Co-funded by the
Erasmus+ Programme
of the European Union

On Service Optimization in Community Network Micro-Clouds

NUNO APOLÓNIA

Doctoral thesis in Distributed Computing
Universitat Politècnica de Catalunya
Department of Computer Architecture (DAC)
Computer Networks and Distributed Systems Group (CNDS)
Barcelona, Spain 2018
and
Doctoral thesis in Information and Communication Technology
KTH Royal Institute of Technology
School of Electrical Engineering and Computer Science
Stockholm, Sweden 2018

KTH School of Electrical Engineering
and Computer Science
SE-164 40 Kista
SWEDEN

TRITA-EECS-AVL-2018:53
ISBN 978-91-7729-900-4

Akademisk avhandling som med tillstånd av Kungl Tekniska högskolan framläggs till offentlig granskning för avläggande av doktorsexamen i Informations-och kommunikationsteknik måndagen den 21 September 2018 10:00 i Sal C Kungl Tekniska högskolan, Kistagången 16, Kista.

Tryck: Universitetsservices US AB

Abstract

Internet coverage in the world is still weak and local communities are required to come together and build their own network infrastructures. People collaborate for the common goal of accessing the Internet and cloud services by building Community networks (CNs).

The use of Internet cloud services has grown over the last decade. Community network cloud infrastructures (i.e. micro-clouds) have been introduced to run services inside the network, without the need to consume them from the Internet. CN micro-clouds aims for not only an improved service performance, but also an entry point for an alternative to Internet cloud services in CNs. However, the adaptation of the services to be used in CN micro-clouds have their own challenges since the use of low-capacity devices and wireless connections without a central management is predominant in CNs. Further, large and irregular topology of the network, high software and hardware diversity and different service requirements in CNs, makes the CN micro-clouds a challenging environment to run local services, and to achieve service performance and quality similar to Internet cloud services.

In this thesis, our main objective is the optimization of services (performance, quality) in CN micro-clouds, facilitating entrance to other services and motivating members to make use of CN micro-cloud services as an alternative to Internet services. We present an approach to handle services in CN micro-cloud environments in order to improve service performance and quality that can be approximated to Internet services, while also giving to the community motivation to use CN micro-cloud services. Furthermore, we break the problem into different levels (resource, service and middleware), propose a model that provides improvements for each level and contribute with information that helps to support the improvements (in terms of service performance and quality) in the other levels.

At the resource level, we facilitate the use of community devices by utilizing virtualization techniques that isolate and manage CN micro-cloud services in order to have a multi-purpose environment that fosters services in the CN micro-cloud environment.

At the service level, we build a monitoring tool tailored for CN micro-clouds that helps us to analyze service behavior and performance in CN micro-clouds.

Subsequently, the information gathered enables adaptation of the services to the environment in order to improve their quality and performance under CN environments.

At the middleware level, we build overlay networks as the main communication system according to the social information in order to improve paths and routes of the nodes, and improve transmission of data across the network by utilizing the relationships already established in the social network or community of practices that are related to the CNs. Therefore, service performance in CN micro-clouds can become more stable with respect to resource usage, performance and user perceived quality.

Sammanfattning

Internettäckningen i världen är fortfarande svag och lokala samhällen måste samarbeta och bygga egna nätverksinfrastrukturer. Människor samarbetar för det gemensamma målet att få tillgång till Internet och molntjänster genom att bygga community networks (CN).

Användningen av Internet-molntjänster har ökat under det senaste decenniet. Community Networkbaserade molntjänster (s.k. mikro-moln) har introducerats för att driva tjänster inom nätverket, utan att behöva nå dem via internet. CN-mikro-moln syftar inte bara till förbättrad serviceprestanda, men också en startpunkt för ett alternativ till Internet-molntjänster i CN. Men anpassningen av de tjänster som ska användas i CN-mikromoln har sina egna utmaningar sedan användningen av lågkapacitetsenheter och trådlösa anslutningar utan central kontroll är dominerande i CN. Vidare, stor och oregelbunden topologi hos nätverket, stor variation hos mjukvaran och hårdvaran, och olika krav på service i CN, gör CN-mikromoln till en utmanande miljö för att driva lokala tjänster och för att uppnå serviceprestanda och kvalitet som liknar Internet-molntjänster.

I denna avhandling är vårt huvudmål att optimera tjänster (prestanda, kvalitet) i CN-mikromoln, vilket underlättar ingången till andra tjänster och motiverar medlemmar att använda sig av CN-mikro-molntjänster som ett alternativ till Internet-tjänster. Vi presenterar ett sätt att hantera tjänster i CN-mikro-molnmiljöer för att förbättra serviceprestanda och kvalitet som kan liknas vid Internet-tjänster, samtidigt som de motiverar samhället att använda CN-mikro-molntjänster. Dessutom delar vi upp problemet i olika nivåer (resurs, service och middleware), föreslår en modell som ger förbättringar för varje nivå och bidrar med information som hjälper till att stödja förbättringarna (när det gäller serviceprestanda och kvalitet) på de andra nivåerna.

På resursnivån underlättar vi användningen av gemensam utrustning genom att använda virtualiseringstekniker som isolerar och hanterar CN-mikro-molntjänster i syfte att ha en mångsidig miljö som främjar tjänster i CN mikro-moln miljön.

På servicenivån bygger vi ett övervakningsverktyg skräddarsytt för CN-mikromoln som hjälper oss att analysera servicebeteende och prestanda i CN-mikromoln.

Därefter möjliggör den insamlade informationen anpassning av tjänsterna till miljön för att förbättra deras kvalitet och prestanda i CN miljöer.

På middleware-nivån bygger vi överlagringsnät som det huvudsakliga kommunikationssystemet enligt den sociala informationen för att förbättra nodernas banor och rutter och förbättra överföringen av data över nätverket genom att använda de relationer som redan upprättats i det sociala nätverket eller i gemenskapen av praxis som är relaterade till CN. Därför kan tjänsteutövning i CN-mikro-moln bli stabilare med avseende på resursanvändning, prestanda och användarens uppfattade kvalitet.

Resumen

Acceder a Internet sigue siendo un reto en muchas partes del mundo y las comunidades locales se ven en la necesidad de colaborar para construir sus propias infraestructuras de red. Los usuarios colaboran por el objetivo común de acceder a Internet y a los servicios en la nube construyendo redes comunitarias (RC).

El uso de servicios de Internet en la nube ha crecido durante la última década. Las infraestructuras de nube en redes comunitarias (i.e., micronubes) han aparecido para albergar servicios dentro de las mismas redes, sin tener que acceder a Internet para usarlos. Las micronubes de las RC no solo tienen por objetivo ofrecer un mejor rendimiento, sino también ser la puerta de entrada en las RC hacia una alternativa a los servicios de Internet en la nube. Sin embargo, la adaptación de los servicios para ser usados en micronubes de RC conlleva sus retos ya que el uso de dispositivos de recursos limitados y de conexiones inalámbricas sin una gestión centralizada predominan en las RC. Más aún, la amplia e irregular topología de la red, la diversidad en el hardware y el software y los diferentes requisitos de los servicios en RC convierten en un desafío albergar servicios locales en micronubes de RC y obtener un rendimiento y una calidad del servicio comparables a los servicios de Internet en la nube.

Esta tesis tiene por objetivo la optimización de servicios (rendimiento, calidad) en micronubes de RC, facilitando la entrada a otros servicios y motivando a sus miembros a usar los servicios en la micronube de RC como una alternativa a los servicios en Internet. Presentamos una aproximación para gestionar los servicios en entornos de micronube de RC para mejorar su rendimiento y calidad comparable a los servicios en Internet, a la vez que proporcionamos a la comunidad motivación para usar los servicios de micronube en RC. Además, dividimos el problema en distintos niveles (recursos, servicios y middleware), proponemos un modelo que proporciona mejoras para cada nivel y contribuye con información que apoya las mejoras (en términos de rendimiento y calidad de los servicios) en los otros niveles.

En el nivel de los recursos, facilitamos el uso de dispositivos comunitarios al emplear técnicas de virtualización que aíslan y gestionan los servicios en

micronubes de RC para obtener un entorno multipropósito que fomenta los servicios en el entorno de micronube de RC.

En el nivel de servicio, construimos una herramienta de monitorización a la medida de las micronubes de RC que nos ayuda a analizar el comportamiento de los servicios y su rendimiento en micronubes de RC. Luego, la información recopilada permite adaptar los servicios al entorno para mejorar su calidad y rendimiento bajo las condiciones de una RC.

En el nivel de middleware, construimos redes de overlay que actúan como el sistema de comunicación principal de acuerdo a información social para mejorar los caminos y las rutas de los nodos y mejoramos la transmisión de datos a lo largo de la red al utilizar las relaciones preestablecidas en la red social o la comunidad de prácticas que están relacionadas con las RC. De este modo, el rendimiento en las micronubes de RC puede devenir más estable respecto al uso de recursos, el rendimiento y la calidad percibidas por el usuario.

Acknowledgments

I am hugely indebted to my advisors, Leandro Navarro (UPC), Félix Freitag (UPC), Sarunas Girdzijauskas (KTH) and Vladimir Vlassov (KTH), without whose guidance and support, this journey would never have been possible.

I would like to give a heartfelt, special thanks to my colleagues and friends from the Distributed Systems Group, DSG at UPC Mennan Selimi (UPC), Roshan Sedar (UCL), Emmanouil Dimogerontakis (UPC), Navaneeth Rameshan (UPC), Vamis Xhagjika (UPC), Khulan Batbayar (UPC), João Neto (IST), Jawad Manzoor (UPC), Leila Sharifi (IST), Anwaar Ali (University of Cambridge) who were a great pleasure to work with and provided a great work environment, the hours of discussions over a ping-pong table and the dire needed coffee breaks at FIB bar. A very special thanks to my colleagues from the EMJD-DC programme and KTH-ICT department Sana Imtiaz (UCL), Muhammad Bilal (UCL), Zainab Abbas (KTH) for their help and patience whenever I required.

My profound gratitude also goes to the Guifi.net and Pangea.org members that kindly help with various phases of this research, my very special thanks goes to Roger Pueyo Centelles, Agustì Moll, Roger Baig Viñas, Llorenç Cerdà-Alabern, Jorge Florit and Ferran Olid.

I would also like to thank my co-author Stefanos Antaris (University of Cyprus), who help me immensely when I really needed.

I would like to thank all the people whose names I did not include here, but provided me with inspiration, help and made it possible to write this thesis.

And finally, I am deeply thankful to my family for their love, support and sacrifices.

Nuno Apolónia

This work was funded by European Commission (EACEA) through the Erasmus Mundus doctoral fellowship, via Erasmus Joint Doctorate in Distributed Computing (EMJD-DC) programme. This work was also supported by European Community Framework Programme 7 FIRE Initiative project Community Networks Testbed for the Future Internet (CONFINE), FP7-288535, CLOMMUNITY, FP7-317879, the European H2020 framework programme projects netCommons (H2020-688768) and LightKone (H2020-732505), and by the Universitat Politécnica de Catalunya-BarcelonaTECH and Spanish government under contract TIN2016-77836-C2-2-R and TIN2013-47245-C2-1-R.

Contents

List of Figures

List of Tables

List of Acronyms

CN	Community Networks
CoP	Community of Practice
DOSNs	Decentralized Online Social Networks
IaaS	Infrastructure as a Service
IoT	Internet of Things
LXC	Linux Containers
OS	Operating System
OSNs	Online Social Networks
P2P	Peer to Peer
QoE	Quality of Experience
QoS	Quality of Service
RTT	Round-Trip Time
SNs	Social Networks
VM	Virtual Machine
VoIP	Voice over Internet Protocol
VPN	Virtual Private Network
WCN	Wireless Community Networks

Chapter 1

Introduction

In recent years, cloud technology has been spreading towards the edges of the network. In fact, modern cloud computing utilizes both edge computing and data center clouds. Edge computing [1, 2, 3] has been growing in its use due to the fact that resources have increased in their computational power, while also maintaining their proximity with the users. Edge cloud computing utilizes the resources at the edge of the network to create a cloud environment. The introduction of the Internet of Things (IoT) has also brought other resources (e.g. sensors) onto the edges of the network, which can use processing at the edges and in data center clouds, this notion has been named as *fog computing* [2] and it is a use case of Edge cloud computing.

The Community networks (CNs) are a collaborative network, built and operated by citizens, created for the common goal of accessing the Internet and cloud services where it was not possible. The network is created through the use of heterogeneous devices interconnected through antennas or limited wired connections throughout different regions, such is the case of Guifi.net[1],

[1] `http://guifi.net`

FunkFeuer[2], AWMN[3] and Freifunk[4]. The limited connectivity towards the Internet is made through gateways or proxies in specific locations of the network, and shared by the community. The development of Community network cloud infrastructures (i.e. micro-clouds) has introduced cloud services inside of the network, without the need to consume them from the Internet. Hence, CN micro-clouds is another case of edge cloud computing, which utilizes the low-resource devices and wireless network at the edges of the network to create a cloud environment.

In this thesis we focus on the particular case of edge cloud computing, micro-clouds [4, 5] within a Community network. The focus on CN micro-clouds allows us to have resource diversity, different geo-location on resources and a participation of the community. These particular aspects may not be found in other cases of edge cloud computing. Thus the impact of this thesis is to help and take part of a broader construction of edge cloud computing in Community networks.

CN micro-clouds differ from fog computing, mobile computing [6] and data center clouds since there is no use of remote data centers, and the focus is on reusing the computing resources on those shared devices available in the community. CN micro-clouds share similar characteristics with fog and mobile computing with respect to the use of wireless methods and computing capabilities of the devices available. However, CN micro-clouds are a collaboration where the community builds and manages the infrastructure, and the use and deployment of the services is dependent on the users choices or available devices instead of the use of data center resources and services' requirements. The CN micro-cloud type of cloud computing focuses on services that the community

[2] `http://funkfeuer.at`
[3] `http://awmn,gr`
[4] `http://freifunk.net`

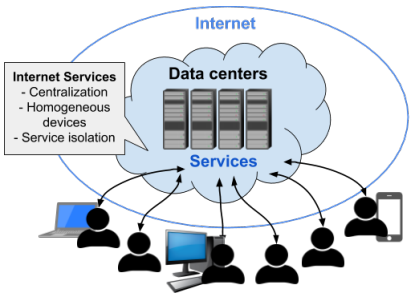
share with each other, such as live video streaming through Peerstreamer [7], or distributed storage through Tahoe [8], among many other services[5].

The use of Internet services (data center clouds) has put a strain and higher cost in the limited connectivity that CNs have to the Internet. Therefore, CN micro-clouds were introduced as an alternative to data center clouds, in order to minimize cost and foster Internet services within CNs. However, the adoption of CN micro-clouds is tied with how users perceive service quality, performance and stability. The main research question is how can services be improved in CN micro-clouds and match how services are perceived from data center clouds?

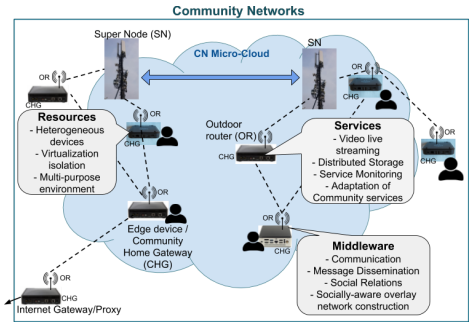
The main objective of this thesis is the optimization of services in CN Micro-Clouds, facilitating entrance to more Internet services and motivating the community to favor them from within CNs as alternative to those in the Internet, which leads to a lower cost (i.e. bandwidth, latency, throughput, monetary) with regards to the limited connectivity that CNs have to the Internet.

In this thesis we tackle the problem by breaking it down into three levels: resource, service and middleware level. Each level allows us to focus on different aspects which compose the CN micro-clouds, and grants information that is used on different levels in order to improve service performance, quality and experience perceived by the users. Figure 1.1 depicts a scenario of Data center cloud, where typically data centers are centralized, have homogeneous devices running services, and are accessed by the users through the Internet. On the other hand, CN micro-clouds are depicted with heterogeneous devices spread across the CNs, where a subset of the devices are used in each micro-cloud, services run on the devices that are shared by the users to the community.

[5] `http://guifi.net/node/3671/view/services`



(a) Typical scenario of Data center clouds, depicting the centralization of devices running services that are accessed by users through the Internet.



(b) CN Micro-cloud overview depicting different devices that compose the micro-cloud, including the three levels resources, services and middleware where improvements are done to augment service performance and quality. Different connectivity between super nodes and outdoor routers, depicting different geo-locations, distances and communication.

Figure 1.1: Comparison between (a) CN Micro-clouds and (b) Data center Clouds. In Data centers, users are separated from services and where they run. On CN Micro-clouds users share their own resources running services from the available devices.

The communication is done through the outdoor routers (OR) between devices of different households, and different geo-locations between super-nodes (SN). Each of the levels are represented as the main points towards improving service performance and quality in CN micro-clouds.

