process we carried on during the project, based on the feedback received during the parallel software development processes in this work package. The booklet can be found in Appendix A to this document, and we consider it an integral part of netCommons work. The booklet format does not match the one of the deliverable; for better readability we point the reader to the on-line pdf version[1].

The booklet is part of future activities by NetHood, and will be in a constant process of improvement and extension in the framework of an online "studio" space,[2] which will be developed in the next years both online and offline, as facilitator of processes organized in the physical spaces of NetHood, like L200[3] .

[1]Current version is 0.6 available at: https://www.netcommons.eu/sites/default/files/pd-methodology-booklet-v0.6.pdf together with the "cards" to be used in the development phase https://www.netcommons.eu/sites/default/files/pd-methodology-booklet-v0.6_cards.pdf.

[2]See http://nethood.org/studio/.

[3]See http://langstrasse200.ch.

6. Conclusions

This deliverable concludes the open source development that was one of the goals of netCommons. During the three year of the project we continued the development of Cloudy and PeerStreamer-ng and we completed the development of AppLea. In this process, we constantly worked with, and within communities, and we received countless pieces of feedback from various community networks. WP3 maximised the effort to disseminate, deploy and co-create open source applications with communities. With three lines of research we were able to study both the adaptation and deployment of existent applications and the creation of new ones from scratch. We worked with CNs of three different countries, we participated to many of the events they organized (more details are reported in D6.3) and we obtained a very high involvement, also considering that the project did not have budget to be allocated to community members. All the people that contributed to our project and used our software were motivated only by their own interest in the software and the methodology we proposed, and they produced appreciation letters to show their interest and enthusiasm for netCommons results.

Furthermore, from WP3 a number of scientific results and new development emerged that open new directions for future research and projects. We tested new technologies to support edge computing in mesh networks, we were the first ones to propose and test the adoption of blockchain technology in the field of community and mesh networks, we adapted and improved distributed video streaming to the community network case, and we stimulated the use and adoption of smart farming techniques. All these activities produced high impact research papers and will be the basis for future collaborations with communities after the end of netCommons. In some cases, we were able to stimulate the interest of existing start-ups (like AmmbrTech) or to favour the initiative of community network members to create their own (like the Turnantenna).

Bibliography

[1] N. Facchi, F. Freitag, L. Maccari, P. Micholia, and F. Zanini, “Release of All Open Source Software for all Applications (v1),” netCommons Deliverable D3.2, Dec. 2016. http://netcommons.eu/?q=content/release-new-open-source-software-all-applications-v1

[2] L. Maccari, L. Baldesi, N. Facchi, R. Lo Cigno, F. Freitag, L. Navarro, R. Messeguer, M. Karaliopoulos, P. Micholia, and A. Pilichos, “Release of All Open Source Software for all Applications (v2),” netCommons Deliverable D3.4, Mar. 2018. https://netcommons.eu/?q=content/release-new-open-source-software-all-applications-v2

[3] A. Papageorgiou, P. Antoniadis, M. Karaliopoulos, G. Klissiaris, V. Chryssos, L. Maccari, R. Lo Cigno, F. Freitag, and L. Navarro, “Deployment experiences with a Multi-Disciplinary approach,” netCommons Deliverable D3.6, Oct. 2018. https://www.netcommons.eu/sites/default/files/d3.6_v1.0.pdf

[4] R. Baig, F. Freitag, and L. Navarro, “Cloudy in guifi. net: Establishing and sustaining a community cloud as open commons,” *Future Generation Computer Systems*, vol. 87, Oct. 2018.

[5] R. Baig-Viñas, L. Navarro, and R. Roca-i Tió, *Multiple Dimensions of Community Network Scalability(book chapter in ”The community network manual: how to build the Internet yourself”).* FGV Direito Rio, Nov 2018, pp. 133–158. http://bibliotecadigital.fgv.br/dspace/handle/10438/25696

[6] P. Antoniadis, I. Apostol, and A. Papageorgiou, “Multi-Disciplinary Methodology for Applications Design for CNs, including Design Guidelines and Adoption Facilitation (v2),” netCommons Deliverable D3.3, Mar. 2018. https://www.netcommons.eu/?q=content/multi-disciplinary-methodology-applications-design-cns-including-design-guidelines-and-0

[7] L. Navarro, R. Baig, and F. Freitag, “Report on the Governance Instruments ant their Application to CNs (v2),” netCommons Deliverable D1.4, Dec. 2017. https://netcommons.eu/?q=content/report-governance-instruments-and-their-application-cns-v2

[8] A. M. Khan, F. Freitag, V. Vlassov, and P. H. Ha, “Demo abstract: Towards IoT service deployments on edge community network microclouds,” in *IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, April 2018.

[9] K. Batbayar, E. Dimogerontakis, R. Meseguer, L. Navarro, E. Medina, and R. M. Santos, “The rimo gateway selection approach for mesh networks: Towards a global internet access for all,” *Proceedings*, vol. 2, no. 19, 2018. http://www.mdpi.com/2504-3900/2/19/1258

[10] K. Batbayar, R. Meseguer, L. Navarro, R. Sadre, and E. Dimogerontakis, “Collaborative informed gateway selection in large-scale and heterogeneous networks,” in *IFIP/IEEE International Symposium on Integrated Network Management (IM)*, April 2019.

[11] M. Selimi, L. Cerdà-Alabern, F. Freitag, L. Veiga, A. Sathiaseelan, and J. Crowcroft, “A lightweight service placement approach for community network micro-clouds,” *Journal of Grid Computing (to appear)*, 2018.

[12] M. Selimi, A. Lertsinsrubtavee, A. Sathiaseelan, L. Cerdà-Alabern, and L. Navarro, “PiCasso: Enabling Information-Centric Multi-tenancy at the Network’s Edge,” *Submitted to Elsevier Computer Networks*, Jan. 2019.

[13] P. Antoniadis, J. Martignoni, L. Navarro, and P. Dini, *Complementary Networks Meet Complementary Currencies: Guifi.net Meets Sardex.net*. FGV Direito Rio, Nov 2018, pp. 189–222. http://bibliotecadigital.fgv.br/dspace/handle/10438/25696

[14] A. R. Kabbinale, E. Dimogerontakis, M. Selimi, A. Ali, L. Navarro, and A. Sathiaseelan, "Blockchain for economically sustainable wireless mesh networks," *arXiv preprint arXiv:1811.04078*, 2018. https://arxiv.org/abs/1811.04078

[15] F. Freitag, "On the collaborative governance of decentralized edge microclouds with blockchain-based distributed ledgers," in *1st International Workshop on Block Chain Technologies 4 Multi-Agent Systems (BCT4MAS)*, December 2018.

[16] L. Navarro, I. Castro, A. Sathiaseelan, E. Dimogerontakis, M. Selimi, F. Freitag, and R. Baig, "Blockchain models for universal connectivity," *Under review in Telecommunications Policy Journal*, vol. -, no. -, 2018. https://www.journals.elsevier.com/telecommunications-policy

[17] M. Selimi, A. R. Kabbinale, A. Ali, L. Navarro, and A. Sathiaseelan, "Towards blockchain-enabled wireless mesh networks," in *Proceedings of the 1st Workshop on Cryptocurrencies and Blockchains for Distributed Systems, CRYBLOCK@MobiSys 2018, Munich, Germany, June 15, 2018*. ACM, 2018, pp. 13–18. https://doi.org/10.1145/3211933.3211936

[18] L. Ghiro, L. Maccari, and R. Lo Cigno, "Proof of networking: Can blockchains boost the next generation of distributed networks?" in *IEEE/IFIP Wireless On-demand Network systems and Services Conference (WONS)*, Isola 2000, France, February 2018.

[19] C. Rey-Moreno, W. Tucker, N. Bidwell, Z. Roro, J. S. Masbulele, and J. Simó-Reigadas, "Experiences, challenges and lessons from rolling out a rural wifi mesh network," in *3rd ACM Symposium on Computing for Development, ACM DEV*, 2013.

[20] D. Vega, R. Baig, L. Cerdà-Alabern, E. Medina, R. Meseguer, and L. Navarro, "A technological overview of the guifi.net community network," *Computer Networks*, vol. 93, Part 2, pp. 260 – 278, 2015. //www.sciencedirect.com/science/article/pii/S1389128615003436

[21] M. Selimi, A. M. Khan, E. Dimogerontakis, F. Freitag, and R. P. Centelles, "Cloud services in the guifi.net community network," *Computer Networks*, vol. 93, no. Part 2, pp. 373–388, 2015. http://dx.doi.org/10.1016/j.comnet.2015.09.007

[22] P. Srisuresh, J. Kuthan, J. Rosenberg, A. Molitor, and A. Rayhan, "Middlebox communication architecture and framework," RFC 3303 (Informational), Internet Engineering Task Force, Tech. Rep. 3303, aug 2002.

[23] E. Dimogerontakis, R. Meseguer, and L. Navarro, "Internet access for all: Assessing a crowdsourced web proxy service in a community network," in *International Conference on Passive and Active Network Measurement (PAM)*, 2017.

[24] M. R. Casters, R. Bouman, and J. van. Dongen, *Pentaho Kettle Solutions: Building Open Source ETL Solutions with Pentaho Data Integration*. Wiley, 2010.

[25] L. Maccari and R. Lo Cigno, "Monitoring Instruments for CNs (v1)," netCommons deliverable D2.5, Dec. 2016. http://netcommons.eu/?q=content/monitoring-instruments-cns-v1

[26] ——, "Monitoring Instruments for CNs (v2)," netCommons deliverable D2.7, Dec. 2017. https://netcommons.eu/?q=content/monitoring-cns-report-experimentations-cns-v2

[27] M. Selimi, A. M. Khan, E. Dimogerontakis, F. Freitag, and R. P. Centelles, "Cloud services in the guifi.net community network," *Computer Networks*, vol. 93, Part 2, pp. 373 – 388, 2015. //www.sciencedirect.com/science/article/pii/S1389128615003175

[28] R. Baig, R. P. Centelles, F. Freitag, and L. Navarro, "On edge microclouds to provide local container-based services," in *Global Information Infrastructure and Networking Symposium, GIIS*, 2017, pp. 31–36. https://doi.org/10.1109/GIIS.2017.8169801

[29] R. Baig, F. Freitag, and L. Navarro, "Cloudy in guifi.net: Establishing and sustaining a community cloud as open commons," *Future Generation Computer Systems*, vol. 87, pp. 868–887, 2018. http://www.sciencedirect.com/science/article/pii/S0167739X1732856X

[30] A. Lertsinsrubtavee, M. Selimi, A. Sathiaseelan, L. Cerdà-Alabern, L. Navarro, and J. Crowcroft,

"Information-centric multi-access edge computing platform for community mesh networks," in *1st ACM SIGCAS Conference on Computing and Sustainable Societies (COMPASS)*, 2018. http://doi.acm.org/10.1145/3209811.3209867

[31] A. Lertsinsrubtavee, A. Ali, C. Molina-Jimenez, A. Sathiaseelan, and J. Crowcroft, "Picasso: A lightweight edge computing platform," in *2017 IEEE 6th International Conference on Cloud Networking (CloudNet)*, Sept 2017.