In the particular case that we study in CNs, our solution within the resource level comprehends the creation of a multi-purpose environment through the use of virtualization technologies. The use of this environment allows services in the CN micro-clouds to utilize resources without being deployed in bare-metal, as is the case with current CN micro-clouds. Furthermore, it allows for a fair and isolated use of the devices where services are deployed within its own

partition. Also, the use of a multi-purpose environment grants motivation for the members to share their resources and use the available services, by making certain that owners can still use their own devices whilst sharing it with the community.

The CN micro-clouds services have either been ported from data center clouds, or not built for CN environment, which creates a problem to deploy them in the environment. The service performance and quality is perceived as lower than what happens in data centers clouds, and therefore users still prefer the data center cloud services. At the service level, we focus our solution in the creation of a monitoring tool tailored for CN micro-clouds, that helps to analyze the service behavior and adapt service configuration that improves service performance and quality under CN micro-clouds, without having to change the environment. It allows to understand how services can be adapted towards being used in CN micro-cloud environments.

In the middleware level, we focus on the overlay networks as main communication system (i.e. message distribution), such as the case with SERF[6], where data dissemination occurs in order to inform all nodes of service availability, communication and location. The problem in creating the overlay networks, is the amount of collateral nodes used (relay nodes) that only relay messages towards other nodes, disregarding the workload or social relations. Our solution in this level is to induce social information (i.e. social graphs of user relations) into the creation of overlay networks in order to improve routing/path of nodes used for message dissemination. Furthermore, the use of a gossip-enabled networks [9], where dissemination is done according to the neighborhood of each node, allows the communication between nodes regardless of the underlying infrastructure. However it has its own limitations

[6]https://www.serfdom.io/

when not including information about the resources, services and social aspects of the micro-cloud infrastructure.

The use of social information to improve services communication [10] and overlay networks [11, 12] has been proved to be successful regarding the timely message dissemination in P2P networks, used by services within CN micro-clouds, and the use of fewer relay nodes. Thus, it minimizes routing and maximizes dissemination across the network. The use of gossip protocols [12] enhances the infrastructure without relying on specific knowledge about paths, nodes or resources in order to disseminate messages.

1.1 Problem Statement

Community networks were created by citizens in order to fill a lack of Internet infrastructure in locations where Internet service providers would not provide Internet. The community collaborates by using their own devices to reach places where Internet was available in order to expand Internet access to regions that had none. The concept of deploying services inside of the CNs, that could potentially be used by the community has grown, such as radio stations, data storage, live video cameras, among other examples[7]. This has also created an alternative for Internet services, which would favor services from within CNs alleviating connectivity to the Internet.

The fact that cloud services have become predominant, has also brought a consequence to CNs. The network has increased the load on its gateways or proxies to the Internet in order to access cloud resources and services. Thus, the creation of CN micro-clouds within community networks is an alternative move towards bringing cloud services closer to the community, while reducing

[7] `https://guifi.net/node/3671/view/services`

communication with the rest of the Internet helping to alleviate the limited connectivity that exists to the Internet.

CN micro-cloud services are in many ways similar to data center cloud services, e.g. data storage. However, services running in community network environments suffer from different issues that did not exist in data centers, such as varied latency between devices, the use of heterogeneous devices, the sharing of devices by the community itself without any centralized operator. Therefore, in order to bring Internet services into CN micro-clouds, the environment requires to be prepared, and to match the service quality and performance which was perceived from data center services. CN members also require to share their resources with the community to create a collaborative environment and to foster micro-cloud environments.

Moreover, services within CN micro-clouds are not adapted to the environment, which creates an issue when deploying more services within CN environments. This may be perceived by the users as a unattractive service (with low quality, performance) that would be avoided to use as an alternative.

The message and data dissemination is mostly required by CN micro-cloud services in order to reach the users and make use of the micro-cloud resources. This is because the services can use different devices that are not co-located and diverge from how services behave and run in data centers. Therefore, issues arise in the services' main communication system, like the routing between devices, which are not found in data centers due to the use of heterogeneous devices and wireless networks, and that hinder the performance and quality of the services.

1.2 Research Questions

The use of services in micro-clouds are becoming more prevalent, thus it is necessary that improvement to message dissemination and communication routing becomes a focus. Otherwise, the network may not guarantee enough support for all the services, since resources have more constraints than data centers. Therefore, by enhancing services with the support of overlay networks that include information about resources, services and social properties guarantees near-optimal paths for nodes communication and message distribution. To this end, the following questions are hereby addressed.

RQ1: How can the support of virtualization techniques improve service performance and quality within the CN micro-clouds? How does the community share their resources to be used in CN micro-cloud and make use of such environment?

Micro-cloud services can be very diverse, such as live video streaming, distributed storage or service announcement. Each service has its own requirements, being it resource or network computationally demanding. Furthermore, micro-clouds are built by a community, where shared devices are used to increase the cloud resources. The use of virtualization is necessary to guarantee a fair use of resources across the community, since micro-clouds resources are donated and shared among the community itself.

The use of virtualization technologies such as Linux containers or virtual machines are a path towards an improved used of the resources available. Therefore, services can be deployed with ease, while also making sure that there are enough resources for services across the network where each service utilizes only an allotted amount of the community resources, making a fair use of resources for all members.

The use of partitioning of resources by using virtualization techniques enables each device to handle and structure the services that they can support for use by the community. Moreover, the optimization at the resource level begins by using virtualization techniques that provide isolation for different services in the same devices, while giving the owners the ability to use their own devices for themselves.

RQ2: How do services behave and perform in CN micro-clouds settings? Can services be optimized for CN environments without modifications in other levels?

CN micro-clouds differ from data centers in that resources are heterogeneous. In fact, most of the network is composed by wireless infrastructure and low-capacity devices, e.g. resources may be affected by weather conditions, or the position of the antennas. Therefore, services require to be adaptable to the unstable network conditions, which is not an issue within data center clouds. It is required for CN micro-clouds to account for low-capacity devices, the excessive use of network links within specific time-frames, or simply the variable latency between devices in order to improve service performance and perceived quality.

Services are required to be adaptable to extreme conditions, that may not appear in data center environments. Thus, understanding service performance and how to augment its use regarding the CN environment is key to enhance service use in CN micro-clouds, while also attracting more of the community towards the CN micro-cloud services instead of relying on the data center cloud services.

Service monitoring is fundamental to understand the impact that services have within the network, and enable the proper configuration of each service to be

adaptable to different conditions. Thus, improvement of services (in terms of performance and quality) can begin by the adaption that comes from a tailored configuration of services towards different environments. Also, the adaptability depends on the type of service (such as network intensive, or computational intensive) and its usage. This leads to improve service performance and quality within CN micro-clouds as well as entrance to other services.

RQ3: How can the introduction of social information further improve services in CN micro-clouds? Does enhancing the middleware level with social information optimize the routing of message dissemination for services in such environments?

Social behaviour has a great influence on service performance, e.g. the demand on service usage within certain time frames or the use of trusted devices, among other examples. Therefore, social properties cannot be disregarded when creating network routes/paths for message dissemination used by services, included in CN micro-clouds. This leads us to understand how social properties can affect the overall performance of services, while improving overlay networks as main communication system, when dissemination of messages occurs.

In the middleware level, overlay networks serve as optimal paths towards dissemination of messages and use of resources in the network. Thus, by introducing social information we can create the paths towards the appropriate or close nodes, e.g. friends, social groups or collaborative groups, instead of blindly or randomly choosing nodes that may not guarantee a near-optimal path/route for message dissemination.

Furthermore, relay nodes in overlay networks usually serve as intermediary nodes towards a receiver. Therefore, improving the overlay network by reducing the number of relay nodes (that may also request such messages), means that

dissemination can be achieved without the use of unrelated nodes. In fact, such solutions have been presented in publish/subscribe systems for P2P networks, and guarantee that dissemination can occur with the use of low number of relay nodes that do not require the messages. However, we can put forth that each factor of the network, such as resources, services or users, should be considered as the input to create overlay networks and taking into consideration the nature of community networks, in order to improve network routing, load balance in the nodes and to minimize the nodes required when disseminating messages.

1.3 Objectives

The main objective of this thesis is the optimization of services in CN micro-cloud environments, facilitating the entrance to more services and motivate the community to share their devices and favor CN micro-cloud services as an alternative to Internet services, in order to alleviate the limited connectivity between CNs and the Internet.

Furthermore, this thesis breaks down the problem into three levels, where we focus on the main objective by improving the services performance and quality within CN micro-clouds. Therefore, our goal in each level is to improve services (in relation to performance and perceived quality) and gather information that helps the other levels understand how services are deployed and run in the CN micro-cloud environment and be able to make necessary changes that will improve the performance/quality of the services. In the resource level our objective is to create an environment that is used in CNs that fosters services and motivates members to share their devices with the community. In the service level our objective is to gather information about service behavior and usage in order to adapt service configuration in CN micro-clouds which

improves service performance/quality. In the middleware level our objective is to understand the communication system, and apply social information in order to enhance routing/path of service communication/message dissemination.

1.4 Contributions

1.4.1 List of Publications

The content of this thesis is based on the following published and peer reviewed publications.

Chapter 3 focuses on the optimization on resource level, by introducing several layers of virtualization, supported by the following publications:

P1 Nuno Apolónia, Felix Freitag, and Leandro Navarro. **Leveraging deployment models on low-resource devices for cloud services in community networks**. *Simulation Modelling Practice and Theory, 77:390-406*, 2017.

P2 Nuno Apolónia, Roshan Sedar, Felix Freitag, and Leandro Navarro. **Leveraging low-power devices for cloud services in community networks**. *In Future Internet of Things and Cloud (FiCloud), 2015 3rd International Conference on, pages 363-370. IEEE*, 2015. [Runner-up for Best Paper award]

Chapter 4 focuses on the optimization on service level, by measuring and evaluating service performance, supported by the following publications:

P3 M. Selimi, Nuno Apolónia, F. Olid, F. Freitag, L. Navarro, A. Moll,R. Pueyo, and L. Veiga. **Integration of an assisted p2p live streaming service in community network clouds**. *In 2015 IEEE 7th*

International Conference on Cloud Computing Technology and Science (CloudCom),pages 202-209, Nov 2015.

P4 Nuno Apolónia, F. Freitag, L. Navarro, S. Girdzijauskas and V. Vlassov. **Gossip-based service Monitoring Platform for Wireless Edge Cloud Computing**. *In 2017 IEEE 14th International Conference on Networking, Sensing and Control (ICNSC), pages 789-794*, May 2017.

Chapter 5 focus on the overlay optimization by introducing social properties of the network, supported by the following publications:

P5 Nuno Apolónia, S. Antaris, S. Girdzijauskas, G. Pallis, and M. Dikaiakos **Select: A distributed publish/subscribe notification system for onlinesocial networks**. *In 2018 32nd IEEE International Parallel and Distributed Processing Symposium*, May 2018.

P6 Nuno Apolónia, S. Girdzijauskas, Felix Freitag, and Leandro Navarro **Socially-aware Micro-Cloud Service overlays optimization in Community Networks**. *In Submission to Journal of Software: Practice and Experience, Special issue on Software tools and techniques for Fog and Edge Computing*, 2018.

1.4.2 Contributions

The contributions of this thesis originate from each level: Resource, Service and Middleware. We analyse and propose optimization for CN micro-cloud services by providing tools and algorithms that leverage resource, service and social information in order to improve communication of services in the CN environments. This is done by introducing virtualization techniques in order to organize services in each device, provide different configuration of

cloud services to improve its performance and QoS and build efficient overlay networks that improve message dissemination.

On the resource level our contribution, explained in more detail in Chapter 3, is the partitioning of the available devices through the use of virtualization techniques in order to give owners motivation to share their devices, while considering a multi-purpose environment in which services can be deployed, detailed in the work presented in [**P1**]. The multi-purpose environment assures that each service can run isolated, without interfering with each other on each device. Furthermore, the work considers the fact that allocation of the resources is necessary between owners and community in order to have a collaborative environment. The work detailed in [**P2**] demonstrates that services continue to run without quality degradation, which in turn can be motivational for the community to collaborate with their own resources, while assuring that resources can run cloud services within their own isolated environments. Therefore, one of the major contributions is a multi-purpose environment that fosters cloud services in CN environments, and a motivation for the community to practice collaboration in order to bring resources and services to the community network.

On the service level, we begin by understanding and analyzing service behavior in CN micro-clouds, explained in more detail on Chapter 4. The information on services is essential to understand and optimize how services run in the CN micro-clouds. Therefore, our first work relates to gathering of information on CN micro-cloud services by providing a platform for monitoring regarding service usage. The work detailed in [**P3**] demonstrates the use of gossip-enabled networks to enhance monitoring of services in CN micro-clouds, such that information flows through the network without interfering with major services and their performance. Furthermore, we give a comprehensive analysis

on services in particular live video streaming, detailed in the collaboration work of **[P4]**, which cultivates the understanding on how cloud services behave in CN micro-clouds. The use of a particular service, such as live video streaming, serves as representation of the behavior of other services that require network and computing infrastructure. Thus, our focus on such a service is considered to be important to understand how other services that can have similar properties would behave and be adapted on CN micro-clouds.

Moreover, CN micro-cloud services QoS and QoE is enhanced through the appropriate configuration when using low-capacity devices and wireless environments. The use of different parameters on services, is a required step in order to provide users an improved QoS and QoE. Data center cloud services are not prepared to be used within CN environments, because data centers have different capabilities. Therefore, aiming for compatibility with CN environments, our contribution is a comprehensive look on the alterations that cloud service require to withstand the CN environments without having to change the entire environment (devices, network).

On the middleware level, we optimize the overlay networks that serve as main communication system, by adding social information, which is detailed in Chapter 5. The overlay networks are created in order to enhance service communication between different instances or nodes. The use of relay nodes for message dissemination is a common method to forward messages to all nodes that require them. Therefore, the minimization of relay nodes within the work of **[P5]** is a contribution that optimizes service communication. However, transporting such work to community networks has its challenges, since community networks do not have a clear social network. Instead, the social interaction is based on community of practice and collaboration within the communities. The work in **[P6]**, enhances CN micro-cloud services by

exploiting the collaboration and established relations within community networks in order to build an overlay network that aggregates the community of practice, and therefore enhances service communication in CN micro-clouds.

1.5 Thesis Limitations

The aim of this thesis is to provide to the CN an improved model for service usage and message routing in CN micro-clouds, that mainly uses wireless infrastructures and low-capacity devices. The topic can be diverse and thus we focus on improving each layer of the infrastructure with respect to service performance and quality perceived, and leave out related aspects that are not in consideration, such as security issues that potentially hinder service deployment under CN micro-clouds. In this thesis we then assume that the community would not have intentional malicious acts towards the community built infrastructure, diminishing capacity, performance and quality of the infrastructure.

To improve the way services can handle message dissemination the use of overlay networks are regarded as improving the overall micro-cloud environment. Services use different devices and routes to communicate between users, nodes or applications. Therefore, optimization of message distribution can bring an overall stability to the network, by minimizing the required network resources for each service, while maximizing the resources available to all services. However, the underlying network needs to be accounted with in the improvement of routing/paths, in order to build a sustainable relation between the overlay and underlay networks (in terms of latency, bandwidth between nodes). Also, the improvement to the performance of CN micro-cloud services brings trust for the community to utilize them as an alternative to Internet

services, but does not impose any member to share devices or utilize services from CN micro-clouds.

The use of social information should come with the consent of the users, however in community networks the users are the main part and contributors of the network infrastructure. Therefore, the use of their social information is regarded as a benefit for the community. Issues can arise from the use of private information, and thus such acts should be evaluated towards being anonymous and to respect users privacy.

Security of private devices is also a concern, however it is outside of the scope of this thesis. Furthermore, security issues are very important when handling information and multiple devices from different sources. In fact, since the community networks are opened to any users, occurrence of malicious intent may need to become more prominent in the creation of CN micro-clouds.

Thesis Organization

The thesis described in Fig.1.2 includes an outline of the context, research questions, contributions and the accomplished evaluation.

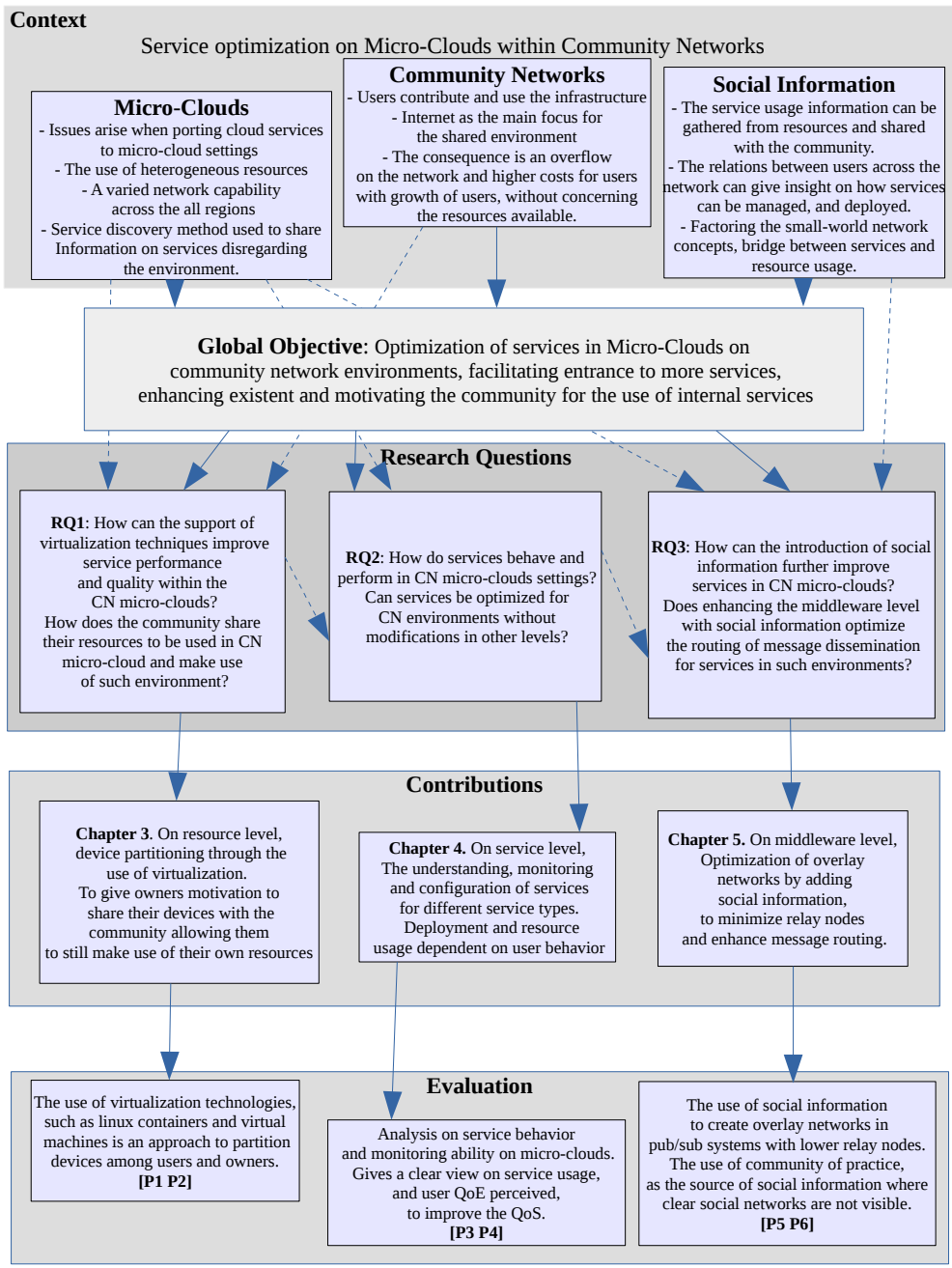


Figure 1.2: Thesis layout

Chapter 2

Background and Related Work

In this chapter, we review some of the background information and the relevant related work with specific focus on the most recent work regarding community networks (CN) aspects in resources, services and social infrastructure. We also review the previous approaches to service optimization in CN micro-clouds.

2.1 Background

2.1.1 Wireless mesh networks

Wireless mesh networks have emerged as a specific model in networking. The use of wireless mesh networks started to generate new concepts and paradigms towards mobility in devices, such as the case of vehicular networks [13]. Our case study for wireless mesh networks is Community Networks, which is explained below in more detail.

CNs have characteristic properties, such as, varied latency between nodes [14], dynamic routing changes and low-capacity devices used for node interconnection. Also, node connectivity is based on mesh routing protocols [15].

Resource sharing within CNs refers, in practice, to the sharing of network capacity from each device to route traffic through routers to its destination. The sharing of services, such as video streaming, storage, VoIP, a common practice in the Internet thanks to cloud computing, has slowly began to expand in CNs. Therefore, a community cloud model could accommodate services and/or resource sharing among community members without relying on the Internet or the major cloud providers.

Furthermore, by understanding services and resources properties, and how users interact, we can start to improve the organization of the CNs micro-clouds.

2.1.2 Edge Cloud Computing

Edge cloud computing is a specific case of cloud computing, which moves computation to the resources at the network edge. Thus, it uses edge resources and relies less on the Internet cloud resources. In this way, edge cloud computing works to share its own services and resources without having to go outside of the local network (i.e. Internet) to utilize cloud services.

There are significant differences between data center cloud environments and edge clouds. An important characteristic is the use of distributed low-capacity devices instead of centralized data centers with powerful computing devices. Additionally, the network between devices has higher variance in latency and bandwidth different to traditional data centers.

Community networks clouds are formed by a collaborative effort to create a computing platform where the infrastructure is shared between a number of organizations and members of the CNs in order to provide a platform for joint computing needs.

The CONFINE research project aimed at expanding community networks and facilitate the deployment of experimental or production services [16], as if they were deployed in any commodity cloud platform. More details can be found in subsection 2.1.3.2

2.1.3 Micro-Clouds

The Cloudy system distribution[1] (Cloudy OS) has been created under the Clommunity research project[2] to provide community networks an easy way to manage and deploy cloud infrastructures and interfaces for service discovery and deployment. The result is that any user can enjoy the benefits of cloud services which are freely available in the community without relying on any specific server infrastructure. The Cloudy OS is a free and open source software based on a customized version of Debian Linux. By default, it comes with an installation of the `tinc` distributed VPN[3] daemon which creates a secured private overlay network between hosts on the Internet. With the help of `tinc`, networked nodes can communicate securely with each other.

Therefore, Cloudy's development was driven by important aspects, such as the ease of usage, deployment in low-capacity devices, automated service discovery and services pre-configuration. Figure 2.1 indicates some of the already integrated types of services on the Cloudy CN distribution. An example of these services are the ones we consider in this work, the video streaming service such as PeerStreamer, and the discovery service named Serf.

Furthermore, community services are included in the distribution, in order to facilitate the process for edge cloud computing, e.g. Peerstreamer as a peer-to-peer based live streaming, Tahoe-LAFS as a decentralized storage

[1] http://cloudy.community
[2] http://clommunity-project.eu
[3] http://www.tinc-vpn.org

service, Syncthing[4] as a data synchronization between various storage nodes, among others. Also, the shared services within Cloudy are expected to be announced to the network (published/unpublished) in an automated way, when initiated by the users.

In addition, Cloudy uses Avahi[5] (or Serf[6] in latest versions), serving as a zero-configuration networking implementation, to publish and discover the services in the community. For the simplicity of service discovery, the Cloudy OS provides an interface that fetches service information through the overlay and lists the services in order for the users to easily connect to them.

In Cloudy, services can make use of an overlay network created through existing technologies, such as Serf, which specifically cluster nodes and manage service availability in the CNs micro-clouds.

2.1.3.1 Cloudy architecture

The internal architecture of the Cloudy distribution is depicted in Figure 2.1, inside the central rectangle. On the bottom part, the virtual Layer 2 over Layer 3 network provides the overlay to interconnect all the servers (nodes) in a micro-cloud. This overlay network is used in the service announcement and discovery processes, that respectively publish local information to the cloud and receive data from other cloud nodes.

Another special service module in the Cloudy instance is the distributed announcement and discovery of services. On the lower layer it provides the mechanisms and the infrastructure to other services to publish their information all over the CN. This is a valuable resource to orchestrate the CN cloud itself

[4] https://www.syncthing.net/
[5] http://www.avahi.org
[6] https://www.serfdom.io

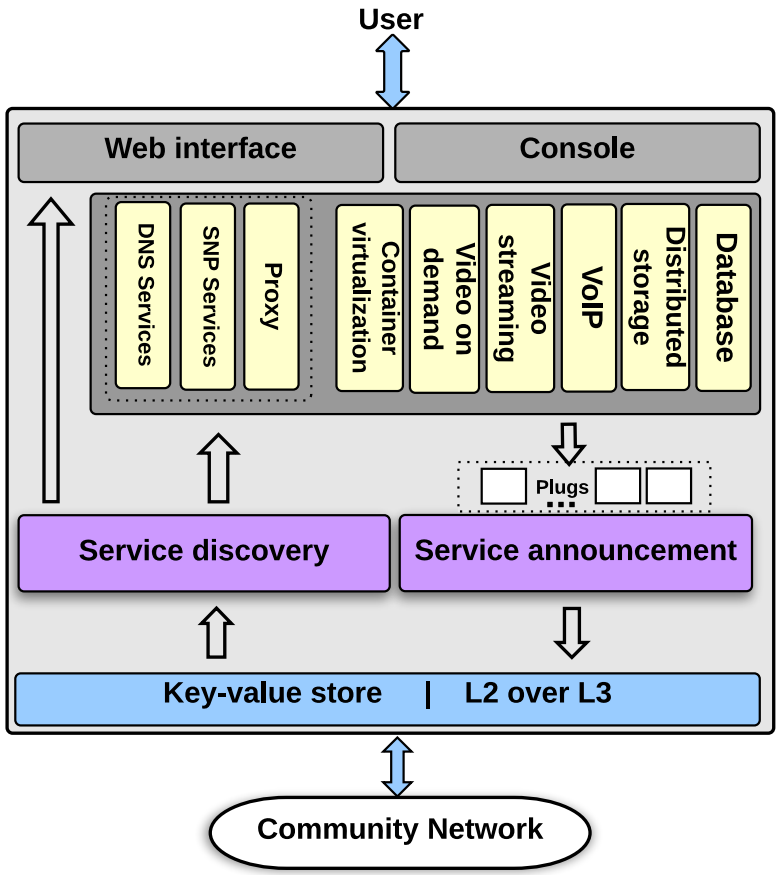


Figure 2.1: Cloudy architecture

as it allows room for self-discovery, management and federation of services and resources. On the user interaction layer, the DADS allows the end user to discover the available cloud services in the CN and decide which service provider to choose according to certain metrics (e.g. network round-trip time (RTT) to the services and number of hops).

The main block of Cloudy comprehends the CN services, stressing the important role of cloud services in the center of the diagram (see Figure 2.1). These

services are the ones that benefit from or embrace the CN cloud environment to operate or offer a richer quality of experience (the list in the diagram is non-exhaustive, but mentions key services like distributed storage or different ways to reach video contents). Among them, virtualization is a special case. While other services focus on interaction and contents for the end user, provision of Infrastructure as a Service (IaaS) by means of virtual machines focuses on fostering the deployment of other services that run on top of this infrastructure.

2.1.3.2 CONFINE Project: Community Networks Testbed for the Future Internet

The CONFINE project goal is to augment the capabilities of community networks by providing a platform for the existing community network in which users can use services with ease [16], as if they were deploying in any commodity cloud platform. Members of the CONFINE community network testbed are privileged to get a set of IP addresses (IPv4) in order for them to be able to run multiple services, as they may require. This enables members to run multiple services on top of the existing network, while sharing the resources with the community.

Community-Lab [17], shown in Fig. 2.2, is an infrastructure that provides a set of tools allowing researchers to easily deploy, run, monitor and experiment community cloud services, protocols and applications in a real community IP network (Guifi.net[7], FunkFeuer[8], AWMN[9] and Freifunk[10]) instead of simulated environments.

[7] `http://guifi.net`
[8] `http://funkfeuer.at`
[9] `http://awmn,gr`
[10] `http://freifunk.net`

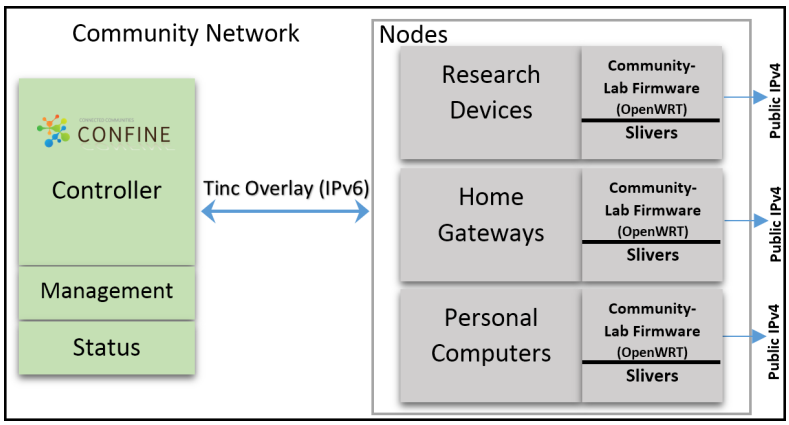


Figure 2.2: System Overview of the Community-Lab Testbed

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

Each Community-Lab node can contain several slivers, shown in Fig. 2.2, which are grouped at a higher level in slices. As such, a slice is defined as a set of resources spread across several physical devices in the testbed which allows users to run experiments over it. A sliver is defined as the partition of the resources (or virtual machine) of a community node assigned to a specific slice.

The purpose of the controller is to manage and control the testbed through simple operations such as managing users, nodes, slices and slivers. This

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

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

To be part of the Community-Lab infrastructure the devices require to operate a specific operating system, based on OpenWrt[11] configured to provide automatically an open network connection with the Community-Lab controller, becoming part of the testbed for the experiments. These devices serve as the infrastructure layer of the Community-Lab and most are low-capacity devices which can be affordable to have at the edges of these types of networks.

2.1.4 Sustainability on Community Networks

The growth of community networks have made an impact on how under-served areas could reach the Internet and enjoy network services and applications. Nowadays, these established community networks can expand its usefulness with cloud-based computing services and community-wide services. This can be achieved by having cloud-like service infrastructures like Cloudy OS that can involve home users (home resources) and provide low-latency services over

[11] http://openwrt.org

community networks that can be a challenging environment (e.g. wireless mesh networks).

It has been described in [18] how these resources and services can be organized as a common pool of resources, shared, accessible, and managed by the members of a community. In this way, the community can govern and use its own cloud resources and services, allowing any community member deploy new services using these resources held in commons.

The feasibility for such cloud-like deployments begins with the interest and the incentives given to the community to share resources. As such, motivations for users to utilize cloud resources need to be balanced with the motivation to volunteer offering community resources for community usage, and the perceived added value of the service offer.

2.1.5 Virtualization Systems

Most of the Community network devices that are used in the CN micro-clouds are low-capacity devices such as Home gateways, set-up boxes, research devices. However, these devices are capable of running multiple CN micro-cloud services simultaneously. For instance, the Cloudy OS comes with a few pre-installed services such as Tahoe-LAFS [12] distributed file storage system and PeerStreamer [13]: P2P video streaming framework. As we discussed previously, these devices are configured to deploy CN micro-cloud services bare-metal. We can say that, as an entry-point, virtualization can give us the means to create multi-purpose environments in a single device. Therefore, we can study two main virtualization techniques and measure the system behaviour of each case in terms of performance and complexity. To choose

[12] http://tahoe-lafs.org
[13] http://peerstreamer.org

one of these mechanisms we need to consider the complexity of the platform configurations, system performance and the hardware support that the devices have.

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

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

2.1.6 Gossip-Enabled Networks

Gossip protocols rely on disseminating information by utilizing a small subset of neighboring nodes to pass on data towards the whole network, instead of flooding the network or using a single server. Thus, each neighbor is required to disseminate the messages only to its direct neighbors, forming a directed graph over the current networks to achieve quick and efficient dissemination.

[14] http://openvz.org

The purpose of having gossip overlays over networks is to overcome the issues of node discovery, detection or data dissemination [19]. In addition, gossip-enabled networks can scale with the network, since each node is only required to perform a fixed set of operations for dissemination; the network becomes resilient to node failures, node failure has little impact on the dissemination of data; avoids overloading the network with data, while ensuring all nodes eventually learn about shared information. Moreover, gossiping protocols rely on eventual consistency, where all nodes will have the data within a time-frame. Therefore, an issue on the gossip approach is that not all points of the network have the same information at the same time.

In CNs micro-clouds, the use of gossip overlay is an efficient way for service discovery, publicizing shared services to the network members. Furthermore, users can utilize and announce shared services without relying on discussion forums or "word of mouth" knowledge.

Our case study for gossip-enabled networks is Serf, a system that creates a gossip overlay between different members of a network [20]. Each node has a local agent that sends and receives messages from the other nodes. Each agent publishes its information to the members of the network, e.g. includes the nodes' name, number of members known, events queued to be processed and other tags with custom information. Thus, additional information can be shared between members, apart from the default information from Serf, by using custom tags. Furthermore, each interconnected node through Serf spreads the information to their neighbor nodes (T_{fanout}), 3 nodes by default. The gossip interval (T_{gossip}) to send data is also adjustable as a configuration option, with a default of 0.2 seconds.