[32] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking named content," in *5th International Conference on Emerging Networking Experiments and Technologies (CoNEXT)*, 2009. http://doi.acm.org/10.1145/1658939.1658941

[33] "RIFE: Architecture for an Internet for everybody," accessed: 2018-02-10. https://rife-project.eu/

[34] "Scalable and Adaptive Internet Solutions (SAIL)," accessed: 2018-02-10. http://www.sail-project.eu

[35] "Docker technology," https://www.docker.com/what-docker, accessed: 2018-02-10.

[36] C.-A. Sarros, A. Lertsinsrubtavee, C. Molina-Jimenez, K. Prasopoulos, S. Diamantopoulos, D. Vardalis, and A. Sathiaseelan, "Icn-based edge service deployment in challenged networks," in *Proceedings of the 4th ACM Conference on Information-Centric Networking*, ser. ICN '17. New York, NY, USA: ACM, 2017, pp. 210–211. http://doi.acm.org/10.1145/3125719.3132096

[37] R. Baig, L. Dalmau, R. Roca, L. Navarro, F. Freitag, and A. Sathiaseelan, "Making community networks economically sustainable, the guifi.net experience," in *Workshop on Global Access to the Internet for All*, 2016. http://doi.acm.org/2940157.2940163

[38] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008.

[39] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum Project Yellow Paper*, vol. 151, pp. 1–32, 2014.

[40] Ethereum Developers, "Clique PoA protocol & Rinkeby PoA testnet." https://github.com/ethereum/EIPs/issues/225

[41] A. Najjar, C. Gravier, X. Serpaggi, and O. Boissier, "Modeling User Expectations amp;amp; Satisfaction for SaaS Applications Using Multi-agent Negotiation," in *IEEE/WIC/ACM International Conference on Web Intelligence (WI)*, Oct 2016.

[42] D. Talia, "Clouds meet agents: Toward intelligent cloud services," *IEEE Internet Computing*, vol. 16, no. 2, pp. 78–81, March 2012.

[43] E. Kafetzakis, H. Koumaras, M. A. Kourtis, and V. Koumaras, "Qoe4cloud: A qoe-driven multidimensional framework for cloud environments," in *2012 International Conference on Telecommunications and Multimedia (TEMU)*, July 2012, pp. 77–82.

[44] I. Apostol, R. Lo Cigno, P. Antoniadis, L. Maccari, L. Navarro, M. Karaliopoulos, M. Dulong de Rosnay, and M. Michalis, "Dissemination Report: Summary of Dissemination Actions and Adoption of netCommons Solutions During the Third Year," netCommons Deliverable D6.3, Jan. 2019. https://netcommons.eu/?q=content/dissemination-report-summary-dissemination-actions-and-adoption-netcommons-solutions-durin-1

[45] L. Maccari and C. Pisa, *Building community networks in Italy: Hacker-led experiments to bridge the digital divide*, 2018. https://giswatch.org/en/country-report/infrastructure/italy

[46] L. Navarro, R. Baig, F. Freitag, E. Dimogerontakis, F. Treguer, M. Dulong de Rosnay, L. Maccari, P. Micholia, and P. Antoniadis, "Report on the Existing CNs and their Organization (v2)," netCommons Deliverable D1.2, Sept. 2016. http://netcommons.eu/?q=content/report-existing-cns-and-their-organization-v2

[47] V. Aubree, M. Giovanella, Federica and Dulong de Rosnay, A. Messaud, and F. Tréguer, "European Legal Framework for CNs (v3)," netCommons Deliverable D4.3, version 1.0, Aug. 2018. https://www.netcommons.eu/?q=content/european-legal-framework-cns-v3

[48] L. Maccari, G. Gabriele, R. Lo Cigno, M. Karaliopoulos, and L. Navarro, "Assistive growth: Towards scalable community networks topologies," *under review in the Elsevier Ad Hoc Networks, invited paper*.

[49] P. Antoniadis, I. Apostol, P. Micholia, G. Klissiaris, V. Chryssos, and M. Karaliopoulos, "Multi-Disciplinary Methodology for Applications Design for CNs, including Design Guidelines and Adoption Facilitation (v1)," netCommons Deliverable D3.1, Jan. 2017. http://netcommons.eu/?q=content/multi-disciplinary-methodology-applications-design-cns-including-design-guidelines-and

[50] A. Pilichos, M. Karaliopoulos, and I. Koutsopoulos, "From Community Networks to Community Data: The AppLea Farming Mobile App," in *IEEE European Conference on Networks and Communications (EuCNC)*, June 18–21 2018.

[51] L. Belli, B. d. S. Ramos, P. Antoniadis, V. Aubrée, R. Baig Viñas, A. Dadoukis, P. Dini, M. Dulong de Rosnay, N. Echániz, K. Heimerl, M. Johnson, P. Kosakanchit, F. López Pezé, S. Mansour, S. Maglavera, J. Martignoni, J. Mavridis, S. Meinrath, L. Navarro, H. Niavis, R. Roca i Tió, S. Sevilla, and F. Tréguer, *The community network manual: how to build the Internet yourself.* FGV Direito Rio, Nov 2018. http://bibliotecadigital.fgv.br/dspace/handle/10438/25696

A. Participatory Design Methodology Booklet

The Booklet reported here is meant to be printed and used as a user friendly guide for interdisciplinary and transdisciplinary groups wishing to engage in participatory design processes for local application in CNs.

As already mentioned the format is not fully compatible, for obvious reasons, with this deliverable, so readability is impaired, and extraction for use impractical. For download, we refer interested readers and users to the page of Deliverable 3.3 where new versions of the methodology booklet will be uploaded as they become available. At the time of writing of this deliverable, the current version is version 0.6, also for the . Further developments and co-creation of future versions of this methodology will take place at the NetHood studio.

COMMUNITY SERVERS: BRINGING COMMUNITY NETWORKS TO THE GROUND

DRAFT

A methodology under construction for the participatory design of local applications in Community Networks

INTRO

Concept & Text:
Panayotis Antoniadis (NetHood)

Graphic design & Layout:
Luisa Lapacciana

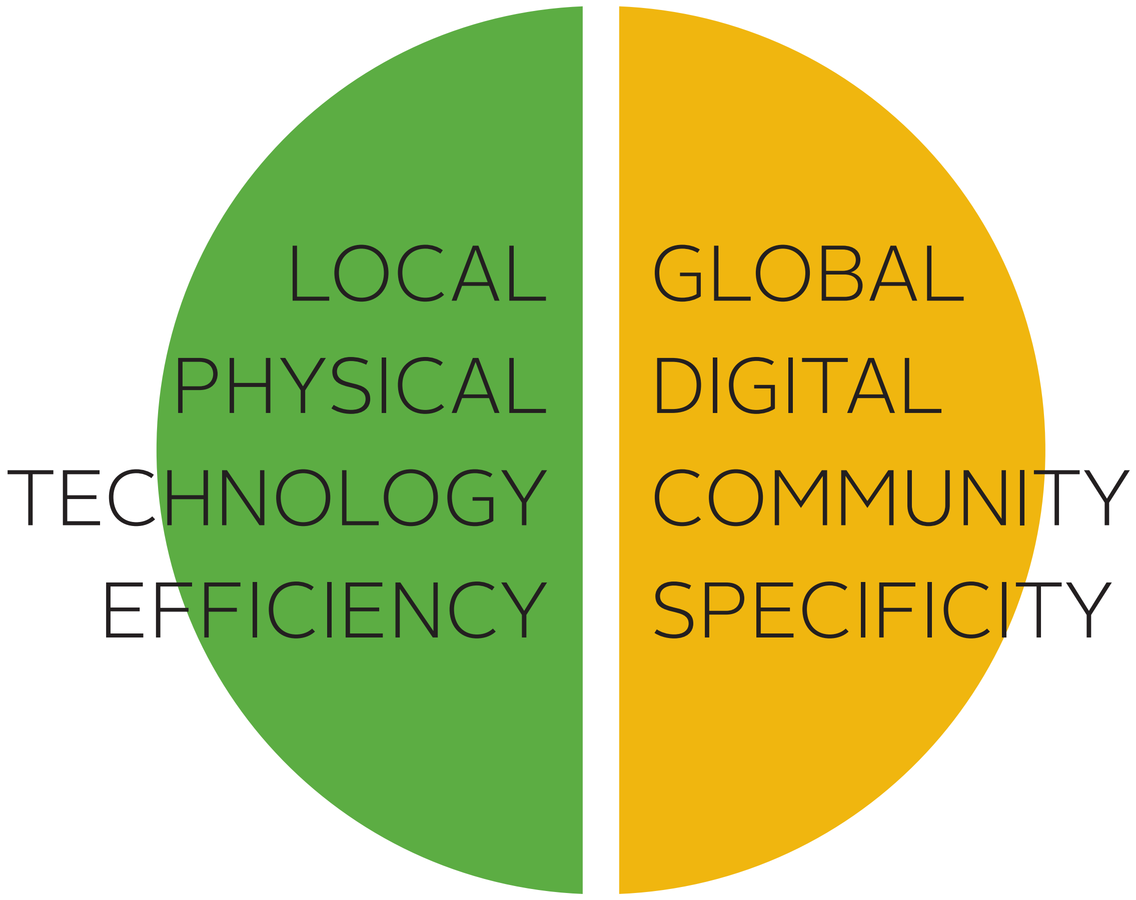
Credits: MethodKit

The methodology is an on-going work designed to be co-created over time.

The first version of the methodology, published as a netCommons report, together with detailed accounts of the different actions. It was based on the experience of a participatory design process for the Sarantaporo.gr Community Network, led by NetHood, Panayotis Antoniadis, Ileana Apostol, and Alexandros Papageorgiou.

This booklet's version of the methodology has further improved through feedback by netCommons partners Merkouris Karaliopoulos, Aris Pilihos, Leonardo Maccari, Felix Freitag, and George Klissiaris.

Future versions are open to feedback and contributions in this wiki: http://nethood.org/studio/


LOCAL
PHYSICAL
TECHNOLOGY
EFFICIENCY
GLOBAL
DIGITAL
COMMUNITY
SPECIFICITY

“Local applications for CNs is a myth”

JUERGEN NEUMANN,
OFF-THE-CLOUD ZONE
@TRANSMEDIALE 2016

The idea that Community Networks can host a wide variety of local applications
has been always part of the vision of an Internet built by the people for the people.

In reality, when Internet access is available, local services tend to atrophite.

There are many reasons for this:

- Self-hosted software applications cannot easily compete in terms of usability with their commercial counterparts.

- Servers require maintenance and applications careful design, and both require significant human resources, often not available locally.

- In a fully connected world locality is losing its importance and the notion of a community itself is becoming more and more blurry.

“A society of simple tools that allows men to achieve purposes with energy fully under their own control is now difficult to imagine”

IVAN ILLICH,
TOOLS FOR
CONVIVIALITY

Despite the challenges, developing successful local applications for Community Networks, designed as tools for conviviality, is important for many reasons: social, economic, political, ecological, practical.

It is critical to find the right balance between the global and the local. Between technology and society. Between the digital and the physical. Between growth and limits. Between global Internet corporations and local Community Servers.

This methodology aims to encourage community networkers, social scientists, designers, software developers, and urbanists to work together and imagine from scratch the concept of a community server, hosting local applications that serve local needs.

The target audience of this booklet is experts on all those fields, already using their own methodologies and practices, but which recognize the need to find a common understanding and language toward the vision of a Community Server.

Keep reading if you are ready to step out of your comfort zone and invest some extra time for imagining a more organic Internet ...

TERMINOLOGY, DEFINITIONS, AND ASSUMPTIONS

A Community Server in a more or less powerful computer, a server, hosted inside an existing Community Network, in one of its network nodes.

A Local Application denotes the software, the digital platform, that runs on a Community Server, and which is accessible in principle or by design only to people residing in a specific geographic location, covered by the Community Network.

Hybrid Space is the complex space created by the combination of the physical space, the geographic area where a Local Application is accessible, and the digital space, the digital interactions enabled by the application.

The current draft of this methodology assumes a running CN on top of which a set of motivated actors wish to design and host a local application. Future versions will include elements related to the creation of the CN itself, in parallel or beforehand.

THE PHYSICAL SPACE

The target region could be as small as a square and as big as a whole city or region. An urban garden, a neighbourhood, a valley, a pedestrian street, a square, a small island.

Any physical space that wishes to build a collective identity and facilitate social interactions for which a locally owned and designed digital space can play a key role.

THE DIGITAL SPACE

The design of an application could range from the customization of an existing self-hosted application to suit a specific scenario to a completely new design and implementation.

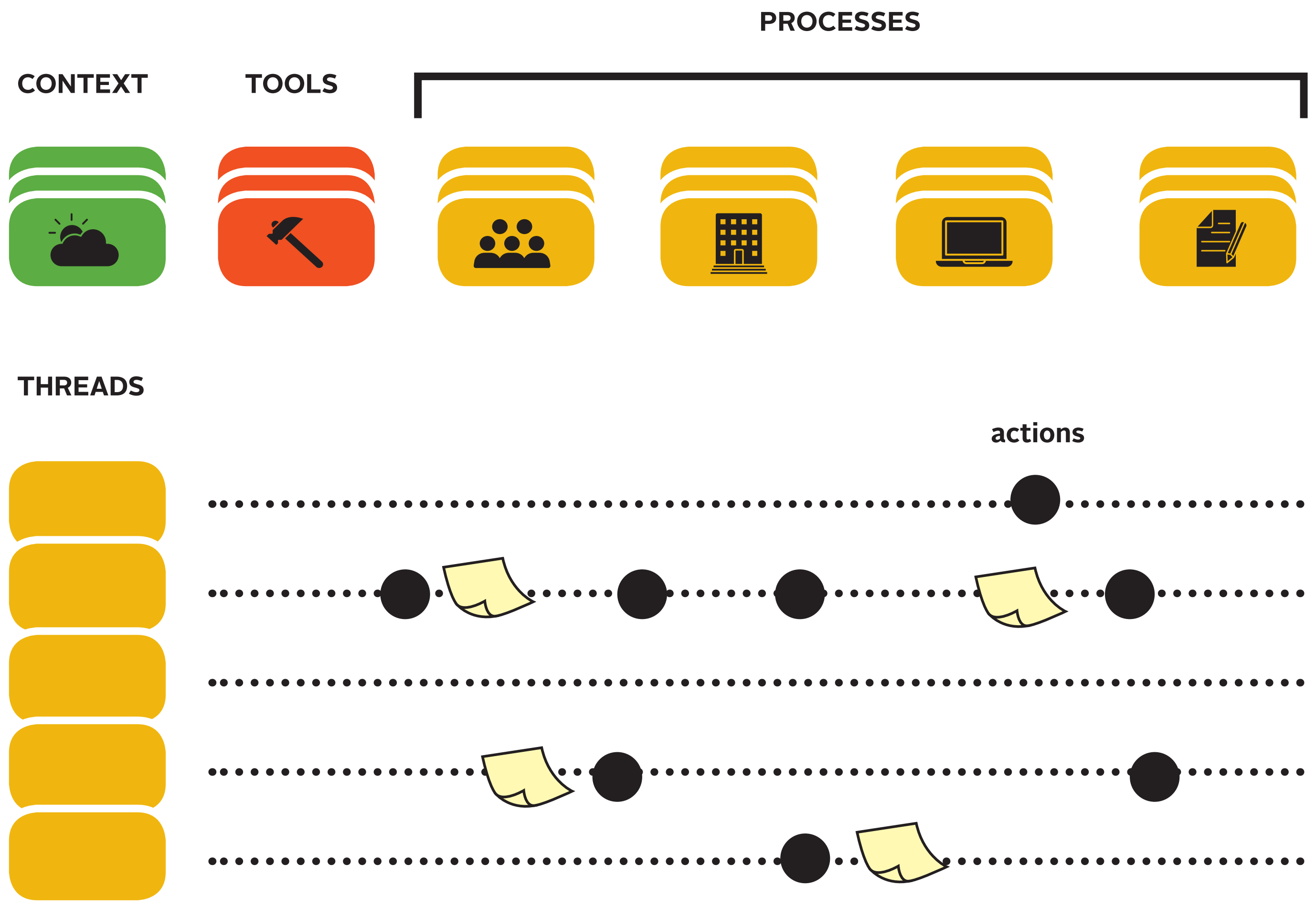
Examples of free software that could be used to support digital interactions tied to specific locations include Wordpress, NextCloud, Etherpad, and more.

THE HYBRID SPACE

In Berlin's Prinzessinnengarten the Neighbourhood Academy in collaboration with UdK built a custom application for nicely presenting and archiving interviews of visitors, making available their collective knowledge literally in the garden through a local WiFi network.

Barcelona's initiative on data sovereignty ...

INTRO



OVERVIEW

The presented methodology is very ambitious in trying to capture the whole complexity of a long-term collaborative project between different actors both internal and external to a community, facilitating the communication across different expertises, disciplines and perspectives.

But it also allows for an incremental adoption, starting with a really simple “MethodKit” version of it, organized in different “parts”: Context, Threads of action, and a few important methodological elements

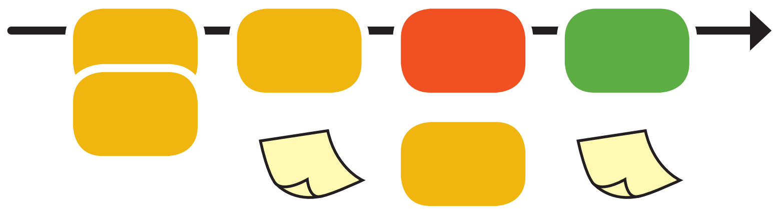
In addition, we propose a very specific, and novel, methodology on how to use those methodkit cards in a long-term iterative process inspired by agile methodologies and jazz improvisation.

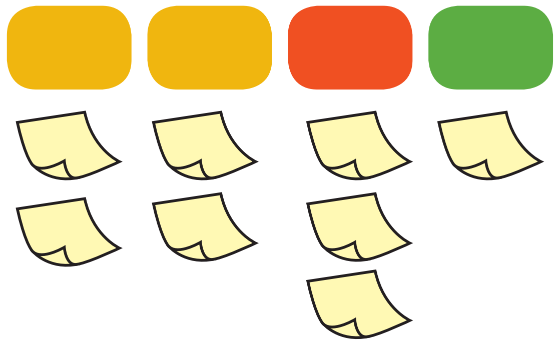
This methodology includes a list of suggested actions or options for every card, together with guidelines and evaluation metrics. In other words, we provide detailed examples of how a methodkit session could evolve, what could be the outcome by considering the different important “things to think of, and act upon”.

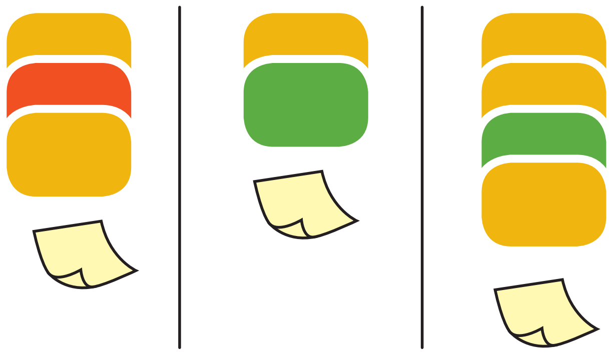
EXAMPLES OF USING THE METHODKIT CARDS

The MethodKit offers already a wide variety of ways to use the methodkit cards for brainstorming and collaboration, and also various "canvases".

https://methodkit.com/how-to-use/







METHODKIT CARDS

CONTEXT
The context cards describe important variables that determine the special characteristics of a community the need to be taken into account at every stage of the participatory design process.

COMMUNITY
Designing for real community needs is a complex process that requires more than inviting a few people in a room to give feedback on specific design choices.

DIGITAL SPACE
The design and implementation of the local application is the core activity of the project but the challenge is understand the equal importance of all the other processes.

TOOLS
During the implementation of the project, there are various existing tools and methodologies for certain threads of action that can be of particular use.

PHYSICAL SPACE
Acknowledging the hybridity of space and placing the design of a local application in the actual physical environment is one of the most challenging and novel tasks for building useful and used community servers.

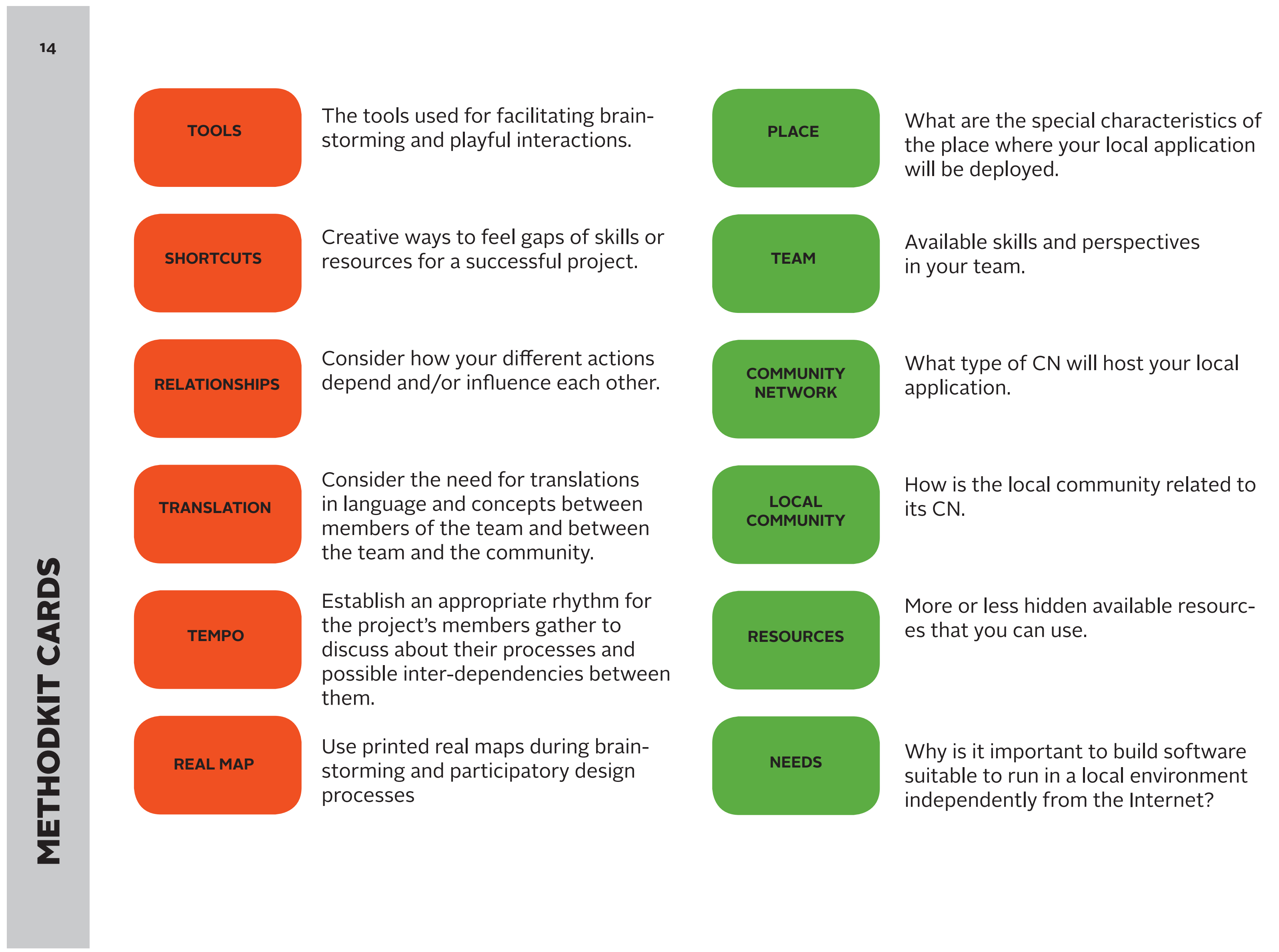
PROJECT
People hesitate to invest effort and time if they are not convinced of the sustainability and overall framing or identity of the project. Make sure to dedicate significant resources and creativity to this respect.