2.2 Related Work

In this section we present the most relevant works related to our topic, and bridge them between what has been done before with the optimization to be made on service overlays on micro-clouds. We begin by explaining the cloud concept and how it is transported out of data centers towards the edges of the network. We also explain the monitoring ability and tools that exist in order to understand how we gather information in community network environments. We include an overview of the services that are deployed in such micro-clouds and the social integration of SNs into publish/subscribe systems, demonstrating how we can integrate social properties into existing systems.

2.2.1 Cloud Computing and Edge Cloud Computing

In the Personal clouds proposal [1], the authors enhance the capabilities of mobile devices, seen as low-powered devices, by using either remote or nearby cloud resources, instead of having to process data locally and thus reducing consumption within the mobile devices. Furthermore, this work explains how network resources are to be integrated in an heterogeneous environment, while enhancing the user experience. In this way, each device can be seen as a single device cloud and active participants can run the services which are of interest to the end users. Their primary goal is to take the data processing and storage on mobile devices into nearby cloud resources, thus focusing their work on the mobility aspects for edge cloud computing.

The work reported in [3] describes techniques that may satisfy the offload computation of mobile devices. A comparison is provided to better understand and succeed when doing processing on low-powered devices. This system only accounts for service concurrency and not user independent, thus only

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

The work in [21] shows how clouds that have under-utilized resources can be enhanced by sharing these resources with other communities, while still maintaining the same aspects that the cloud owners have agreed upon. They define formulation for under utilization of resources, by VMs, and construct an analytical model on unreliable VMs in order to understand the optimization on sharing resources with other communities. However their premise on community clouds is based on costumer and service providers pricing agreements for cloud computing.

In the Paradrop system [22] the authors have created a platform in which low powered resources are used, such as home gateways, in order to deploy different services running and processing concurrently and with different data. Trying to get the most use out of such devices, while also taking advantage of the parallelism that devices can create between computations. This system only accounts for service concurrency and not user independency, thus it only considers the virtualization layer for services to run concurrently in a low-powered device.

The area of Fog computing [2] is related to our work in the concept of integrating edge devices. To this end, Fog computing aims to extend data center-based cloud computing by integrating hosts at the edges of the networks into the cloud process. Edges are seen as being proactive components for services, data usage and storage.

The work in [23] demonstrates that container-based system virtualization is performed well over hypervisors by giving the opportunity of isolation in terms of security and resources. Their results show that container-based

system virtualization provides up to 2x the system performance of hypervisors for server-type workloads and scale further while maintaining the system performance at a higher level.

Docker [24] is a modern virtualization option that has succeeded in bringing a lightweight and high performance to computing platforms. Virtualization makes applications able to execute in an isolated way from the host system, while also able to secure them against other parallel applications interference. Moreover, this technology brings us closer to the cloud paradigm without requiring closed software.

Kubernetes [25] is aimed at being the management layer for containers, as an open source platform for automating deployment, scaling, and operations of application containers across clusters of hosts, providing container-centric infrastructure. This system demonstrates the forthcoming options to manage edge and cloud computation that we need to consider when creating our own platforms to handle services on networks such as community networks.

2.2.2 Monitoring tools in Cloud resources

The decentralized monitoring of resources and services have been studied before. Cluster monitoring and management, in the work of [26], is done through an hierarchical overlay network of the available resources. The use of virtual IP system is required to identify each node of the network and to exchange monitor data. In their work, the nodes of each cluster periodically push their information to the master nodes, in this way information is sent hierarchically to the masters of the network in order to control each of the clusters. This work focuses on monitoring resources for management of clusters, and does not account for the actual information shared or the amount exchanged. The dissemination of data is done in an hierarchical manner, between nodes and

masters. Also, the information sharing is done as a push based system in order for the masters to obtain the monitoring data from the nodes.

In the work of [27] monitoring tools are used for workstations in clusters. Information sharing is done through a communication interface between nodes and monitoring proxies. The proxies act as receivers of data from a particular group of nodes, which can be based on resource types or job allocation policies. An important lesson learned is the behaviour of monitoring clusters, which needs to be open environments, flexible and scalable. The behaviour of the current monitoring tools only account for the system they are based on, and may not be flexible enough for handling different loads or resource types. This work uses workstations as clusters, and the inter-connectivity in the infrastructure is intended to be as the Internet infrastructure, which does not account with wireless connectivity and the issues that are added when accounting with wireless infrastructures. The monitoring process is done through the use of agents in the nodes to gather local information and relate to a central monitor to be processed. The central node will then contain the data statistics for the clusters.

Monitoring Large-Scale Cloud Systems with Layered Gossip Protocols [28] presents with an alternative to monitoring services through additional infrastructures in cloud systems. Their work is focused on the gossiping communication for data collection and monitoring on large scale cloud systems, aggregating data from the virtual machines deployed with the services, for a self monitoring process among the grouped virtual machines. The grouping process is done by evaluating each virtual machines primary applications and their location related to each other, e.g all virtual machines running a web server would be grouped together. The grouping of virtual machines helps on the dissemination of monitoring data across the network.

The monitoring in cloud environments is an extension to previous works done with grid and clusters. In the work of [29], they analyse the concepts for monitoring cloud systems, and make reference on the solutions available, the trends and future directions to be taken into consideration. In the concepts on how monitoring systems support the cloud features and requirements they include scalability, elasticity, migration, accuracy, autonomy and comprehensiveness. These concepts are required to be handled in some level, matching the requirements of the cloud systems. Each of the concepts behind monitoring are required to be handled in some level, in order to match the requirements of the cloud systems. This step is done in order to enable the monitoring process handling the requirements of the system it is built for.

In the work of [30] they present a way for monitoring data across nodes of the network, by using gossip protocols. The dissemination occurs in a gossip-based method, in order to analyse the resource information and as well as failure detection of the nodes. The monitoring functions are mostly for networking and resource failures, and do not account for the services themselves. Their work accounts for nodes data consistency and dissemination to other nodes, by using a layered network to interconnect various nodes. Data is gathered through sensors, which are external to the services being monitored, and stored in local persistent state processes, which will then disseminate to other nodes through the use of gossiping protocols. Their work only accounts for wired networks, maintaining the same state of network for each node.

The monitoring solutions can be divided in three categories: generic, cluster and grid, and cloud-specific. The solutions proposed are designed to handle the cloud requirements, however in each case they are specific to each environment or a generic way for dealing with monitoring without accounting for all the requirements. The solutions presented are mostly made for cloud, cluster

or grid environments and do not account for network instability, only node failures. The current monitoring tools available are only capable of handling one of the categories and few of the requirements of the systems, mostly built with specific vision according to a given setup and system. Furthermore, the monitoring solutions presented do not account fully with the service and resource utilization, or do not have a decentralization of monitoring data over wireless mesh networks.

2.2.3 Micro-cloud Services

In [5] the authors report on how the deployment of the cloud model on top of guifi.net was undertaken. They elaborate a system where users can benefit from cloud-based services inside of the network without having to consume them from the Internet. Instead they can utilize the resources that already exist in their community network, while also granting access to cloud-based services. Their work is based in the same infrastructure and environment, however they do not account with social networks or community of practice as the primary source to create an overlay towards optimizing the routing.

In terms of evaluating the performance of PeerStreamer in unreliable networks, the work of Baldesi et al. [31, 32] is the most relevant to our work. The authors evaluate PeerStreamer, a P2P video streaming platform, on the CommunityLab, the wireless community network (WCN) testbed of the EU FIRE project CONFINE. Their experiments highlight the feasibility of P2P video streaming, but they also show that the streaming platform must be tailored ad-hoc for the WCN itself to be able to fully adapt and exploit its features and overcome its limitations. However they evaluated with a limited number of nodes (16 Guifi.net nodes), which were located in the city of Barcelona and they do not use live video stream. A recent PhD dissertation [33] includes some discussion

on P2P streaming on WCNs, but does not elaborate on live streaming, but consider streaming of Video on Demand (VoD) retrieval.

Another work [34] studies different strategies to choose neighbours in a P2P-TV system (PeerStreamer). The authors evaluate PeerStreamer on a cluster and on Planetlab. In wireless networks PULLCAST [35], is a cooperative protocol for multicast systems, where nodes receive video chunks via multicast from a streaming point, and cooperate at the application level, by building a local, lightweight, P2P overlay that supports unicast recovery of chunks not correctly received via multicast.

The impact of uncooperative peers on video discontinuity and latency during live video streaming using PlanetLab is studied in [36]. The paper in [37] investigates the impact of peer bandwidth heterogeneity on the performance of a mesh based P2P system for live streaming.

2.2.4 Enhancement of Overlay Networks with Social information

In other works, the construction of P2P pub/sub systems aims to minimize the number of relay nodes. The proposed approaches can be divided in two main categories: i) the design of a routing tree, the construction of which relies on the routing process of the underlying P2P overlay network [38]; and ii) the construction of a P2P topology such that the paths in the routing tree contains the minimum number of relay nodes [39, 12, 40].

In the first category, Bayeux [38] organized peers into a DHT, where each peer maintains $O(logN)$ connections. Then a routing tree is built for each topic with a rendezvous node at the root, which delivers the events to the peers that join the tree. This approach, however, forces many nodes to relay the messages for which they have not subscribed. Consequently, Bayeux-based

systems suffer from high traffic overhead as they fail to minimize the number of relay nodes.

Rahimian et al. [12, 41] proposed a gossip-based hybrid P2P overlay for pub/sub systems, called Vitis. Peers in Vitis are organized in a ring structure and run a gossip-based peer sampling algorithm to identify the subscription and establish connections so that peers that are interested on similar topics are organized in clusters. Although Vitis manages to reduce the number of relay nodes, peers with high social degree present high traffic overhead since the rest of the peers aim to connect with the social users that maintain the most social friends in common.

Finally, OMen [11] is one of the most recent approaches that emphasizes on the design of the P2P overlay network in order to provide a P2P pub/sub system. OMen [11] incorporated the design of a Topic connected Overlay (TCO) [42], which is an approximation of the GM algorithm [39]. Forming a P2P small world overlay network of [40], each peer in the OMen pub/sub system maintains a *shadow set*, which is a subset of backup peers that maintain the information to repair the TCO when churn occurs. Although OMen provides a fast recovery mechanism, while maintaining low number of relay nodes, no monitoring on the peers' online activity is performed, thus presenting high traffic overhead to the peers that establish connection to peers with extremely low online behavior.

Other works such as [43, 44], can be considered as improvements to the above in their respective fields, however, each still not consider to build the overlay with the social information, only establishing the social strength between peers as the main part for connectivity; or consider utilizing community of practice as the source for social information.

Recent works such as SpiderCast [45] and PolderCast [46] extend the mentioned works however we do not compete directly with them. Furthermore, such works build upon the same frame of reference works, not fully exploiting the underlying social structure for efficient routing, which can result in heavy relay costs or different handling of the social and P2P graphs.

Community of practice are not well documented, in [47] they describe CoP as a vertical evolution of social networks, members share common interests and cooperate with each other for certain goals. Based on their study, such CoP can achieve the same metrics in betweenness, centrality and closeness as other social networks. CoP can be described as a social network

Table 2.1: Comparison between Data center clouds and CN micro-cloud environments in relation to the creation of services.

Data center clouds	Vs.	CN micro-clouds
Central authority		Decentralized, no owner entity
Homogeneous, rack-based servers (high computing)		Heterogeneous, low-capacity devices
Wired Infrastructures		Generally wireless infrastructure
Services are mature, easy access		Services ported from clouds or not designed for CNs environment
High perceived quality, performance		Low perceived quality, performance
Improved traffic to the Internet		To avoid congesting traffic to the Internet
Far away from people (remote)		Closer to the people (local)

2.2.5 Discussion

Data centers clouds and CN micro-cloud environments are built with different properties. Table 2.1 gives a comparison on properties that appear in data center clouds versus CN micro-clouds, where we see different aspects such as homogeneous infrastructure within data center clouds versus an heterogeneous infrastructure within CN micro-clouds. There is clear evidence that the environment created within each cloud approach is different, however a question remains on how services perform (and of their perceived quality) within CN micro-clouds, which is included in the first part of this thesis.

The cloud services presented in the related work, serve as a baseline and representation of different services, which require different aspects from the infrastructure, e.g. Peerstreamer represents services that demand real-time communication and computational use of resources, while Tahoe-LAFS is a representation of services that demand storage, and computational resources. Each service helps to have a broader view of other services that require similar properties within clouds, and be ported to CN micro-clouds.

Moreover, services do have social interactions, formed from either by their usage, content or the relation between people, depending on the type of service. This means that, commonly with state-of-the-art solution, when creating P2P overlay networks, the social interactions are lost. Peers are positioned according to other metrics, or randomly within the overlay network, and links between peers are found to favor high degree nodes, which can create latency and traffic issues for this type of solutions, and can become a hinder to the overlay and underlay network activity.

Why state-of-the-art solutions are not applicable to CNs micro-clouds? The state-of-the-art works center and tailor their solutions towards the data center

infrastructure, therefore do not account with issues, such as latency/traffic variation, services deployed without environment awareness, service communication patterns that come from a tailored network topology based on tree topology. Solutions that apply in edge cloud computing, have assumed a mobility aspect of the resources, such as the case of mobile networks, or do not take into consideration the fact that CNs are built by the community, with limited resources. Therefore, state-of-the-art solutions are not tailored to deploy services within CN environments, and service quality and performance remains as a secondary step towards bringing services into CN micro-clouds. However, in order to motivate the community to favor CN micro-cloud services, we need to account with the perception of service performance and quality that users can have in CN micro-clouds and match to what is expected from data center clouds.

Chapter 3

Resource utilization, monitoring and evaluation

In this chapter we bridge between the optimization of services at the resource level, and analyze resource performance in order to understand the behavior of the resources and be able to act on other levels according to how resources are being used. The separation of levels, in this case, helps to understand the issues that arise from each and make certain that information is not lost between levels, i.e. using the resource information to improve services regarding how resources are used or spread throughout the network can be used in the improvement of service communication. The creation of a multi-purpose environment allows to prepare CN micro-clouds, by creating isolated spaces between owner and community services, and motivate the community to share resources and use services from the CN micro-clouds.

Community networks are crowd-sourced IP networks that evolved into regional-scale computing platforms. This has led to adapting the cloud computing model for services that can operate and use computing resources inside a community network. Resource sharing is done by users, from their own low-capacity devices, such as home gateways, routers, or limited computers. The

resources are fully shared with the community disregarding any organization or their owners needs, where services are deployed bare metal in the devices.

Therefore, the use of shared devices can be turned into multi-purpose execution environments, by applying virtualization techniques in order to address the resource sharing within community networks, and giving owners motivation towards allowing the community to run micro-cloud services.

Our comparative analysis with the current infrastructure in community networks gives evidence about how devices can concurrently run multiple services, the trade offs between the number and resource requirements of services and the degradation of quality that services may suffer.

3.1 Overview

Community networks are large-scale, self-organised and decentralised networking infrastructures built and operated by the community itself. They are open, free and neutral IP networks. The infrastructure is contributed by individuals, companies and organizations in a joint effort.

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

The environments used in this work have restrictions in using computing resources for private purposes (in Community-Lab) or for the deployment of generic services (in Clommunity). We address this limitation by designing

an environment in each device that can benefit both the community and the device owner. Consequently, this produces a multi-purpose environment in a single device, such as one environment for the device owner and another environments shared with the community network. With the support of machine or operating system virtualization, these environments can achieve the requited security or performance objectives.

The main contribution of this chapter include:

- The design of container-based resource virtualization on top of low-power devices, enabling a multi-purpose environment isolated from each other. Users can share a portion of the device resources, isolating the portion allocated to the owner from the portion for third-party services.
- Evaluation of the performance of devices when running services on the virtual environments, and comparison with the current Community-Lab devices.
- Evaluation of the quality degradation on services in our proposed approach with an heterogeneous environment, when increasing the number of services running concurrently.

For validation we create a small-scale physical Community-Lab infrastructure using several low-power devices together as computing and storage devices and its own Community-Lab testbed controller. In the experimental system we deploy services such as file storage (Tahoe-LAFS) [8, 48], video streaming (PeerStreamer) [32, 49] and IoT (Thingspeak)[1], as micro-cloud services. These services serve as an entry point to measure their performance when running in an environment according to our design. This way, we can gather knowledge

[1] http://thingspeak.com

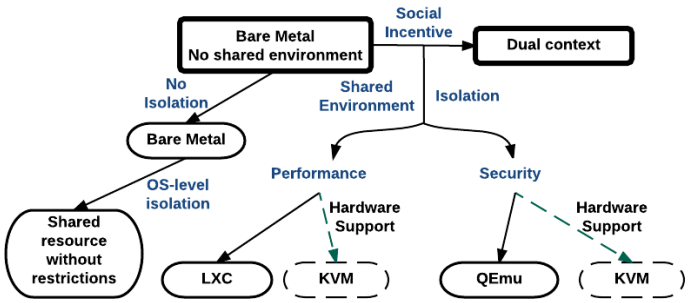


Figure 3.1: Models for execution environments.

on the quality of service when running services from both owner and community networks simultaneously and in an heterogeneous infrastructure. These services represent typical network, processing or storage demanding services of community networks.