METHODKIT CARDS

LISTENING	How do you listen to the communities characteristics and needs.
COMMUNITY ENGAGEMENT	Events and processes that help you engage the community in the design of the applications.
DOCUMENTATION	Capture and communicate your understanding of community needs and special characteristics.
LEARNING	Have you reserved enough time for training the community in new concepts?
TRUST BUILDING	Show who you are and be engaged, to be trusted.
NETWORK VISUALIZATION	Tangible ways through which your local network, infrastructure and applications are made visible in the ground, like posters, printouts and various artefacts.
DEDICATED SPACES	Permanent or temporal locations where someone can learn about the community network and its local applications, engage in learning and participatory process, and meet in person the people behind the project.
SPACE INFRASTRUCTURE	Internal and external design of the space, tools to facilitate interactions, and artefacts to communicate the selected framing and overall identity of the network.
RUNNING THE SPACE	Governance mechanisms and processes that guarantee the sustainability of the space and their proper functioning.
HYBRID INTERACTIONS	Links between the digital and the physical through displays and other visualizations of online interactions combined with face2face gatherings.

METHODKIT CARDS

APPLICATION	Continuous design for user experience, needs, and appropriation in the core functionality offered by your local application.
SOFTWARE DEVELOPMENT	A realistic plan for the deployment of different versions of the software application, including a Minimum Viable Product
ADMINISTRATION	Making things easy and flexible for the administrator is the key ingredient of a local application for CNs
CURATION	Online spaces, like physical ones, need presence and curation. No one will use your application if you are not “there”.
CONTINUOUS FEEDBACK	The users of the software should be encouraged and facilitated to send you feedback on issues and feature requests, which can both help you improve the functionality but also reveal their needs and priorities.
PROJECT IDENTITY	Define your project’s identity in its full complexity, and keep it updated.
COMMUNICATION	Share your project’s objectives and results.
ORGANIZATION	Make sure the right people are working on the right tasks
NETWORKING	Build relationships with local actors but also external communities and international networks.
FUNDING	Collaborate with the community to find complementary funding for your project.



14

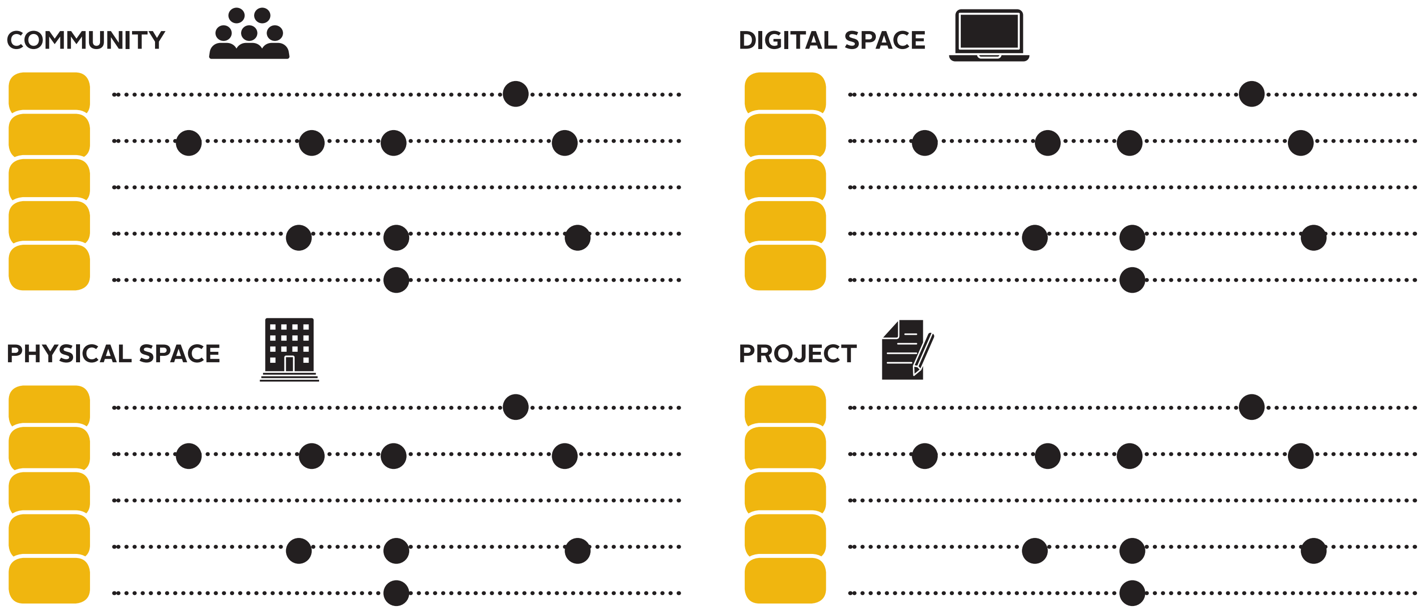
METHODKIT CARDS

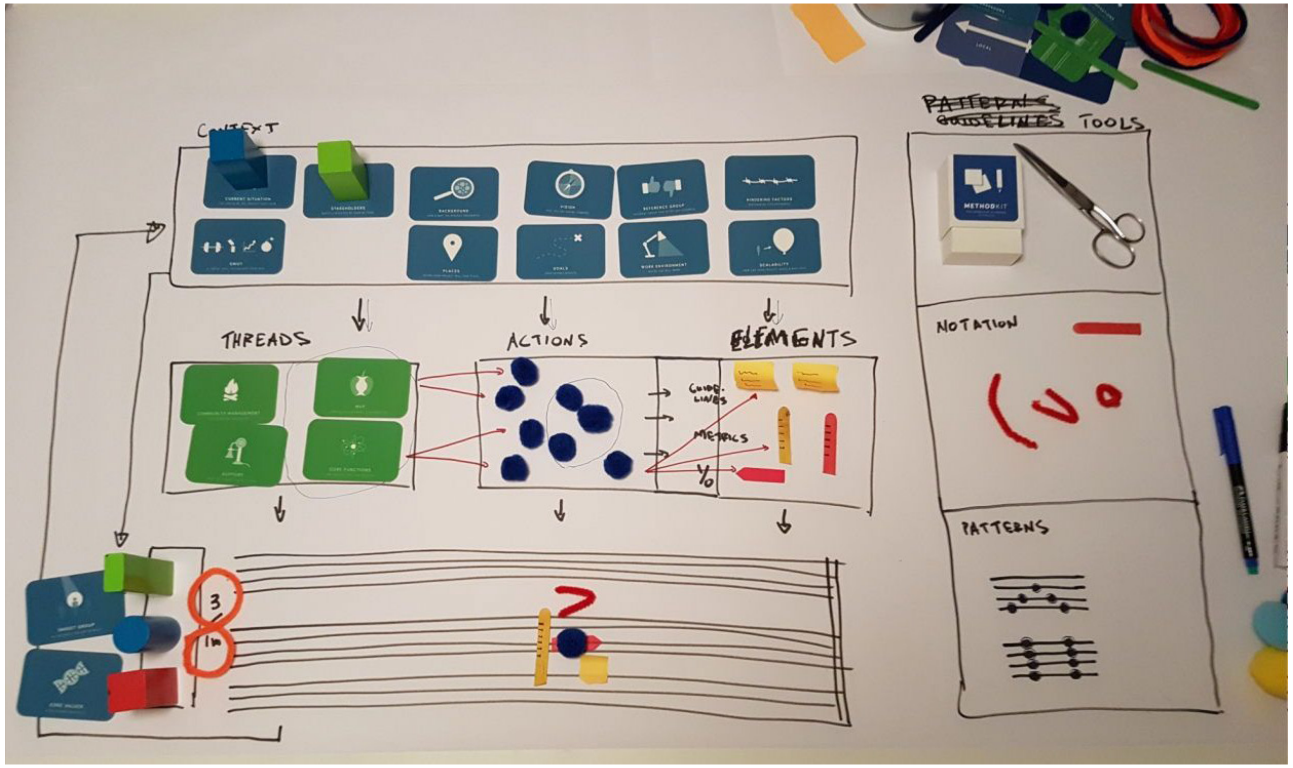
TOOLS	The tools used for facilitating brain-storming and playful interactions.
SHORTCUTS	Creative ways to feel gaps of skills or resources for a successful project.
RELATIONSHIPS	Consider how your different actions depend and/or influence each other.
TRANSLATION	Consider the need for translations in language and concepts between members of the team and between the team and the community.
TEMPO	Establish an appropriate rhythm for the project's members gather to discuss about their processes and possible inter-dependencies between them.
REAL MAP	Use printed real maps during brain-storming and participatory design processes
PLACE	What are the special characteristics of the place where your local application will be deployed.
TEAM	Available skills and perspectives in your team.
COMMUNITY NETWORK	What type of CN will host your local application.
LOCAL COMMUNITY	How is the local community related to its CN.
RESOURCES	More or less hidden available resourc-es that you can use.
NEEDS	Why is it important to build software suitable to run in a local environment independently from the Internet?

PROJECT SCORE

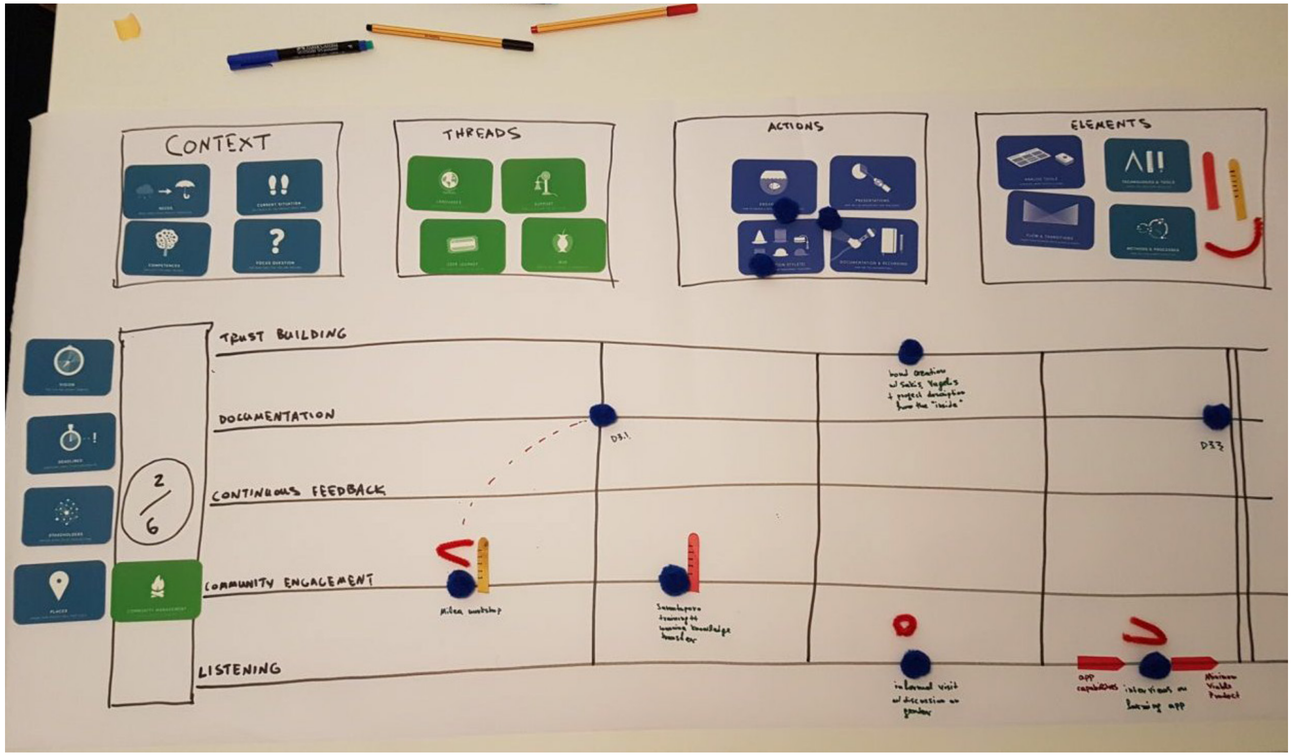
Given the particular focus of this methodology we propose a novel way to collaborate with the customized set of methodkit cards presented above, using the so-called Project Score as a triangulation and self-reflection device between the groups that lead the different processes. The objective of the Project Score is to visualize in a playful and inspiring way the different actions that the different involved actors in a participatory design process have implemented or plan to do so.

The presentation resembles intentionally a music score under construction, which unlike classical music is not predefined, but like in Jazz contains just a few guiding elements, subject to improvisation according to the dynamics of the group and the reactions of the audience, the community.





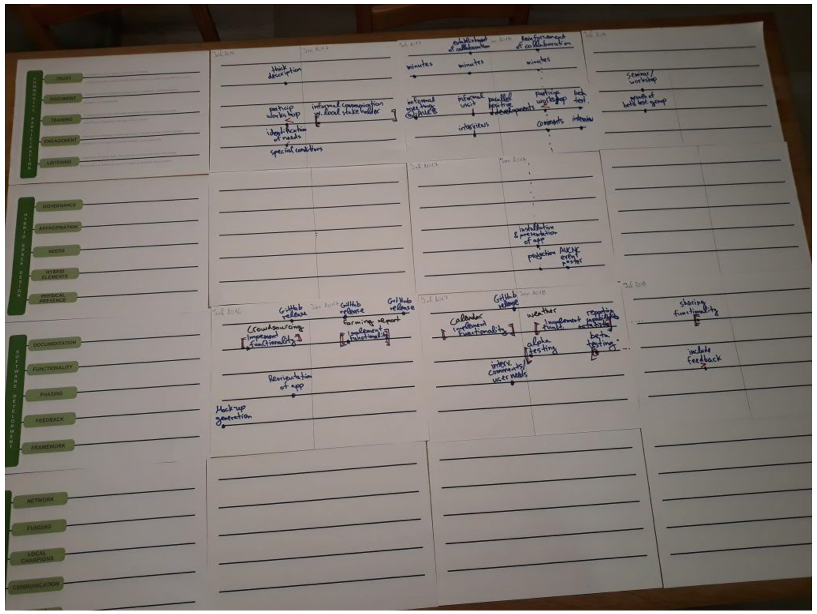
An early depiction of the overall methodology using existing methodkit cards compatible with those proposed in this methodology

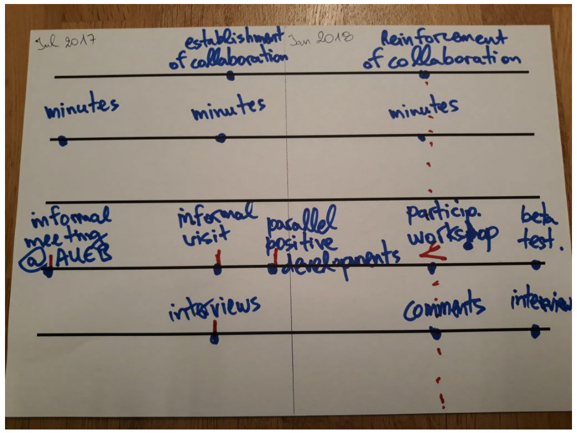

CONTEXT
THREADS
ACTIONS
ELEMENTS
TRUST BUILDING
DOCUMENTATION
CONTINUOUS FEEDBACK
COMMUNITY ENGAGEMENT
LISTENING
2/6
D3.1
D3.3
Milan workshop
Sarantaporo training to ensure knowledge transfer
brand creation w/ Sakis, Vagelis + project description from the "inside"
informal visit w/ discussion on gender
app capabilities
interviews on learning app
Minimum Viable Product

A first representation of the Community process in the Sarantaporo case study

A training workshop at Pithio village in the Sarantaporo area, using for the first time a real map and toys for representing the different nodes of the network and their properties, as suggested by the "planning for real" methodology (see Tools).

In later training workshops in the Saranta-poro area, the printed real maps of the area became a standard tool for planning the deployment of new nodes in the different villages.





A full project score during a gathering of the community engagement and software development teams of the netCommons project

Images from a participatory design workshop for netCommons Applea farming app

Bringing international experts in the Sarantaporo village and connecting remotely with a similar Community Network in a very different environment (New York city) had a great impact in showing to the local community how important is what they are doing.

Public presentations of the the Sarantaporo case study at the Onassis foundation (left) and the EU parliament (right). Presenting the project in international and local forum attracts attention and provides opportunities for synergies and funding.

Informal discussions during visits to the community without a particular reason can prove critical for building trust and discovering the real needs not often expressed in formalized settings.

PLACE

This context variable refers to the given environment where the project takes place: the geographic location, the demographics of the population, some high-level social, political, economic, and cultural characteristics.

It is out of the scope of this booklet to provide a detailed analysis on how these variables could/should influence the implementation of the methodology, and this is why they are all put together under the same context variable.

But it is important to identify the corresponding values and keep them always in mind in order to reflect on how they influence the different decisions at different levels. This is critical in the beginning of the project but also during its evolution and especially when important "discoveries"' about the nature of the place are made that can help to improve the common understanding of the team of this very important contextual element.

RESOURCES

Especially for low budget projects, one should carefully identify the available resources which will determine the priorities and feasibility of the different steps. In short, the most important resources on the side of the team are the available BUDGET and TIME, and already available SOFTWARE and INFRASTRUCTURE.

On the side of the community, there might be many visible (and non-visible) resources, like available OPEN SPACES for gatherings or training sessions, old unused devices that could be recycled, and so on.

SKILLS

The first important question one needs to ask before determining the right strategy for a participatory design process is related to the available SKILLS and resources of the leading team and the potential external partners.

As a basis, there should exist in the team, on the one hand, an Application-Designer and Software-Developer, who will implement the actual application according to the local needs,
and on the other hand, a Community-Organizer and Event-Facilitator responsible to engage and interact with the local community and try to identify matches between the local needs and the functionality potentially offered by the application.

The setting up of such a team can lead already to a quite costly process in terms of human resources and overall expenses. But there are still many key skills that should be ideally covered by specialized people such as community outreach and communication, education, documentation, funding, and more. Community-Networking-experts would be also needed on the technical side if they are not already part of the target community.

The most challenging aspect is the “cultural” differences between the two types of expertise that need to be combined, especially in light of the non-obvious reasons why local applications are actually needed, especially from the perspective of a non particularly technical and/or political person.

In case where there is only one side represented in the actual team it would be important that someone from the team takes the “missing” role, e.g., an engineer playing the role of the facilitator, or a community organizer playing the role of the software developer.

COMMUNITY NETWORK

One could identify three radically different case studies:

Affordable-Internet: This category includes CNs which are built to provide affordable or even free Internet access to small or large communities. For example, there are numerous rural or small-scale urban community networks built by experts with varying levels of engagement of the local community, with the clear goal to provide affordable Internet access. Typical examples that fit this category could range from the Sarantaporo.gr network serving more than 10 villages all the way to the OTI initiatives in the US in Detroit and NY, among others. Large parts of the Guifi.net network also fit this category and the same for Freifunk.net in Germany and FunkFeuer in Austria, as well as the many community ISPs that form the \ac{FFDN} in France.

Alternative-Internet: This category includes CNs built as big ``sovereign'' networks that do not depend on the Internet to provide useful services at a smaller scale. Typically, these are city-wide or even region-wide community networks, whose members are mostly technically savvy and key requirement for participation in the community is the installation of a node. Some projects are built exactly around this idea, like AWMN and ninux.org. Other projects, like Freifunk and Guifi.net, while focusing on Internet connectivity have some of their core members actively building local (sometimes local-only) services along these lines.

Outside-the-Internet: This category includes typically small-scale CNs or offline networks, built to provide local services in a specific location, often through a single node like the PirateBox or the MAZI toolkit.

In all these three scenarios the participatory design of local applications makes sense but possibly for different reasons and most importantly it is a different ``community'' that needs to be considered.

LOCAL ACTORS

For each of the two main types of CNs described above, there are two different options for the corresponding type of engagement of the local community.

In the first CN type, Affordable-Internet, the community building and maintaining the network is typically much smaller than the community using the network, for Internet access. In the most participative scenario, the first community is fully “contained” in the second one, and it is actually members of the ``social'' community, a village, a neighbourhood, a wider urban area, that have built the CN to serve the needs of the whole community, an Independent-CN-for-Affordable-Internet.

On the other extreme, there are the cases that the main actors that built the network come somehow from the outside and it is only a handful of local actors that help to maintain it with the continuous support by the external experts, a Supported-CN-for-Affordable-Internet.

For the second CN type, Alternative-Internet, the social community typically overlaps with the network community. The candidate applications are to be used primarily by the ``node owners'' of the CN, those actively engaged in the construction of the network itself, an Independent-CN-for-Alternative-Internet. Ninux.org is a typical example of this category, while AWMN is another one, very proud for the wide range of local services replacing all major Internet services, developed by its members. But there are also cases that Alternative-Internet CNs are meant to serve the wider community as was the case of RedHook WiFi, a Supported-CN-for-Alternative-Internet.

Finally, there is a third category in which the CN does not exist already but is only ``potential'' and the creation of the CN (together with its local applications) is part of the objectives of the overall process.

NEEDS

Before entering in the analysis of the needs of the community one must tackle the single question that very often rises before, during, and after the design and implementation of a local application: “Why local?” . Why it is not enough to connect to the Internet and use the generic application (cloud-based or not) that everyone who has Internet access uses every day?