3.2 System Architecture

The architecture for our proposed system explores two approaches for virtualization. In the first approach we used *virtual machines* (QEMU) to optimize separation of the context in which every service is deployed. In the second approach we used *virtual operating systems* (LXC containers) a lightweight virtualization to optimize performance while running services.

3.2.1 Models for service deployment

The models for execution environments are an approach to have shared resources between contexts, which achieves a multi-purpose environment. As a social incentive to share resources and still being able to use ones' own resources, the approaches demonstrated in Fig. 3.1 result from properties that can be dealt with virtualization technology, such as isolation, performance

or security. Each approach can also depend on the infrastructure that is available, with newer resources or the intent that the services are given for. The creation of different contexts for individuals and community within the devices can result in an incentive to sustain the community networks clouds without losing perspective on the users properties that are fundamental to the usage of micro-cloud environments.

The proposed approaches, leveraging virtualization resources, are enhanced with different contexts where owners can share (shared context) only part of the device resources while also giving access to their own (private) context, where owners can run their own independent services, as demonstrated in Fig. 3.2.

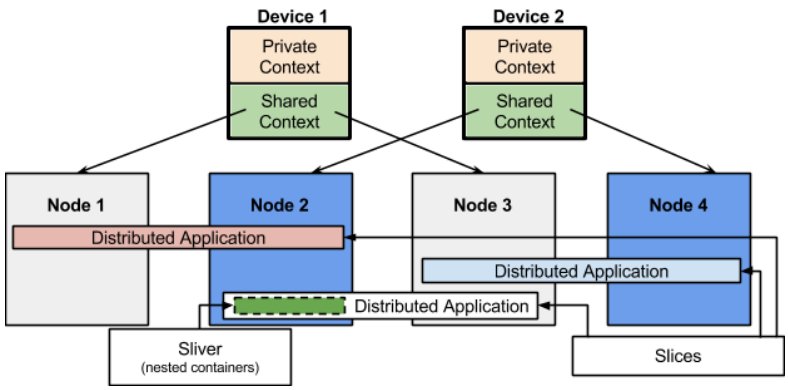


Figure 3.2: Example of two device deployment. Devices are divided in two contexts for owner and shared to the CN. In shared context several slivers run as nested containers belonging to each slice deployed.

3.2.2 Virtual machine deployment

Our initial approach consists of the deployment of virtual machines, using QEMU, installed on top of the Cloudy OS. This allows us to concurrently run services while still isolating the device to be used by its owner. In a physical

device we install Cloudy OS where we can execute several instances of QEMU creating multiple virtual environments, one for each service instance. As an example, Fig. 3.3 shows this scenario using two Community-Lab nodes and its slivers as independent services on top of one instance of Cloudy OS. The figure demonstrates the ability to have multiple environments on one device, therefore one of the virtualized Community-Lab nodes can be considered as the owners' context. The virtualization context is done in order to gain control over the device and enable multiple services running from different user contexts while maintaining the owners' ability to take advantage of its own resources.

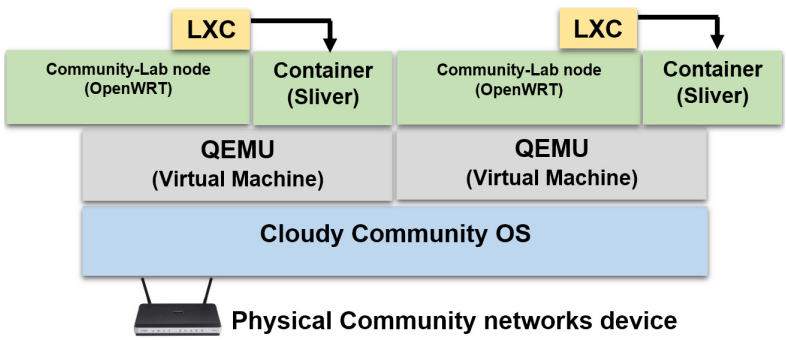


Figure 3.3: Example of the VM deployment approach. Two services running (serving as Community-Lab nodes) on QEMU virtual machines. LXC runs from within the Community-Lab node to create slivers.

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

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

3.2.3 Virtual operating system (containers) deployment

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

In this approach we consider the isolation of the services while guaranteeing a higher performance for services since there is no hardware emulation involved. The virtualization layer considered is in the OS-Level, where the host kernel is shared between containers and the host system. In Fig. 3.4 we show an example of the deployment of this approach using the Cloudy OS as the host system and deployment of a Community-Lab node in a LXC container. It is worth mentioning that the Community-Lab nodes deploy their own slivers as LXC containers and with our approach this does not change. This is provided by tuning the configurations of LXC to deploy nested containers, while adding the AppArmor [2] policies for security concerns.

With this approach, we can use the device as a shared environment between the owner and the micro-cloud, maintaining the isolation from each user context.

[2] http://wiki.apparmor.net

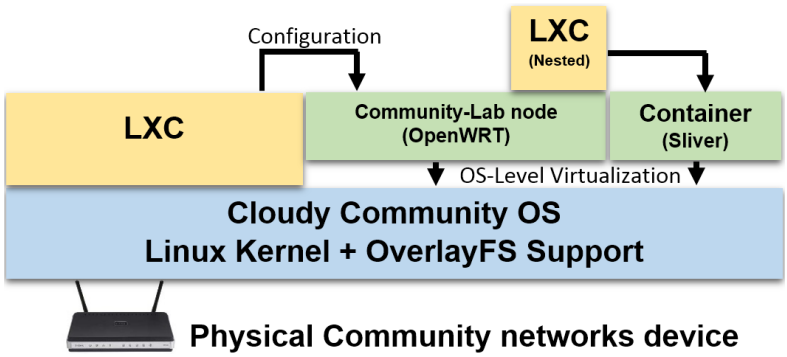


Figure 3.4: Example of LXC deployment approach. One service running (serving as Community-Lab node) on a OS-level container virtualization. Nested LXC runs in the Community-Lab.

The security issues that arise from such usage are respectfully handled by LXC or the Linux security policies. Therefore the containers have access to the host kernel. If this way is not acceptable for some services, the alternative is the virtual machines approach as it provides better isolation.

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

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

A main feature of our proposed deployment scenario is the introduction of a virtual environment in which services are able to run with different contexts

and maintaining an isolated part of the device to be used by the owner. As a result when using the Community-Lab node as a service each node registered in the Community-Lab controller can deploy its own slivers in the physical device without interfering with other services or the usage by the owner. This enables sharing of resources for a micro-cloud environment while maintaining the owner's exclusive access to the device.

3.3 Experimental Setup

For our experimentation setup, we used four physical research devices (computing devices in Community-Lab) with different configurations. These devices are built with Intel powered Atom N2600 CPU processors. Two of them have 2 GB of RAM, 60 GB storage, another has 2 GB of RAM, 120 GB storage, and the last device has 4 GB of RAM with 500 GB of storage disk, and all are linked to the community network (Guifi.net). A desktop computer was used to deploy a local Community-Lab controller in a virtual machine environment. It was necessary to use such deployment in order to not affect the performance of the production Community-Lab infrastructure. We also included a newer device called Minix that is built with Intel Z3735F CPU (with hardware-enabled virtualization), and 32GB of storage. This is the typical device deployed in guifi.net by the Clommunity project with Cloudy OS.

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

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

In our experiments we make use of services such as PeerStreamer, Tahoe-LAFS and Thingspeak, which are processing, storage and network intensive types of services. This allows us to test a realistic use of the infrastructure in our experiments. The experimental setup was created similarly to the current community network cloud environment, therefore we are able to directly compare the performance resulting from the new approach.

3.4 Evaluation

In the evaluation of the proposed approach (LXC deployment approach) we augment our findings with several scenarios. Each scenario is designed to gather knowledge about the feasibility, the environment progression with heterogeneous devices and complex network infrastructure. These devices as explained in section 3.3, have less resources compared to Desktop PCs, and similar to the Community-Lab infrastructure, which users have disposed at home to share with the community network.

Table 3.1: Summary of evaluation scenarios and settings

Scenario / Service	**# Devices**	**# Slivers**	**Metrics**
1 - Streaming service (PeerStreamer)	4 RD	8	Chunks Received, Play-out
2 - File service (Tahoe)	4 RD	8	Storage Benchmark
3 - Tahoe and PeerStreamer	4 RD	16	Storage Benchmark, Chunks Received, Play-out
4 - PeerStreamer	4 RD	8 (*2, *3, *4)	Average Quality Loss
5 - Tahoe	4 RD	8 (*2, *3, *4)	Storage Benchmark
6 - Tahoe and PeerStreamer (minix)	4 RD / 1 Minix	10 (*4)	Average Quality Loss, Storage Benchmark
7 - IoT service Tahoe and PeerStreamer	4 RD / 1 Minix	10 (*2)	Average Quality Loss, Storage Benchmark
8 - Device Performance	1 RD / 1 Minix	1	Processing Time

3.4.1 Experiments

Our experiments were performed to evaluate the proposed deployment scenarios, summarized in Table 3.1, by using different combinations of services with different purposes and with heterogeneous devices. Also, each device supports two (Community-Lab) nodes, in order to account for owner services and community services running concurrently. In this way our experiments can validate our proposed deployment as being a multi-purpose execution environment, running owner's services and community services separately. The following describes each scenario in detail.

- The first evaluation scenario uses PeerStreamer, a peer-to-peer video streaming application, to evaluate the impact on the services that have time sensitive data processing.

- The second scenario uses Tahoe-LAFS distributed storage, and a storage benchmark application to evaluate the impact of the proposed deployment on the physical devices. These first two scenarios establish a baseline on the feasibility and performance of the services involved.

- The third evaluation scenario combines both services and allows an evaluation of the concurrency of services within the same physical devices with our proposed approach.

- The fourth and fifth evaluation scenarios are done to know how the environment progresses when adding more services of the same type using the proposed deployment. Each experiment is run with either 2, 3 or 4 services concurrently (each service has 8 slivers across the devices used).

- The sixth evaluation scenario adds an heterogeneous infrastructure, by adding a different type of device (Minix) and the network interference. This provides us with not only information about how these newer devices can perform, but also allows to compare the performance of the services in an heterogeneous environment as Community networks.

- The seventh evaluation scenario is done with the addition of an IoT-based service (ThingSpeak which is a data collector, analyser for sensor data) to account for a combination of user services and community services.

- The last scenario is used to understand how different types of virtualization impact the processing time of services. In this last scenario an

encryption job is performed in the same way and its processing time is recorded in each of the environments tested.

For each scenario we collected results and plotted them against the baseline values. The baseline values were obtained by running the same set of experiments on the Community-Lab infrastructure (from earlier works in our research group [4, 50]) under the same set of configurations. This gives us the behaviour of the proposed system in terms of performance and user experience against what is currently done.

In the Tahoe-LAFS experiments we measured the performance for read and write operations in order to understand the impact these type of operations have on the proposed deployment. In the PeerStreamer experiments we measured the average chunk rates (data that is received in the peers side); average chunks played out on peers (data that is sent to be watched in the peers side) in order to measure the quality of the video stream; and the average quality loss which is the percentage of chunks that did not arrived from the overall number of chunks sent by the source. In the first scenarios we measured the CPU utilization to demonstrate that these low-power devices can deliver enough performance in a multi-purpose execution environment while maintaining the multi-service community cloud model.

First Scenario: In our first evaluation scenario, we used one sliver deployed per node, running concurrently a Cloudy OS template. In each of these slivers, we ran a PeerStreamer experiment.

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

network to exchange data (known as chunks) between its peers. We also setup one sliver to serve as the source peer which disseminates the source video partitioned in chunks (as default, one frame of the video is one chunk) to be played out by the other peers.

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

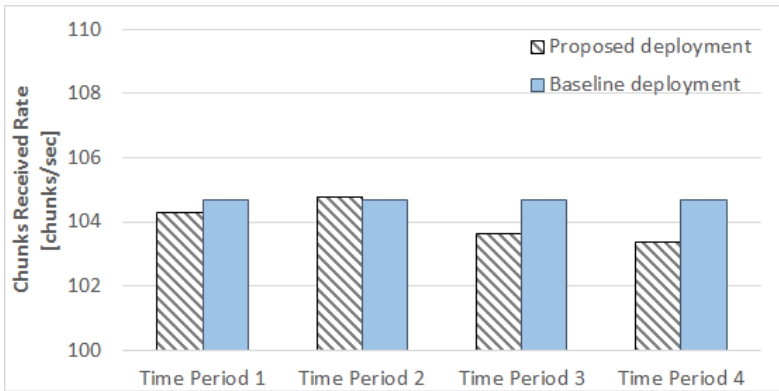


Figure 3.5: Average chunks received rate at peers from PeerStreamer execution in the second evaluation scenario. Baseline as the current deployment in Community-Lab.

Results: Fig. 3.5 and Fig. 3.6 depict the measurements of the average of chunks that a peer can receive and the percentage of chunks that were sent to be played out, averaged by all peers. As a baseline we have the current deployment from previous experiments with the Community-Lab testbed in which the proposed deployment is able to reach without losing too much data on average. It is also noted that because of the shared resources there is a minimum amount of chunks that are not received in the proposed deployment. These time sensitive applications require that data should arrive on time to

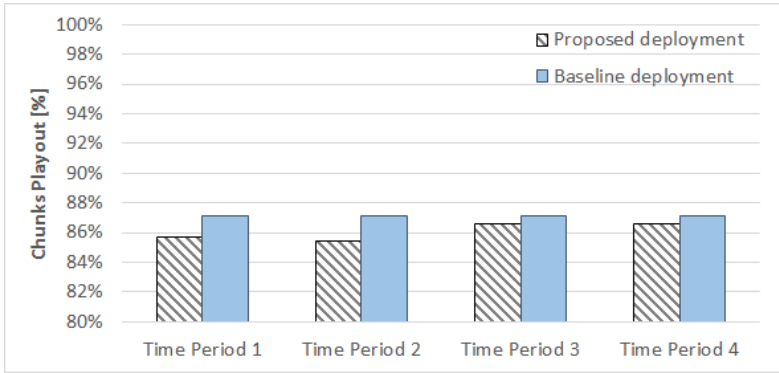


Figure 3.6: Average play-out ratio at peers from PeerStreamer execution in the second evaluation scenario. Baseline as the current deployment in Community-Lab.

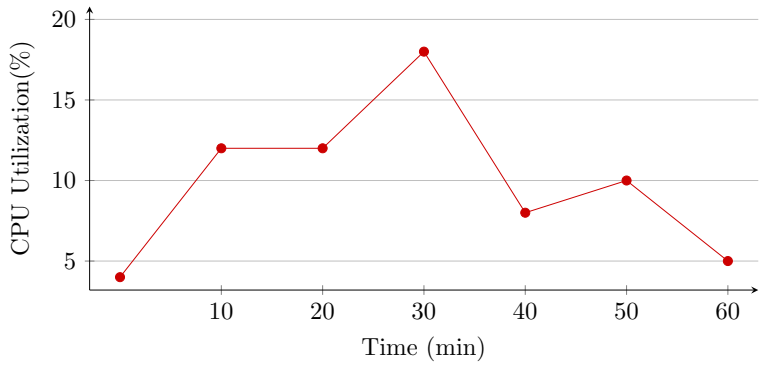


Figure 3.7: CPU utilization on PeerStreamer (PS) source node on second evaluation scenario

be displayed/processed, if the data does not appear in the time allotted it is discarded. In the proposed deployment this amount does not vary much from the baseline (2% on average).

Fig. 3.7 shows the average measurements of CPU utilization during an hour, when the service runs continuously. The alterations we see in the CPU utilization is in fact because the PeerStreamer neighbourhood size changes

over time (the overlay network is constantly updated even when there are no new peers) and therefore the utilization of the resources change when the operations of reorganising the topology of the network are performed. Moreover the CPU utilization on the source node is higher than on the peer nodes since it transcodes the video stream and partitions it into chunks that will be sent to the peers. Thus the CPU utilization on the peers is considerably lower on average and does not interfere as much with other services running concurrently. This is a result of the peers running a more lightweight process such as gathering and decoding of chunks.