This is a list of possible reasons, starting from the more practical reasons toward the more political ones.
NO-INTERNET-ACCESS:
in cases where Internet access is simply not available or very limited, local applications can actually enable a wide range of basic digital interactions not possible otherwise. This is perhaps the most obvious scenario in which local applications make sense.

RESILIENCY:
local applications could be seen as an alternative to the Internet-based services when the latter fail for var-

ious reasons (a physical disaster, an economic or political crisis, among others), increasing the resiliency of the system and the community.

HIGH-PERFORMANCE:
for a certain range of applications, local servers could help to achieve better performance, which is especially the case when Internet connectivity is limited of low quality (e.g., highly asymmetric).

NET-NEUTRALITY:
the access to local applications in a CN can enjoy the net neutrality principle of fair treatment leading to better performance, support of local actors, and also openness to innovation.

PHYSICAL-PROXIMITY:
local applications running on a CN can have useful information about the physical location of its users without the use of any private information such as GPS coordinates or IP addresses.

DIGITAL-SKILLS:
hosting local services and applications, exposes the local community to the challenges of running Internet platforms and complex issues like privacy, freedom of expression, and more, providing the means for digital emancipation and education on digital skills.

COMMUNITY-EMPOWERMENT:
the engagement of the community not only in the creation of a community network but also in the design of a local application can contribute to feelings of empowerment and in general increase the community spirit and social cohesion.

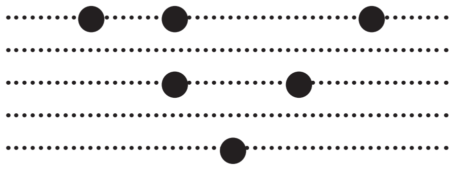
DATA-OWNERSHIP:
by construction, the data generated and stored through a local application are owned by members of the community. This ownership could/should lead to the appropriate governance structures for the management of this data for which there is the unique option, compared to Internet-based platforms, to be democratic.

SELF-DETERMINATION:
the power over the design of a local application, is a more subtle than "ownership", but very critical power potentially offered to a local community, which could be also democratically shared among all of its members.

PRIVACY:
derived from the data ownership and self-determination reasons, local applications could be seen as a means to build services that collect and manage information according to the needs of the local community and could lead to systems that are more respectful to privacy and freedom of expression, without providing an a-priori guarantee for this.

COMMUNITY PROCESS

Designing for real community needs is a complex process that requires more than inviting a few people in a room to give feedback on specific design choices.



LISTENING

RANDOM WALK
INFORMAL DISCUSSIONS
Walk around and observe, focusing on the numerous details of everyday life, and engage in informal discussions. Needs are not always conscious and not always expressed in public, but they express themselves in the most unexpected moments. So, consider to Stay More and Let Yourself Be Surprised when visiting the community.

EXPLORE-LOCAL-MEDIA
Nowadays a lot of a community's character is expressed through online interactions in forums, social media, news outlets. Exploring these interactions, such as discussions, photos, and videos, through appropriate hashtags and groups can give invaluable information. This can be explored also from a distance and complement the more costly in-person visits.

PERSONAL RECORDINGS
Recording the everyday life of a community through short audio interviews, photos and videos can be a very informative process that operates in multiple dimensions. Observing these recordings, and revisiting them from time to time, reveals different hidden layers of information on a community's character but also the changing perspective of the observer.

COMMUNITY ENGAGEMENT

PARTICIPATORY WORKSHOP
Participatory workshops are the most explicit form of participatory design and must be used with caution. While applying different methodologies, adaptability and improvisation, and honesty and transparency are the two most important qualities that you need to develop. Note that it is often that informal meetings are more productive than official workshops.

ESTABLISH SMALL BETA-TESTER GROUP
Select a few motivated people from the community to work closer with and engage them in testing your application since the early stages, and share with them regular updates based on their feedback.

ONLINE GROUP COMMUNICATION
E-mail lists and messaging groups can play a key role in building a community spirit and provide quick support and receive feedback in different phases of the project.

DOCUMENTATION

SHORT SUMMARY
Short summaries of events and informal meetings are fundamental tools for the internal communication and coordination between the project's teams. They should be written in a way to highlight the most important findings, "user stories", and non-obvious observations. It is highly recommended that some of these summaries are made available to the community through an online collaborative space or even in a public web site or social media.

DETAILED MINUTES
When possible short summaries could be complemented by detailed minutes with optional comments, which should be made available to the participants to provide feedback and annotations. Sometimes the annotations provide even more useful information than what has been said during the meetings.

THICK DESCRIPTION
Detailed accounts of visits and informal meetings can reveal important details that often do not make it to short summaries and minutes. Such details might seem unimportant at a first glance but they often contain a lot of subtle information that can make a big difference in the long-run.

LEARNING

TRAINING SEMINARS
Meaningful participation requires deep understanding of the object of design and its potential role in community's life. The use of real maps and toys can make a big difference while describing technical aspects both on the digital and physical space. Training local people to become themselves trainers is both empowering and effective.

PRODUCE EDUCATIONAL MATERIAL
Learning processes need to be supported by adequate educational material. Ideally, this material should be translated in the local language and adapted to the local needs.

ESTABLISH AN EDUCATIONAL PROGRAMME
Independently or in collaboration with existing institutions or educational centres, it would be very helpful if education becomes a separate complementary project.

TRUST BUILDING

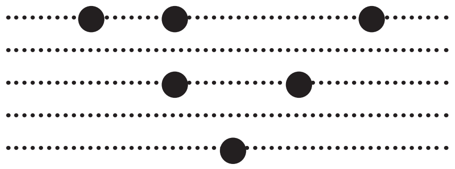
ORGANIZE A PUBLIC EVENT
Public events on the overall project, including demos and invited guests, can add to the credibility and transparency of the project. They can also become instrumental in identifying key local actors and generating a feel of trust regarding the intentions and integrity of the project leaders.

PARTICIPATE IN LOCAL PROJECT
The most typical reason for failure of an "external" participatory design project is the perception that the project leaders want to "do their project and leave", caring only to push their technology or receive their funding. Participating in local projects and activities and link them with one's own project can reverse this stereotype and build trust, but only if it is genuine. Caring for a community cannot be faked. Better not start the project otherwise!

ENGAGE IN SOCIAL INTERACTIONS
Becoming part of the community through social interactions, participation in local events and rituals, sharing thoughts and personal stories are the best ways for the community to get to know you and trust you. But again, don't pretend to care (if you don't)!

PHYSICAL SPACE PROCESS

Acknowledging the hybridity of space and placing the design of a local application in the actual physical environment is one of the most challenging and novel tasks for building useful and used community servers.



NETWORK VISUALIZATION

VISUALIZE ACCESS
Being geographically constrained, your local network needs to communicate its coordinates and access methods (e.g., WiFi SSID, URL, etc), but also its special characteristics compared to traditional Internet services (e.g., being a local-only network), in public space.

VISUALIZE TOPOLOGY
It can be very helpful and inspiring if your local network is represented with physical objects on a real map or maquette, which ideally could be always accessible in various dedicated spaces, allowing people to understand and deliberate on its design and coverage, and create feelings of ownership and pride.

VISUALIZE INFRASTRUCTURE
You can make the infrastructure visible through posters, signs or objects designed according to the selected visual identity of the network, designating the presence a network node (e.g., an antenna) or a server where they actually are.

DEDICATED SPACES

RUN A KIOSK
Places, like a kiosk in a square or event or a desk in a library, where there is information and educational material about the network. Member of the networks could contribute being present to address specific questions and give customized advice on the potential involvement of someone based on their needs and skills.

PARTICIPATE IN A COMMUNITY SPACE
Info points and meetings of a Community Network could be hosted in welcoming community spaces, creating opportunities for interactions and synergies with other like-minded groups.

CREATE A COMMUNITY HUB
A dedicated space for the Community Network can provide more visibility and opportunity for the development of long-term learning, governance, and community participation processes through events, assemblies, seminars and courses. But it needs more funding and human resources!

SPACE INFRASTRUCTURE

INTERIOR DESIGN
The way a space is designed can make a difference in terms of engagement, e.g., the placement of chairs toward one-to-many talks vs. many-to-many gatherings. Other elements of interior design can make it easier for people to "step-in" or facilitate hybrid interactions.

SURFACES
Surfaces for projections, announcements, timetables, displays, message boards, or brainstorming boards are very important and should be carefully selected and placed in the space. Gathering with a few key people to discuss informally about your concerns and plans of action on top of a real map of the area might lead to very valuable feedback.

SERVER ROOM
Even if not technically necessary, having a community server installed in the location which hosts also physical interactions can be convenient and empowering.

RUNNING THE SPACE

ORGANIZE EVENT
Organize a wide variety of events ranging from participatory design workshops and training seminars to informal meetings and gatherings around the network and the community. Keeping a digital memory of them in the local network can be a good starting point for motivating the use of local applications.

OPEN DOORS
Spaces work better when they are open in regular times announced in advance. It will help a lot of cause if there is always someone during the selected opening hours to inform the public about your Community Network and ideally organize interesting activities of interest.

ORGANIZE ASSEMBLY
Make sure that there are regular meetings around the governance of the space and the establishment of appropriate rules that will guarantee the proper functioning and avoid misunderstandings.

HYBRID INTERACTIONS

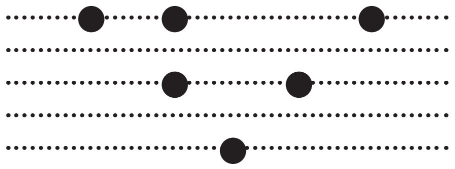
HYBRID HAPPY HOURS
Establish a certain appropriate time of the day where people come to interact through the local application in proximity, perhaps using a big display to visualize their interactions, but at some point stop and take away the devices to talk face to face.

PERMANENT INTERACTIVE DISPLAY
Project on a visible display an interactive element of your local network (e.g., a chat room, an etherpad page or an interactive poll). Note that you might need to be present or check regularly to moderate the content contributed.

SPACE ENCOUNTERS
Consider organizing digital "encounters" with other relevant spaces using a large projection screen and appropriate equipment that can allow people to communicate seamlessly with the other side as a group, forming a "hybrid" roundtable.

DIGITAL SPACE PROCESS

The design and implementation of the local application is the core activity of the project but the challenge is understand the equal importance of all the other processes.



APPLICATION DESIGN

DEFINE VISUAL IDENTITY
Design the logo and decide on important visual elements, color coding, fonts, representation of servers and wireless access points, and more. The visual identity is a really critical component of your application and it is worth to spend time engaging the community in the process. But note that at some point someone has to make a decision!

DEVELOP USER STORY
Describe in detail how your application will be used over time by a specific “target user”, based on input received from the Community Process. For every step, write down the information requested and delivered and the corresponding interface actions needed for the desired outcome to be reached.

CREATE MOCK-UP
Translate a specific user story to specific functionality offered by your application, visualized through a series of screens that the user will be exposed to during this process. Take your time exploring different options before actually implementing the user interface.

SOFTWARE DEVELOPMENT

CHOOSE DEVELOPMENT FRAMEWORK
This is one of the first and important actions for the software development thread. The selected framework will determine the possibilities for integration with other software solutions and the culture of developers who will be engaged over time, among others.

DEFINE MINIMUM VIABLE PRODUCT
Decide on a minimal but functional version of your application and establish the whole lifecycle of the development process based on it.

INTEGRATE EXISTING SOLUTION
You should try to minimize the “new” software developed during the participatory design process and depend on core functionality on existing free software solutions like NextCloud, Etherpad, and more. These are improving day by day and it is important to keep an eye on the developments in this scene.

ADMINISTRATION

SELF-HOSTING PROCESS
For local applications to be easily adopted in different contexts they need to be easily “self-hosted” by someone with limited technical skills. This will make it also easier to engage more developers in the implementation of the software.

CUSTOMIZATION OPTION
Make sure that you offer reasonable and adequate options for customization in terms of appearance, visualization, permissions, and more, paying attention to the increased complexity. Ideally, a new customization option could correspond to a specific need and should be made available to a specific person responsible for setting this option.

FEDERATION API
Allowing local instances of your application to communicate between them or with online “aggregation” servers can offer the option to balance the local with the global according to the needs of the community.

CURATION

INITIALIZE ONLINE INTERACTIONS
Make sure to start using the digital platform you develop among the most engaged users but also the developers and create a welcoming atmosphere for those connecting for the first time.

INSERT MOTIVATIONAL MESSAGES
Make sure that the users of your application feel rewarded when they perform important actions and get informed about the overall activity, but do not overdo it.

MODERATE CONTENT
The more content is inserted in the platform the more it will become important to do some sort of moderation and filtering. Making transparent the reasoning behind your moderation decision can increase the level of trust and engagement of your community.

CONTINUOUS FEEDBACK

GIT* ISSUES MANAGEMENT
Online git platforms like github and gitlab contain a very useful feature (the “issues”) which is managed appropriately it could serve for a feedback platform from all types of users (more or less technical).

PERFORM USER TESTING
Engaging a few motivated community members (the alpha-testers group) to use your online platform without any assistance while you are watching can reveal many imperfections in the design of the user experience of your applications. Consider asking your test users to create new issues on your selected git platform documenting these imperfections and keep a close communication with them through a telegram group or similar.

MONITOR USAGE
Consider implementing ways to gather implicit and/or implicit feedback through the actual use of the interface. Although more difficult technically understanding how the platform is used in practice can be very helpful.

PROJECT PROCESS

People hesitate to invest effort and time if they are not convinced of the sustainability and overall framing or identity of the project. Make sure to dedicate significant resources and creativity to this respect.

PROJECT IDENTITY

DEVELOP MAIN NARRATIVE
You need to distinguish between the identity of your "product", the local application, and this of the overall project for which the application is only a small part. And then develop carefully the narrative, the storyline, of your project. Who are you and why you are doing it. Note that this might change over time so keep your mind open to adjust it accordingly.

MAINTAIN AN INTERNAL PROJECT DOCUMENT
It will prove very useful if the team could collectively edit and maintain a document where the main storyline of your project is developed, but also the history of important developments, resolutions, self-reflections, etc. A sort of a collective project diary, which could be a wiki, a list of Etherpad documents or another collective editing tool.

SWOT ANALYSIS
A classic tool that is worth to use regularly and observe differences over time.

COMMUNICATION

DIFFERENT STORIES FOR DIFFERENT AUDIENCES
Such a multi-dimensional project requires the collaboration with different actors. Make sure you adjust your storyline according to the different target audiences, the local community, local stakeholders, funders or supporters from around the world.

PRESENCE IN SOCIAL MEDIA
Today it seems obvious that a project needs to be present in popular social media, and keep a regular schedule of posts. But note that this is more work than one can imagine and perhaps it could be a wise decision to choose only one or two most appropriate social media channels, those most relevant for your project and community.

ALTERNATIVE MEDIA CHANNEL
Being a project advocating for the deployment of local applications it makes sense to establish a presence also in non-mainstream media channels using self-hosted software like Mastodon.

ORGANIZATION

SHIFT ROLE OF COORDINATOR
In such a complex project, individual teams need significant freedom and for this coordination becomes critical. It is very important to shift the role of the coordinator over time to give responsibility to multiple members of the project

BALANCE BETWEEN VOLUNTEER AND PAID WORK
Depending on the funding sources some members of the team might be possible to get paid while others not, and the same hold for the local community. Make sure to take "corrective" actions, formally or informally, to establish a culture of fairness and trust inside the project.

IDENTIFY NEED FOR HELP
If certain threads of action are considered important but do not perform as expected, discuss with the team about the possible reasons and seek for help either inside the team or by engaging external actors. Sometimes you can get support with clever win-win collaborations, e.g., with students working on related topics, local actors having complementary objectives, etc.

NETWORKING

FIND THE COMMUNITY CHAMPION OF YOUR PROJECT
If you don't manage to engage a few key local actors that believe in your project and would be willing to invest some effort to support it, it will be very difficult, or even impossible, for it to be adopted by the wider community.

ORGANIZE LOCAL EVENTS WITH EXTERNAL GUESTS
If possible it would be very helpful to make your process a special case of a wider (e.g., international) project and link to activities of other communities. Bringing visitors from these international communities in local events can be very effective in gaining the attention and trust of your local community.

BUILD SYNERGIES WITH COMPLEMENTARY PROJECTS
Local networks share many values with other similar initiatives on housing, food, public spaces, money, energy, and more. Make sure that you are in touch with key people from such initiatives and join forces on different fronts: communication, funding, engagement.

FUNDING

EXPLORE LOCAL AND EXTERNAL FUNDING SOURCES
Produce a comprehensive list of possible funding sources identifying the key objectives of the funder, the time frame, and the resources required from your side to apply. There are very often many neglected and underestimated sources of funding both at a local and global level.

SUPPORT FUNDING EFFORTS OF LOCAL ACTORS
If you have an already funded project, consider to use it as a driver and support structure for other local funding efforts, which will then provide complementary resources and also help to build trust.

ORGANIZE A FUNDRAISING WORKSHOP
Bringing together key actors in a workshop dedicated on this topic (possibly public) can reveal complementarities and common objectives of funders, not clear beforehand.

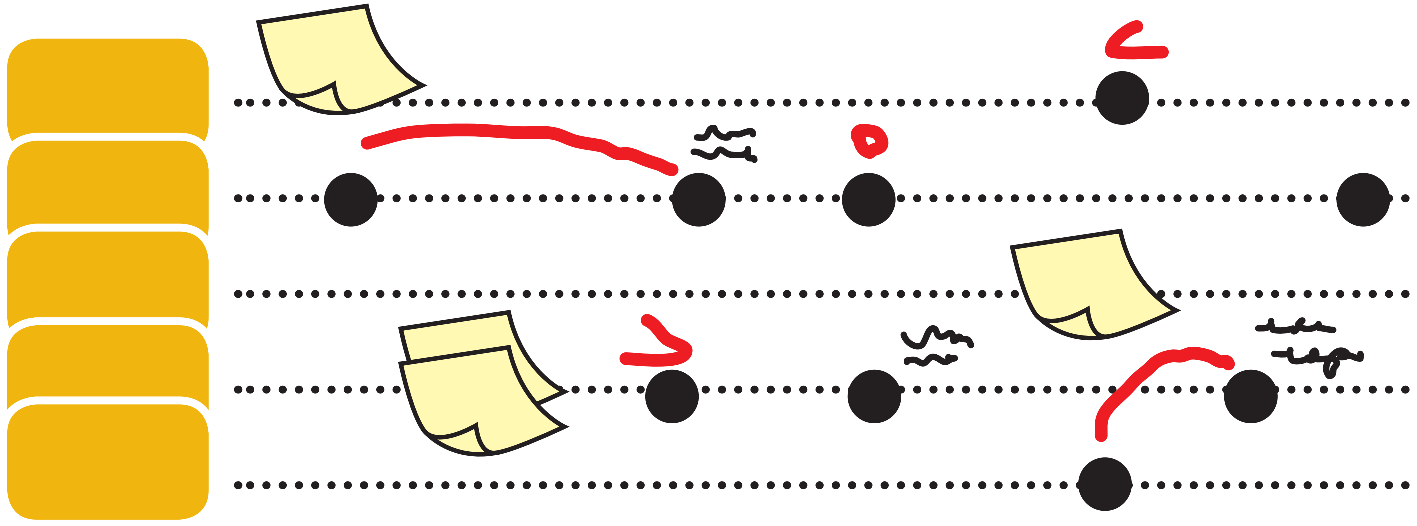
RELATIONSHIPS AND NOTATION

Many of the actions exemplified above depend on each other's input/output or have other types of relationships like before/after vs. parallel or different forms of dependence like the success of one influences the success of the other.

It can be very inspiring for a team to reflect on such relationships between their actions and try to draw them on the PROJECT SCORE

The actual Notation might differ from project to project depending on the actual relationships that are useful to identify between the different actions and it does not need to be formalized.

Improvising during the Checkpoint gatherings might prove an inspiring and playful group experience that will add to build common understandings.



GIT*

A very interesting feature of the github/gitlab platforms, which is worth exploring is the so-called "Issues". What is interesting with this feature is that it has the potential of mixing the design with the software process in very interesting ways, but it is not straightforward how to achieve a good balance since the primary use of github/gitlab is by the software developers and mixing bug fixes and low-level technical issues with high-level UX design might be complex.

In short, github can be a little intimidating for non-technical people but mostly in terms of content and not in terms of functionality since as a discussion forum, for example, github is rather user-friendly. In any case, github will likely not succeed to engage all typologies of actors in a given community. For this, it is important to include in the team "translators" that can get feedback from the field and translate it into the more technical language that will be developed inside github.

CANVASES

MethodKit provides nicely designed versions of standard and customized version of "Canvases" for Strengths Weaknesses Opportunities Threats (SWOT) analyses and business models, which are freely available as pdf.

PLANNING FOR REAL

There are numerous methodologies for community engagement through participatory workshops of various kinds. If there are experts on this topic as part of the team, most probably they will have their own preferences about which event, workshop, brainstorming session methodology is most appropriate and it is very important that someone feels confident and comfortable in applying such a methodology in public.

The "planning for real" methodology is an especially interesting approach not typically present in related handbooks:

1. create a physical model of the area of interest;
2. catch people's eye and interest for simply coming over at the meeting in the first place, in a non-committal free and open way;
3. open up the discussions toward expressing interests, values and desires;
4. try things out, before making commitments;
5. create implementation options by means of triangulators (e.g., option cards);
6. engage those interested gradually in the participatory process, by getting nearer and nearer to a commitment, and develop an action plan according to the revealed skills;
7. form action groups around a particular kind of action.

FACILITATION

There are numerous event facilitation guides There are numerous event facilitation guides but in our context the Project Planning and Facilitation tools by OTI, are a good starting point.

TIPS ON USING THE METHODOLOGY

Notice that it will be very rare that all required skills and resources will be present from the beginning in a team. For this it is important to creatively plan for "shortcuts" in the proposed methodology and make it possible to develop a project even with the tiniest resources.

As in music, it is possible to produce interesting results even with one chord. So, don't hesitate to choose only those threads of action according to the project's resources and community needs.

Be ready for improvisations and "shortcuts" in the implementation of the overall methodology. What is really important is that the effort invested produces re-usable results that add to a common pool of achievements in this area. For this, the development of adequately "free" software and the corresponding documentation are a fundamental requirement.