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

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

Furthermore we ran a well-established disk benchmark application, i.e. IO-Zone [51], to measure the storage operations of each node in order to evaluate

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

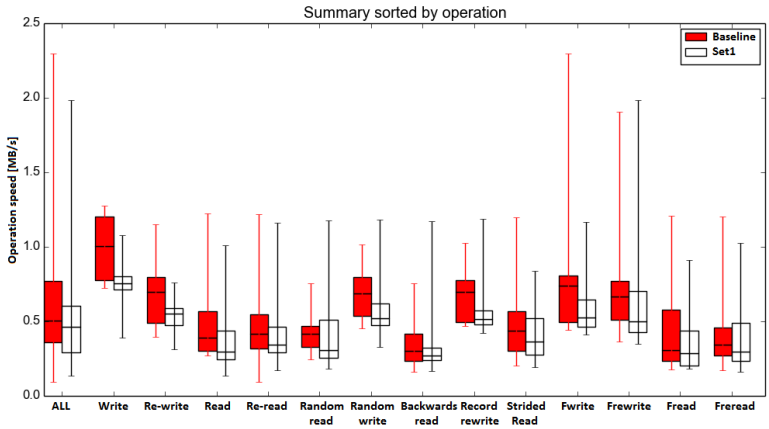


Figure 3.8: Performance of Tahoe-LAFS service, baseline as the current deployment and set1 as within the proposed deployment on the first evaluation scenario (operations shown as average All, write, re-write, read, re-read, among others). Representing average, and deviations for each operation.

the proposed deployment and compare it with the performance of the same services when using the current Community-Lab infrastructure. The benchmark uses the Tahoe-LAFS system as a disk, writing and reading file happens across all storage nodes, making all nodes working together for the given task and therefore the performance taken on the client includes the performance of the storage nodes.

Results: Fig. 3.8 shows the measurements of the baseline system corresponding to the current Community-Lab environment, and the same set of operations on the proposed deployment (named set1). Note that all operations are completed when the transactions between instances are finished therefore the results account with the performance of all Community-Lab nodes used. In the proposed deployment there are two services running on the same physical devices therefore each Tahoe-LAFS system has concurrent access to the resources which is, as expected, reflected by a general lower operation speed

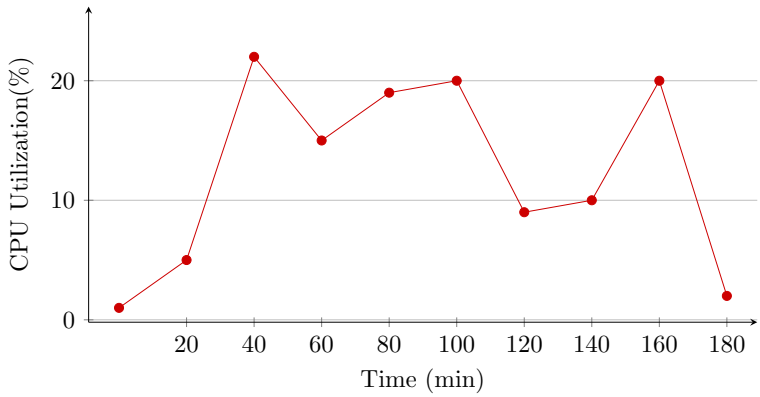


Figure 3.9: CPU utilization on Tahoe-LAFS client node in the first evaluation scenario

for all the operations measured. However this still accomplishes the operations with the approximated speeds as the baseline evaluation.

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

Fig. 3.9 shows the average CPU utilization during three hours of running the IOZone benchmarking application in the Tahoe-LAFS client instance. It has consumed around 20% of CPU time on average during the complete test. The client node performs encryption/decryption and erasure code computation on each file block in the phase of writing and/or reading to/from the disk. Furthermore the Tahoe-LAFS client instance consumes more CPU time as opposed to the Tahoe-LAFS storage nodes.

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

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

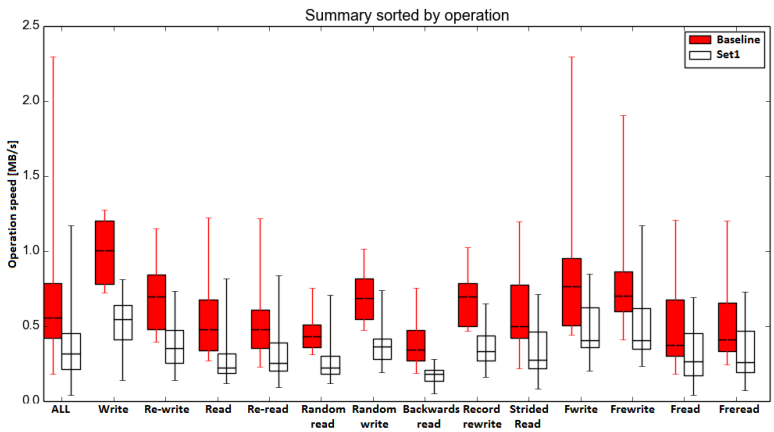


Figure 3.10: Performance of Tahoe-LAFS service, baseline as the current deployment and set1 as within the proposed deployment in third evaluation scenario (operations shown as Average All, write, re-write, read, re-read, among others). Representing average, and deviations for each operation.

Results: Fig. 3.10 shows the results from the current deployment in CommunityLab as baseline, against the proposed deployment in Set1. This shows that when different concurrent services are running on the same device, the impact on the services are noticeable, with higher loads on the system. This can be

attenuated with service scheduling or with a more social understanding of the services usage to avoid very high peaks on devices.

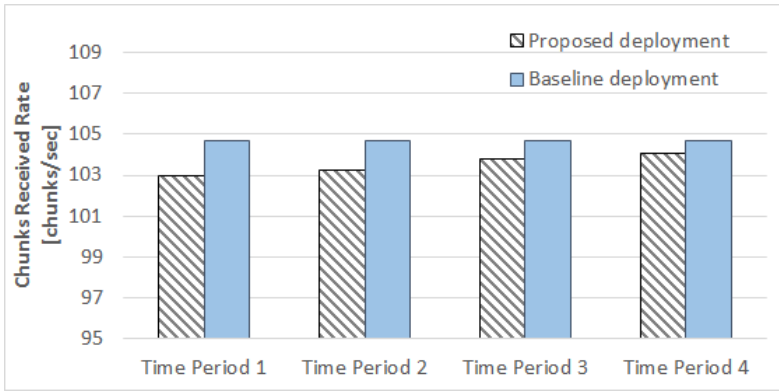


Figure 3.11: Average chunks received rate at peers when Tahoe-Lafs is also running (third evaluation scenario). Baseline as the current deployment in Community-Lab.

Fig. 3.11 and Fig. 3.12 show the results of the current deployment on the Community-Lab as a baseline against the proposed deployment results. We can see a noticeable, to a certain degree, variation of the chunks played out which affects the video quality perceived. This is due to the CPU time being shared among more processes. However, while concurrent services may differ, for our results we can say that the loss is minimal when using the proposed deployment against the current Community-Lab deployment.

The average CPU utilization of the Tahoe-LAFS client instance is shown in Fig. 3.13, demonstrating that the CPU utilization of the Tahoe-LAFS client instance is around 35% - 40% on average over the course of the experiment. This also means that it uses at most half of one core of the device. This is a result of the added processing in the client instances while the storage instances have a lower process utilization, performing only read and write operations.

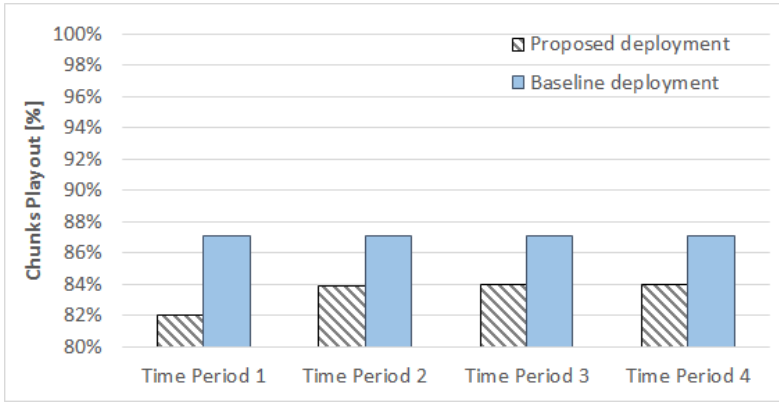


Figure 3.12: Average chunk playout at peers when Tahoe-LAFS is also running (third evaluation scenario). Baseline as the current deployment in CommunityLab.

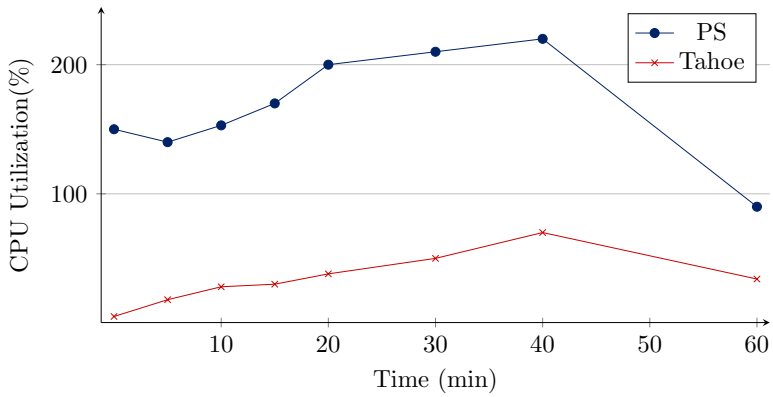


Figure 3.13: Average CPU utilization of Tahoe-LAFS and PeerStreamer (PS) in third evaluation scenario

On the other hand, the average CPU consumption for PeerStreamer is nearly five times higher than what Tahoe-LAFS uses. This behaviour is accounted by the fact that while running the service, we also saved the video to the disk (in all the peers, including the source), therefore driving more utilization of the resource for that service. The CPU utilization on the peers remains lower

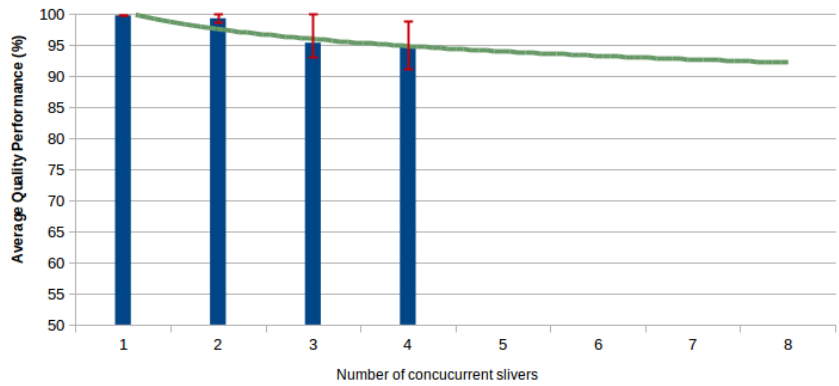


Figure 3.14: Performance degradation evaluation of RD devices when running multiple Peerstreamer service.

than in the source peer and still feasible to be running concurrently with other services.

Fourth Scenario: In this scenario we ran the PeerStreamer service adding instances of the service for each time frame. Each service runs concurrently in the nodes available, totalling 32 slivers running with its own intent. Each test runs as the first scenario, in time periods of 1 hour, seven slivers as peers and one sliver as source that disseminates the video to its peers. Each addition of a service means a creation of a sliver within each node and therefore isolated from each other. The scenario is done is this way, in order to establish a correspondence between increasing the same type of services, and the resources available with our proposed approach.

Results: Fig. 3.14 shows the progression of the quality degradation that services suffer, on average, when increasing the number of services concurrently running on our proposed approach. As demonstrated there is a loss of quality in each service, when adding this type of services, however the quality loss

is compensated by the higher number of services that can be run isolated, diminishing 1 to 2% of quality each service added, also constraining the device with more network activity. For a higher number of concurrent services the quality loss is expected to drop drastically since the network and the device will be overused. Thus, a concrete number of services that can run concurrently is an open issue that should be addressed, that include the network and the device usage along time.

Fifth Scenario: In this scenario we ran the Tahoe-LAFS service adding instances of the service for each time frame. Like in the fourth scenario, each service runs concurrently in the nodes available, totalling 32 slivers running with its own intent. Each test runs as the second scenario, in time periods of 3 hour, seven slivers as Tahoe storage nodes, one sliver as the Tahoe-LAFS Introducer and one sliver as the Tahoe client that stores files into the established service. Each addition of a service means a creation of a sliver within each node and therefore isolated from each other. Also, each client (from each service running) uses IOZone to measure the reads and writes response of each service in order to gather the necessary information about the performance of each service, and therefore establish results on the performance of this type of services when increasing the number of concurrent services.

Results: Fig. 3.15 shows the average operations speeds of this type of service when increasing the number of concurrent services. As demonstrated the operation speed decrease less than 1% for each service added. This is because the devices use memory cards to store data instead of the conventional disks, therefore the operation speed depend more on the transfer of data between nodes. This type of service can be more reliable when activity increases, since the processing on the storage nodes is minimal compared with the client. Also, with a higher number of concurrent services it is expected that the operation

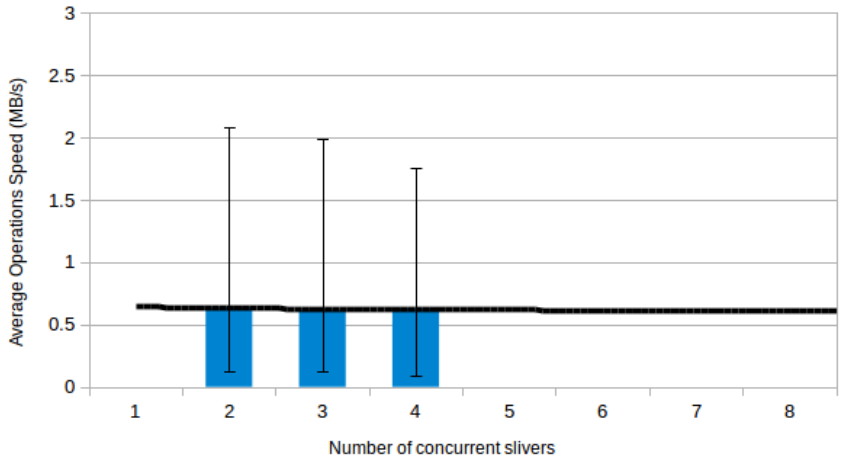


Figure 3.15: Performance degradation evaluation of RD devices when running multiple Tahoe service.

speed for all services would decline rapidly, since the network will become saturated with requests. Further study on this issue would be required.

Sixth Scenario: This scenario serves as an argument to our findings in order to assess our proposed approach with more realistic infrastructure and with other devices that can be deployed for community network clouds. Therefore, we added a new device with different specifications (Minix) in order to account for an heterogeneous infrastructure. We also acknowledge the network activity in these tests as a more realistic setup. Therefore we use the new device with our proposed approach and compare with current Community-Lab infrastructure and the evaluation previously done. In this scenario we ran four concurrent services, with either PeerStreamer or Tahoe-LAFS as a service, within different time periods of 1 hour each and with the same setup as the previous scenarios.

Results: Fig. 3.16 represents the average quality loss of the service when adding the new device and the network normal activity. We observe that

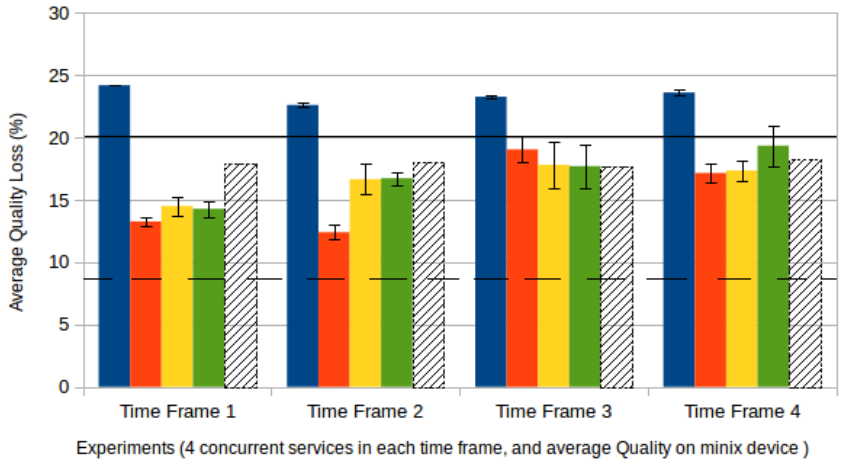


Figure 3.16: Quality Loss evaluation, with Minix device. Line represents quality loss on Community networks, dotted line represents quality loss on research devices.

PeerStreamer suffers quality loss when running four services concurrently and in an heterogeneous environment. We can also notice that the first service in each time frame has more degradation due to the source being farther away (in the network) than on the other services. The figure also shows a constant loss across all experiments because of the network involvement; from previous work [50] the loss happens with less than 20% when network is involved; and from our previous experiments the loss is around 7%. Also, the figure shows, with the last bar on each time frame, that the new device performs on average equally across all experiments. This means that having an heterogeneous environment with newer devices for this type of service only serves to increase the number of services that can concurrently run. We also notice that the different geo-location of each source affects the quality of service, degrading the quality since the nodes will be further away in the network.