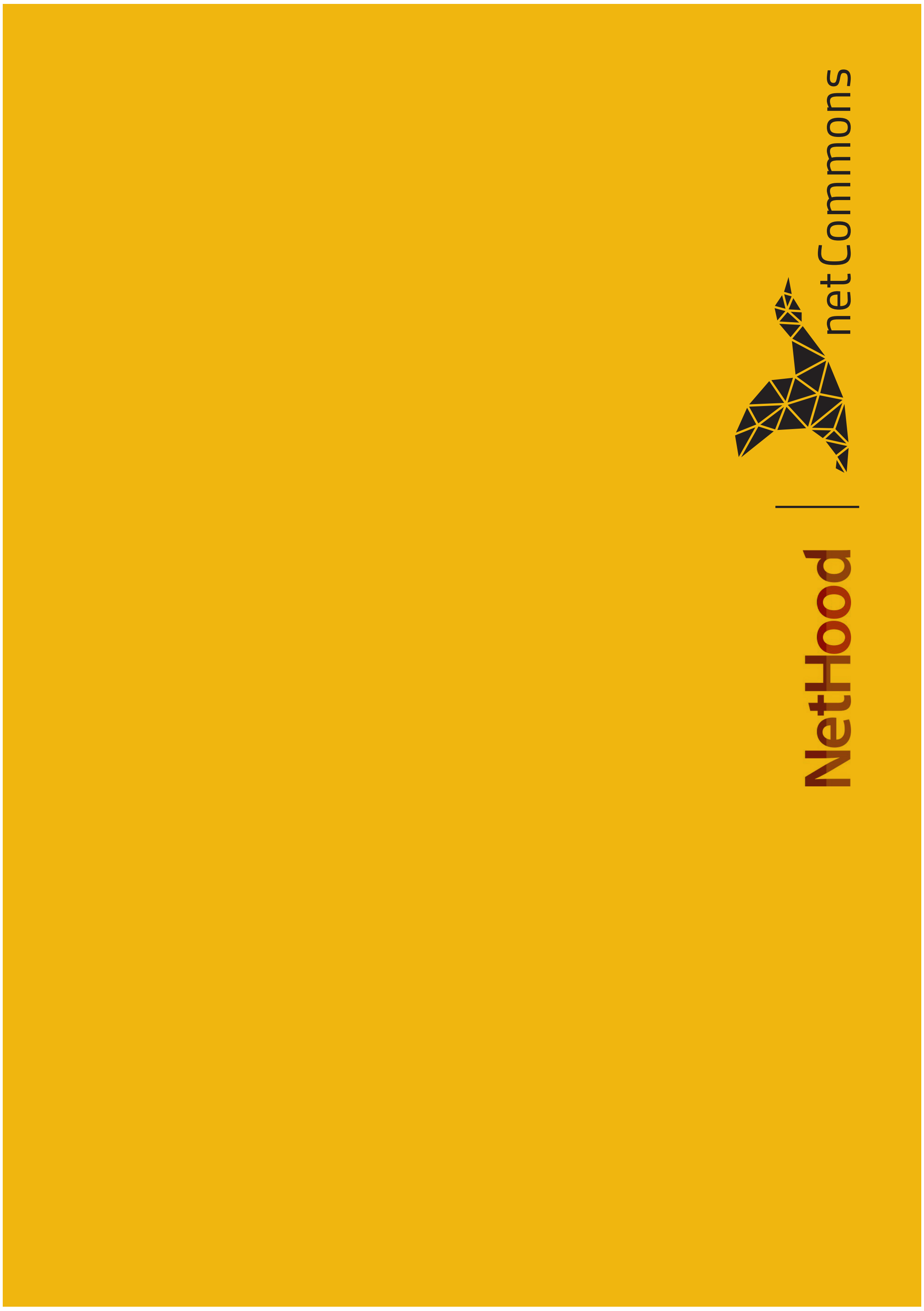
Finally, note that the focus of the suggested actions and guidelines are on activities that are important for the communication between different teams. Internally each process can follow more detailed and relevant methodologies for the corresponding tasks.

FURTHER READING

The netCommons reports D3.1 and D3.3 contain a detailed account of a participatory design process that is under development in the area of Sarantaporo. The netCommons report D3.6 contains a detailed review of the initial version of the methodology produced based on our experiences in Sarantaporo. All reports are available at:
https://www.netcommons.eu/?q=content/deliverables-page

Co-creation of the methodology:

You can contribute to the CommunityServer wiki with feedback on currently listed methodological elements and proposing new ones based on your experience in this wiki:
http://nethood.org/studio/


NetHood | netCommons

PLACE

What are the special characteristics of the place where your local application will be deployed.

COMMUNITY NETWORK

What type of CN will host your local application.

LOCAL COMMUNITY

How is the local community related to its CN.

TEAM

Available skills and perspectives in your team.

RESOURCES

More or less hidden available resources that you can use.

NEEDS

Why is it important to build software suitable to run in a local environment independently from the Internet?

TEMPO

Establish an appropriate rhythm for the project's members gather to discuss about their processes and possible inter-dependencies between them.

TRANSLATION

Consider the need for translations in language and concepts between members of the team and between the team and the community.

TOOLS

The tools used for facilitating brain-storming and playful interactions.

SHORTCUTS

Creative ways to feel gaps of skills or resources for a successful project.

RELATIONSHIPS

Consider how your different actions depend and/or influence each other.

REAL MAP

Use printed real maps during brain-storming and participatory design processes

LISTENING

How do you listen to the communities characteristics and needs.

COMMUNITY ENGAGEMENT

Events and processes that help you engage the community in the design of the applications.

DOCUMENTATION

Capture and communicate your understanding of community needs and special characteristics.

LEARNING

Have you reserved enough time for training the community in new concepts?

TRUST BUILDING

Show who you are and be engaged, to be trusted.

NETWORK VISUALIZATION

Tangible ways through which your local network, infrastructure and applications are made visible in the ground, like posters, printouts and various artefacts.

DEDICATED SPACES

Permanent or temporal locations where someone can learn about the community network and its local applications, engage in learning and participatory process, and meet in person the people behind the project.

SPACE INFRASTRUCTURE

Internal and external design of the space, tools to facilitate interactions, and artefacts to communicate the selected framing and overall identity of the network.

RUNNING THE SPACE

Governance mechanisms and processes that guarantee the sustainability of the space and their proper functioning.

HYBRID INTERACTIONS

Links between the digital and the physical through displays and other visualizations of online interactions combined with face2face gatherings.

APPLICATION

Continuous design for user experience, needs, and appropriation in the core functionality offered by your local application.

SOFTWARE DEVELOPMENT

A realistic plan for the deployment of different versions of the software application, including a Minimum Viable Product

ADMINISTRATION

Making things easy and flexible for the administrator is the key ingredient of a local application for CNs

CURATION

Online spaces, like physical ones, need presence and curation. No one will use your application if you are not "there".

CONTINUOUS FEEDBACK

The users of the software should be encouraged and facilitated to send you feedback on issues and feature requests, which can both help you improve the functionality but also reveal their needs and priorities.

PROJECT IDENTITY

Permanent or temporal locations where someone can learn about the community network and its local applications, engage in learning and participatory process, and meet in person the people behind the project.

COMMUNICATION

Share your project's objectives and results.

ORGANIZATION

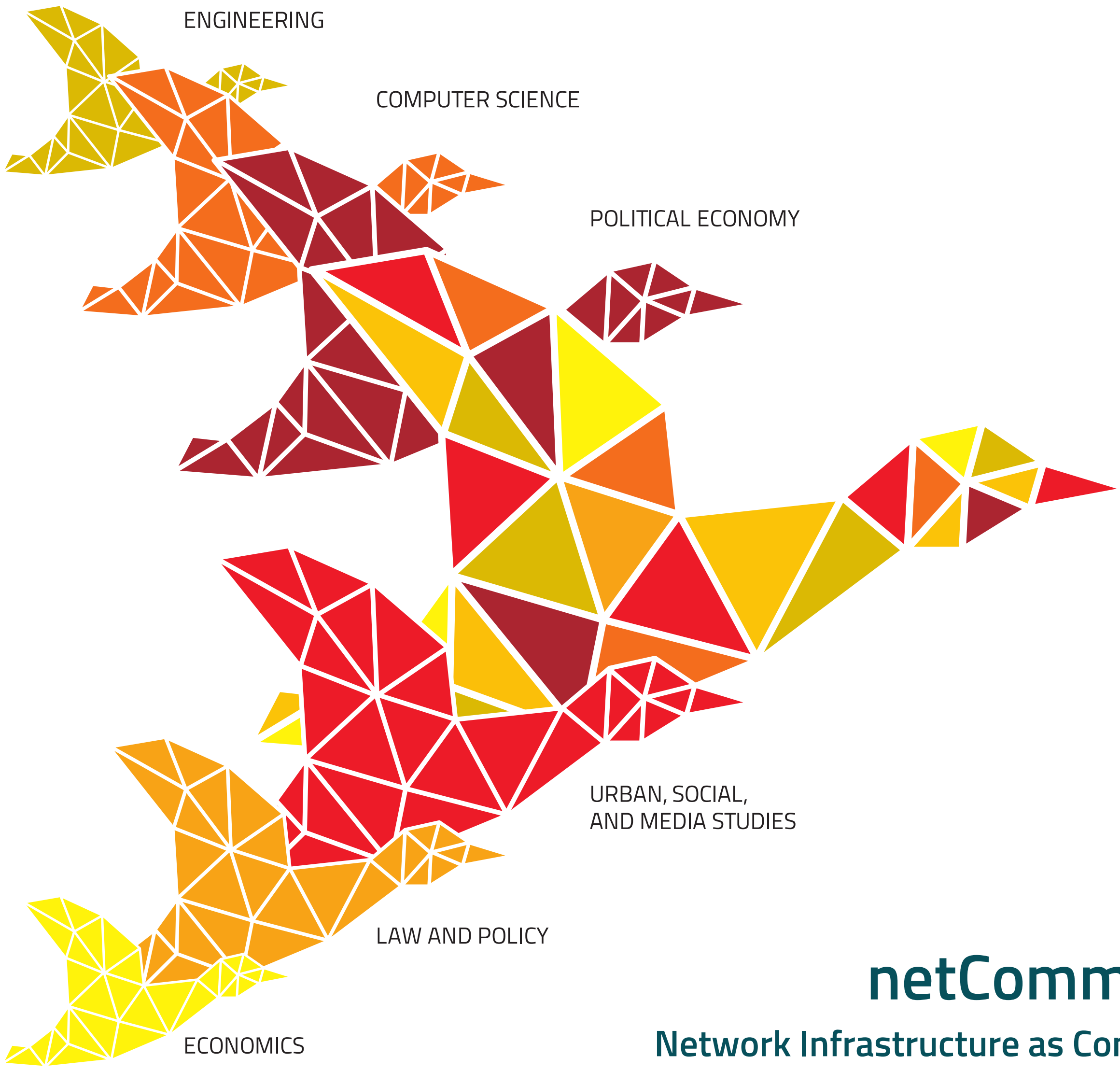
Make sure the right people are working on the right tasks

NETWORKING

Build relationships with local actors
but also external communities
and international networks.

FUNDING

Collaborate with the community
to find complementary funding for
your project.



netCommons

Network Infrastructure as Commons

Report on the Results of the Socio-Technological Experimentation of Open Source Software

Deliverable Number D3.5
Version 1.0
January 24, 2019

netCommons.eu