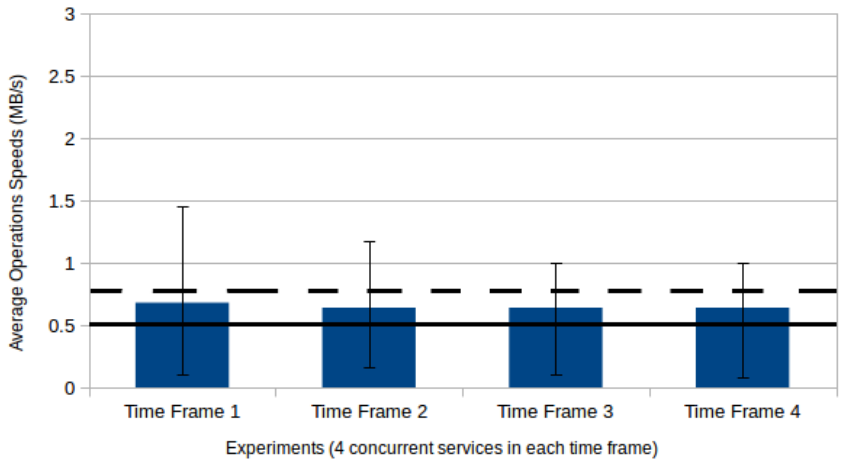


Figure 3.17: Average Operations speed evaluation, with Minix device. Line represents operations speed on Community networks, dotted line represents operations speed on research devices.

Fig. 3.17 shows the average operations speeds of four Tahoe-LAFS services running concurrently in different time frames on an heterogeneous environment and with normal network activity. The figure shows that the operation speed results on average in less performance than our previous experiments (around 0.6 MB/s) due to the network activity, and more performance than the average from previous work [4], which was done with current Community-Lab infrastructure. As with the previous experiments, the operation speeds on average only suffers with network activity and higher demand on storage operations. Therefore, having an heterogeneous environment with low-power devices for this type of service can be advantageous to still share with the community network, while having a owner private environment.

Seventh Scenario: In this scenario we augment the services by adding an IoT-based service as a user service close to their homes. ThingSpeak[4] is a data collector and analyzer for sensors and IoT devices. This service added to the scenarios already studied can enhance our knowledge about different types of services working concurrently and isolated, and the interference each service can have on our proposed approach. We setup two services each from PeerStreamer and Tahoe-LAFS, with the same properties as the other scenarios and the same time periods as before; and we create a ThingSpeak server (the server gathers all the data, analyses and creates real-time graphics of the data) for each sliver (ten slivers in total from each node available) that runs concurrently with other services. For each test, we send data to each ThingSpeak server every 30-45 seconds during the whole time frame that the other services run. In this way the ThingSpeak service is constantly being used to update the current data (this data can be like temperature, noise, wind conditions, among others).

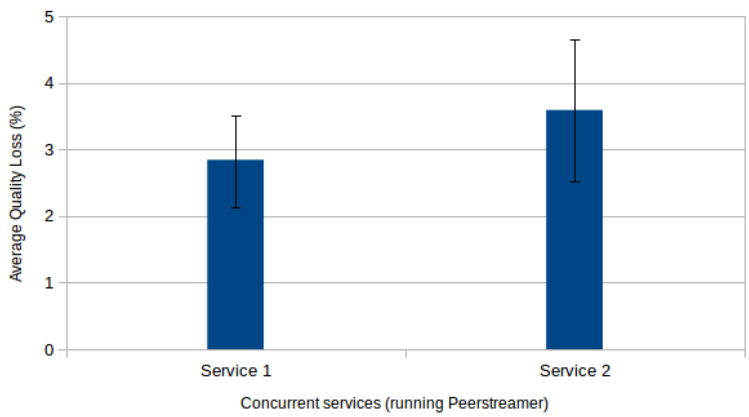


Figure 3.18: Average Quality loss of the two concurrent PeerStreamer services, when other services are running concurrently (Tahoe-LAFS and Thingspeak)

[4]http://thingspeak.com

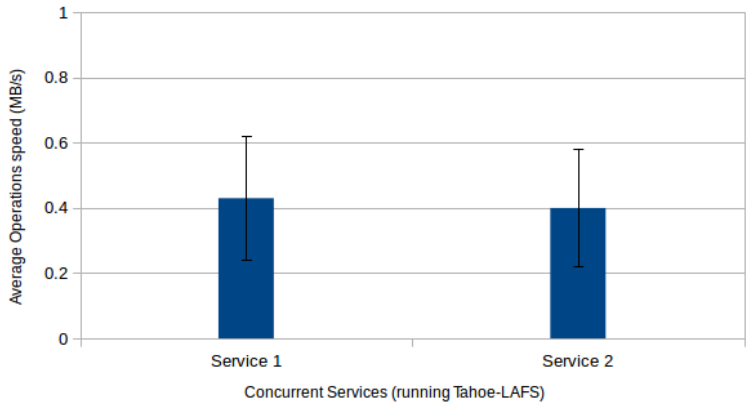


Figure 3.19: Average operations speeds of the two concurrent Tahoe-LAFS services, when other services are running concurrently (Peerstreamer and Thingspeak)

Results: Fig. 3.18 shows the average quality loss for two PeerStreamer services running concurrently with Tahoe-LAFS and Thingspeak. As we can see the loss is under 4%, meaning that the other services can influence the degradation that happens, however still the same it has with previous experiments. Fig. 3.19 shows the average operation speed for the two Tahoe-LAFS services, within the same values as previous experiments, however can be with less performance since the IoT-based service demands more from the data storage. The figures shows that running many services concurrently and isolated with our proposed approach can have beneficial results in terms of increased activity, with lower quality loss. However the environment has a threshold where it may not support an increase in services without losing too much of its quality.

Eighth Scenario: The last scenario is done to expand our findings in order to understand what are the impacts that different types of virtualization can have on the processing environments, such as our proposed approach. Therefore, we used one of our research device and the Minix device, with LXC

approach or the QEMU approach explained in section 3.2. We then ran an encryption application sixty times in bare metal as a baseline for the processing time and do the same encryption in LXC and QEMU environments to obtain the processing times for each environment. With this we can understand how each virtualization interferes with the services that can run in each device and with our approaches.

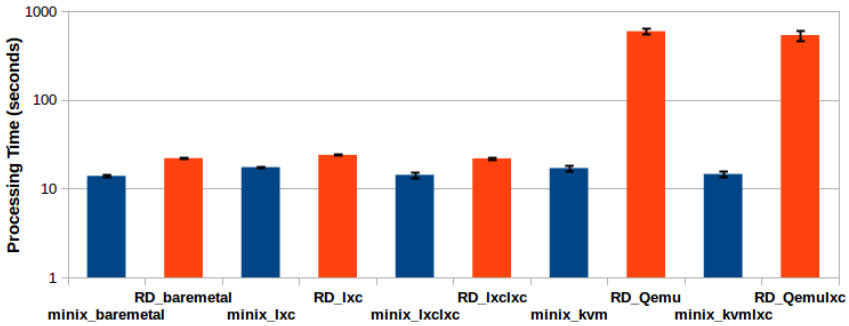


Figure 3.20: Performance evaluation of devices Minix and RD, using LXC or QEmu.

Results: Fig. 3.20 shows the processing times with each type of device used (RD or Minix) and with each type of virtualization (LXC or QEmu and LXC nested). It is shown that the performance between using LXC and without virtualization (bare metal) is mostly the same, with 1 to 3 seconds for context exchange, and that the QEMU without virtualization support has very low performance, while in the Minix device with virtualization support (KVM) it achieves mostly the same performance as running in bare metal. The time difference between each experiment gives us motivation to accept that our proposed approach can work with little difference between each service while running isolated and concurrently.

3.5 Discussion

In the proposed approach we see an impact that affects the performance of the services, which is minimal compared to running more concurrent services within a physical device. We also state that both approaches (virtual machines and containers) can be used according to the resources available. Virtual machines should be more suitable for services that demand higher isolation from the host system or with higher security concerns. Containers should be chosen when having lower-capacity devices or to achieve higher performance, and still maintain some isolation from each service. The main problem still remains on who has the rights to dictate the best trade offs between security and performance. In overall, the users may want both performance and security when running their services externally; while the owners would only need security between services, and the highest performance of their own services. The community may prefer higher performance over security, increasing the amount of computing power available to all, while security concerns are addressed by common secured protocols.

In our experiments the resource utilization was key to understand the impact and how we can optimize the shared resources. Further study with simulation or even high scale deployment will narrow down other issues that can arise, such as the maximum number of services, how many services per user one device should handle or even how much of the device should be given to the owners and the community. These issues are important to understand and to have a better utilization of the resources spread on community networks, and should be addressed with further social studies.

Furthermore each type of service explored (being network, processing or storage bound) has its own characteristics, and impact on other services in specific

ways. Adding scheduling or resource allocation control mechanisms across all devices available can minimize this impact.

Cloud-like services can be deployed on the edges of the network with the added effect of motivating the owners to share only part of their resources.

Moreover we can say that using such deployments we can guarantee isolation of services while maintaining mostly the same quality of service. Therefore, our approach shows to be feasible for the current deployment of devices on community networks with minimal performance loss in the services, and that the threshold for the increase of services on the edge of network can be large enough to accommodate most demands of community network services.

In our experiments the failure of nodes and network in not mentioned as a problem in the scenarios. Although an important issue for the services, each service deals with failures in its own way (i.e. in a P2P fashion). The Community-Lab nodes do reconnect to the network and resume operating. Therefore such issues can occur on the scenarios without long lasting effects on performance, or with graceful degradation in a service specific way i.e. if the source node in case of Peerstreamer fails, is equivalent to the video stopping being streamed, thus not accounting as a failure.

3.6 Conclusion

We argue that community networks are underutilized if only Internet access is targeted. The resources in the network can actually support more services, which can give community members advantages and benefits when participating in shared services.

The container-based lightweight virtualization approach proposed in this work for contributed low-capacity devices, maintains the benefits of running

concurrent services on these low-power devices and delivers a multi-purpose system. Also it features isolation of both resources and security whereby guaranteeing a better overall system performance even on resource-constrained devices. The work also addressed the current limitations of micro-cloud infrastructures such as the scalability of the existing infrastructure, and the performance of low-power devices.

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

Moreover, in our evaluation we demonstrated the advantages that our proposed approach can have on the quality performance of services, when increasing their concurrent running usage on these devices. We show that there is a balance between having multiple services and the quality loss, which depends on the type of service in use (network, processing or storage intensive), which can still function as a cloud-like infrastructure with limits on resource consumption while granting resources to the owners.

Furthermore, the proposed approach can augment the established network infrastructure of community networks and the ability for users to have access to more computing power, exploring the micro-cloud environments and achieving a multi-purpose system. This way we can add value to the community networks and lower the impact on the services when running concurrently within the community network infrastructure. However, cloud services can become disruptive in micro-cloud environments even when the infrastructure

is prepared to handle multiple services. Can the same cloud services work out-of-the-box in micro-cloud environments, or do resources affect their behavior?

Chapter 4

Services performance and optimization in CN Micro-clouds

In this chapter we address the issues raised within the service level were we give an overview of monitoring tools in order to understand how services behave in CN micro-clouds, and analyze service performance under CN environments. The analysis of services is an important step to understand within the service level how services can be adapted in order to enhance services, QoS and users' perceived QoE.

Section 4.1 describes an approach to monitoring in CN micro-clouds, and general wireless edge clouds, using gossip technique that offers the possibility to disseminate and gather information of the services among all the devices of the micro-clouds. In addition, this work is central to obtain the information and behavior of resources, services and users in order to optimize the services configuration and management.

Section 4.2 describes the analysis done on a particular service, such as live video streaming. The analysis gives motivation to the different configurations that a service can have under CN micro-clouds, and obtain advantages over such networks. The modifications are done on the service layer, without requiring

other layers to be altered. Therefore, services can be made to withstand the different capabilities of micro-clouds.

4.1 Service Monitoring in CN Micro-Clouds

Edge cloud computing proposes to support shared services, by using the infrastructure at the network's edge. An important problem is the monitoring and management of services across the edge environment. Therefore, dissemination and gathering of data is not straightforward, differing from the data center cloud infrastructure. We consider the environment of community networks for edge cloud computing, in which the monitoring of cloud services is required. We propose a monitoring platform to collect near real-time data about the services offered in the community network using a gossip-enabled network. We analyze and apply this gossip-enabled network to perform service discovery and information sharing, enabling data dissemination among the community. We implemented our solution as a prototype and used it for collecting service monitoring data from the real operational community network cloud, as a feasible deployment of our solution. By means of emulation and simulation we analyze in different scenarios, the behavior of the gossip overlay solution, and obtain average results regarding information propagation and consistency needs, i.e. in high latency situations, data convergence occurs within minutes.

4.1.1 Overview

The edge environment we introduce in this section is based on community network (CN) environments. Participants in CNs can also share local cloud computing resources and provide local cloud services. In this way, the community creates its own edge cloud computing environment without relying on

data center clouds from outside the network. In our case, Cloudy[1] is used to manage the edge cloud computing and services such as Peerstreamer [32, 49], an open source P2P Media streaming, or Tahoe-LAFS [4], an open source decentralized cloud storage system.

Current solutions for monitoring services under data center cloud systems support and are tailored towards the use of data centers, which disregards the unique properties that an edge cloud environment has, such as the high latency between nodes, changes in network (nodes churn rate), and most importantly the use of low-capacity devices.

Furthermore, an issue found in the edge cloud computing (such as community network clouds) is the lack of a suitable mechanism for logging, monitoring of resources and services, and dissemination of information. Thus, this work is intended to bring the best features of monitoring services from data center cloud environments, towards the edge cloud computing. Also, by understanding the properties of community edge environments, we can tailor monitoring service to better suit the community requirements. And in part, granting knowledge of the cloud infrastructure to the community.

We developed the monitoring platform for Cloudy, in order to give community network clouds an efficient monitoring service. The platform considers the way to handle the dissemination of data, by using a gossip overlay to inter-communicate with the nodes. Also, the platform gathers distributed data, by using the gossip-enabled network through which it disseminates data. The gossiping properties are aligned towards its use on edge cloud computing, since it provides eventual consistency of the data without relying on a single entity, and can still be used when node churn is evident. Other methods, such as

[1] http://cloudy.community

flooding, direct communication does not deal gracefully with the issues that arise from these types of environments.

The main goal of our work is to bring an efficient way of monitoring services on wireless community network clouds, as a study case of edge cloud computing, using gossip-enabled networks to achieve efficient data dissemination and sharing.

The main contributions of this section are summarized as: a) The characterization and implementation of a monitoring platform for edge cloud computing in a wireless mesh network environment over a gossip overlay; b) The understanding of service, resource and network properties that relate to the functionality of monitoring with the use of a gossip overlay for data dissemination.

Our work leverages a gossip overlay in wireless mesh networks to disseminate monitoring information in a fast and efficient manner. Gossiping protocols allow for rapid transmission of information across the network, i.e. each node only needs to contact a subset of neighboring nodes, instead of the whole network. The gossiping mechanisms gives guarantees (such as resilience to node failures, eventual data consistency) towards its optimal application when instability of the network is constant, with high node churn or high latencies. Furthermore, we integrate the monitoring platform into the already deployed technologies (e.g. Cloudy distribution) that are part of community network clouds and its services. The monitoring platform in Cloudy is also designed to become a Software-As-a-Service for users to install on their own devices, to support users in monitoring usage.

4.1.2 Monitoring platform for Edge Clouds

We extended the Cloud distribution with a monitoring platform, towards enhancing the information gathered from edge cloud services. The platform

aims to gather raw data from the shared devices, and disseminate the relevant information to the community.

The platform requires to have shared data among the network members; to be able to access information about services and resources at each of the shared nodes; and, to have enough resources to process the raw data and store the processed data. Therefore, making use of the available devices and services, and support the knowledge of their usage to the community.

The CN environment creates its own challenges, differing from classic cloud environments, which need to be addressed, such as low-capacity devices used, network changes (node churn rate), low bandwidth and high latency between nodes and user related privacy concerns. In our work, we address these challenges by using eventual consistency and gossiping methods on the shared data, while also making sure that, by means of Serf, that each node can join or leave the network without affecting the overall information within the network.

The type of information monitored are the resources, services and social aspects of the community. For our work, one of the main reasons to gather knowledge on the community cloud system, is to understand social behavior on the network and the usage of services and resources. Also, the monitoring data can be extended with additional types of information, such as service configuration, resource configuration and usage. Nevertheless, the boundaries for such information sharing, need to meet the security issues that can arise from contributing users' privacy related aspects, out of the scope for this work.

The monitoring platform is split in three stages: 1) Logging of raw data from services, resources and user interaction shared by the member nodes; 2) processing of the raw data, into a format that is user readable and can be shared among the community, such as a time-line of service usage; 3)

dissemination and presentation of the results to the users in the community, through the user interface, which is an additional service in Cloudy. These stages represent the major objectives on which we focus our attention, tailoring them towards an efficient monitoring service on the edge community cloud.

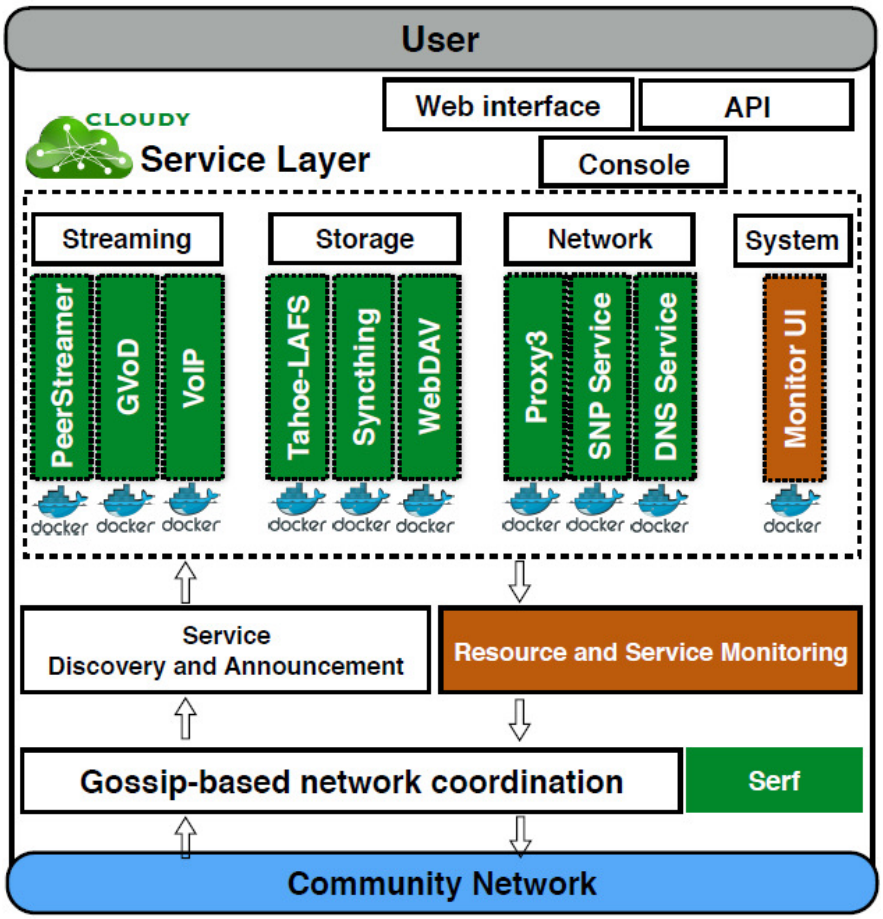


Figure 4.1: Overview of Cloudy modules, showing the monitoring components and the integration with the gossip overlay.

Figure 4.1 presents an overview of the modules in the monitoring platform integrated in Cloudy. The integration of the monitoring part is done in the

middle layer, where a service is added to enable users to install and see the results of monitoring across the network. Additionally, the resource and service monitoring module is added in the communication path with other nodes and services, to gather the necessary information from the services and to disseminate it to other nodes using the gossip overlay. The monitoring module utilizes the already existing gossip overlay, created through Serf, to disseminate the relevant information to the other nodes. Meanwhile, the module also uses the same overlay to retrieve information from other nodes. Data dissemination and convergence in nodes happens at Serf level, since data is retrieved from the local Serf instance.

The reason for integrating the monitoring platform in Cloudy, as explained above, is because logging of services is required to occur during service initialization. The use of a monitor UI as a service is done to simplify the view of shared data from the users perspective, in this way, users have access to relevant information from their services and the usage of services across the network. Also, the implementation involves both low-capacity devices and edge cloud computing paradigms.

Additionally, the information gathered from the monitoring service has two functions: presented to the users to motivate them on how services are being deployed and used on their community network; as well as, for management purposes, the information gathered can give an current picture on how the network, service utilization and allocation is done across the network, e.g. hotspots of utilization, time-frames where resources may be overused, among other examples.

4.1.3 Monitoring Platform Prototype

The prototype for the monitoring platform was developed using the available low-capacity devices and connected to the community network (Guifi.net). The devices used are equivalent to those deployed by the Clommunity project[2] in the community network at users' homes.

The prototype shows the feasibility for monitoring services and resources under edge cloud computing. Therefore, services were started and terminated at certain intervals of time in the available nodes, in order to gather service usage information. The information for each service is sent to Serf when a service is published. The information is updated in Serf when the service terminates. Thus, all nodes can gather the logs of services within the data coming from Serf members (nodes interconnected in the gossip overlay).

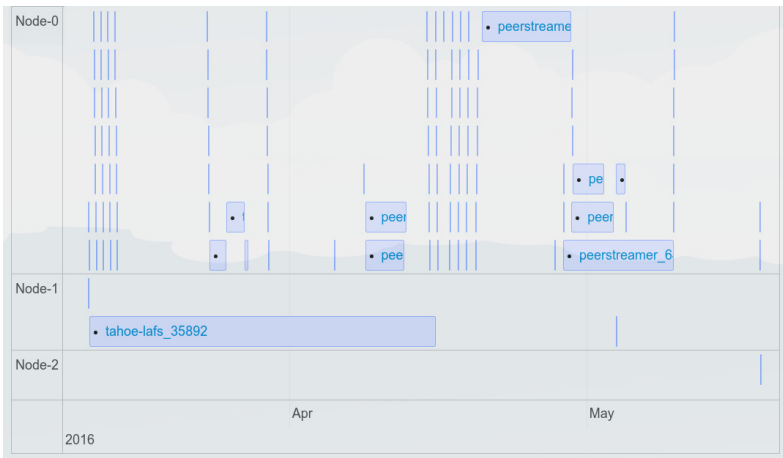


Figure 4.2: GUI of the prototype for the monitoring of Cloudy services, using three devices interconnected.

[2]http://clommunity-project.eu/

In our prototype, we gather information and process it on one node, to be shown as a time-line graph of service usage, as seen in Fig. 4.2. The figure depicts the time of actions across three nodes, such as publishing and unpublishing of services, where each bar represents time-wise the service usage for a given node. Moreover, information relevant to the users, that comes directly from the nodes in the network, can be displayed in the GUI. Furthermore, since data travels through the gossip overlay, the whole network information (services from all nodes) becomes more accurate over time, when all nodes' data converges, as low as one second in Serf[3].

From the prototype monitoring platform, we could observe that using a gossip overlay to disseminate information is as a good option for non-critical analysis of the overall network. The shared information can be gathered at any of the nodes that are members of the gossip overlay, within certain conditions such as the time delay for the eventual consistency of the data, and the amount of data that is updated to each node. Moreover, the size of the data sent to other nodes appears small enough (in the range of kilobytes) to not affect significantly the available network bandwidth. However, it is foreseen that clearing and storing past data is required so that nodes can efficiently exchange information between each other.

4.1.4 Experimental Results

We conduct an evaluation by emulation of the monitoring platform, as the means to understand the characteristics and properties relevant to a real deployment. The simulation of network data gathered gives us insight on the best practices for an efficient way of dealing with dissemination of data across a wireless mesh network. Whereas, the emulation of nodes provides details

[3]https://www.Serfdom.io/docs/internals/simulator.html

about the scalability of the monitoring platform for edge cloud computing. In this way, we can understand the properties and issues that arise when dealing with higher number of nodes, higher latencies and the use of low-capacity devices.

The characteristics of the monitoring platform come into evidence when we analyse the results of data dissemination and convergence, scalability of the platform and the tuning of Serf properties. Data convergence gives us the amount of data that a node receives across time, from the total disseminated data. This means that a certain amount of time passes until a node gathers the total data (or convergence time). The time elapsed between disseminating and convergence of data is then important to understand how reliable the system is when using a gossip overlay.

For our evaluation, we emulate nodes and simulate the network, giving us the average of time for dissemination and convergence data rates, while also being able to tune certain aspects (such as gossip interval, gossip fan-out) of the overlay created through Serf. Therefore, we used the Mininet simulator [52], merged with Mininet-Wifi [53] and Mininet ContainerNet[4]. Mininet-Wifi adds to Mininet the ability of simulating wireless links. The capability of simulating wireless links is then more attuned with the environment created with community network clouds. The ContainerNet project enables each of the emulated nodes to run as Docker containers, guaranteeing execution isolation for each of the emulated nodes. Furthermore, we are able to run different executions of the same applications, such as Serf, without interference among each emulated node.

In the experiments done, we used as network topology a random geo-positioning of the nodes, where each of the nodes positions itself within a maximum range

[4]https://github.com/mpeuster/containernet

of 100 meters of another node. The characteristics of the topology are drawn from the CNs, where each person connects to their nearest neighbors to join the network. Therefore, each experiment run uses a randomly created topology, in an attempt to not be influenced by a given network topology.

Moreover, the positioning of the nodes influence the overall latencies in the network. The nodes average latencies can be as high as 800 milliseconds. Also noting that the simulated network shows as a worst case scenario, in fact other experiments done on CNs [50] tend to experience, on average, lower latencies and higher bandwidth between nodes, on normal usage of the network. However, our evaluation comprises the worst cases, to infer on the monitoring activity when high latencies are dominant in the network.

The evaluation is performed with several runs (around 10) and their results are averaged. The average on these runs are enough to point out the characteristics that determine the efficiency of the monitoring platform. Each experiment has 40 virtual nodes (Docker containers), interconnected through a virtual mesh network and randomly positioned in the network.

For each of the experiments the services are started (Publish action) and terminated (Unpublish action) within a time-frame of 10 minutes in each of the nodes. The two actions are propagated to other nodes where each of them will publish the information of the service and update it afterwards. Noting that the first action happens before the gossip overlay is fully known (nodes require to know about other nodes in the network to send data to a subset of known neighbors). The expected time elapsed for each action across the network is under the time-frame given. Also, the actions monitored are the same as in the real world situation, where users start sharing their services and terminate them.

Furthermore, the monitoring process will gather the shared information through the gossip overlay, over the time-frame. In our results we show the data convergence on nodes to understand how much time it takes for nodes to have the same view of the shared data.

For a scalability evaluation, we performed several emulations with different number of emulated nodes, with the same conditions as previously mentioned. The conditions are maintained to be similar to the environment created on CNs (high latencies, low-capacity nodes). In this way, we can understand the issues on deploying the monitoring platform across bigger networks, and address the requirements for edge cloud computing.

Furthermore, we extend our findings by tuning the properties of the gossip overlay to be used under wireless mesh networks. Thus, we changed the profile for the Serf gossip properties, adjusting the gossip interval, gossip fan-out and overall timeouts, to gossip less frequently, but to an additional node. These changes are made to improve the performance of dissemination and convergence of data, therefore enhancing the monitoring data exchange between wireless nodes.

The reason for the number of nodes used in the experiments, is because a virtual environment was used to deploy Mininet. Therefore, the virtual machine was constrained and the deployment of an higher number of nodes would lead to be unable to reflecting realistic conditions of resource usage.

Results: We can observe the rate of data convergence across the simulated network and infer on the data dissemination that occurs with our solution.

Fig. 4.3 shows the percentage for average monitoring data convergence for all nodes, after the services were published (dashed line) and unpublished (continuous line), averaged from all the experiments done. The actions are not

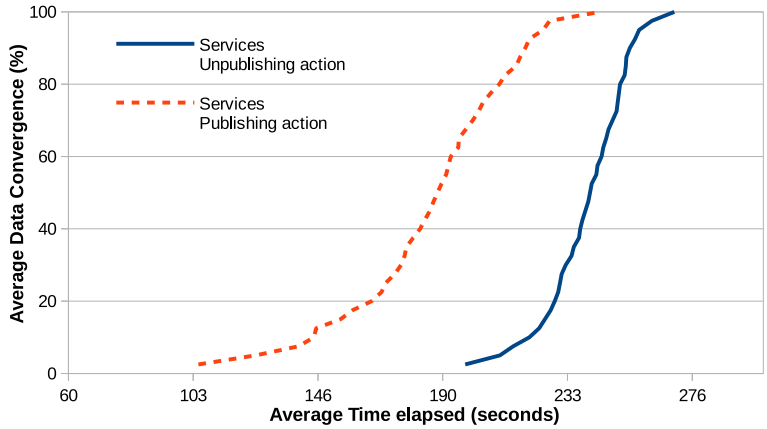


Figure 4.3: Averaged data convergence in the time elapsed for the actions of publishing and unpublishing services.

immediately propagated to the network, therefore the dissemination occurs some time after starting. Also, we can see that the convergence of data in both actions happens within 1 minutes in high latency conditions. In this figure we observe the convergence is done faster for the unpublish action than for publish, this is because the nodes already know about others in the network, and thus it only requires to update the current information. The figure also demonstrates for each action the data convergence on all nodes, on average, is 130 and 80 seconds in each action respectively.

Fig. 4.4 demonstrates the scalability of having the monitoring data exchange with a gossip overlay. The figure shows the time elapsed of the combined actions for starting and stopping a service and the fully convergence of information within a node. Furthermore, we can say that the results obtained imply a linear evolution of the time elapsed when adding more nodes to the network with high latency connections, taking into account the overall latency (edge to edge) increase when adding more nodes. The results show that in the community network scenario and with the high latency conditions, the nodes

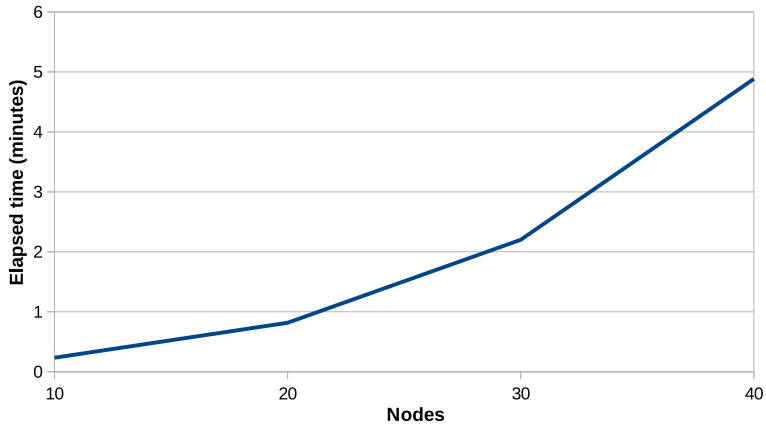


Figure 4.4: Scalability results, between number of nodes and time elapsed until service is unpublished.

can still gather information quickly enough to have the knowledge of the network without issues.

Fig. 4.5 depicts the same actions (Publish and Unpublish of services) as previously, however the gossip properties (Gossip interval and Gossip fan-out) were adjusted for wireless environments. The figure shows that the optimization done has an effect on the data dissemination and convergence on the nodes. Furthermore, the figure also demonstrates that the convergence of data for the nodes, on average, is 87 and 38 seconds in each action respectively. Therefore, each of the actions shown, on average, are faster (around 25%) than with the previous gossip properties.

4.1.5 Discussion

The evaluation provides relevant characteristics of monitoring in community network clouds. Therefore, we discuss about the usage of a gossip overlay to disseminate information, and monitoring non-critical information (information that is not necessary in real-time). However, we can also be assured that the

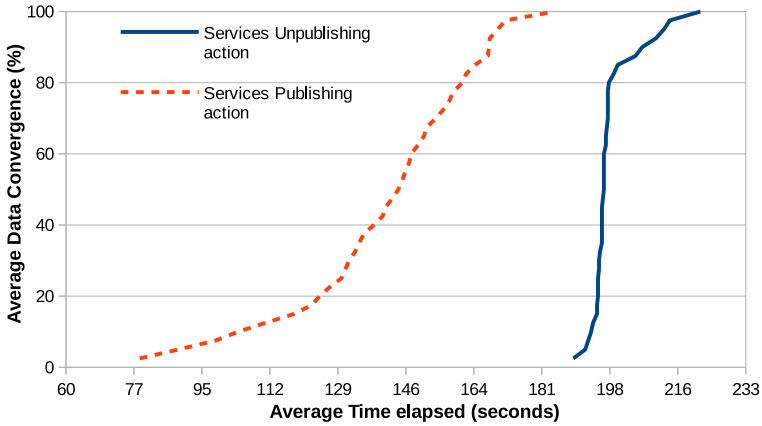


Figure 4.5: Average data convergence in time elapsed for the actions of publish and unpublish services, using the tuned gossip properties for wireless devices.

dissemination is done quickly (within 1 minute under high latency situations), depending on other factors such as number of services running, size of shared information, bandwidth to each node, offline nodes or failures in the network.

The usage of a gossip overlay for communication between nodes can have its own drawbacks, such as delayed dissemination of data. However, the case of community network clouds information gathering is an optimal solution, since the system does not need real-time knowledge of the network. The usage of a non-gossip overlay would require prior knowledge of the nodes in the network, or when using other techniques (e.g. one to all nodes) the network could become flooded and unsustainable for communication.

In our work, a real usage evaluation scenario can give us insight on how monitoring should behave. However with emulated nodes and a simulated network we can understand the trends that monitoring systems can have on community network clouds, such as the high number of updates that may occur for the dissemination of data. We can also understand the properties that gossip-enabled networks have and customize them for wireless mesh networks,

such as varying the number of neighbors to disseminate data and the gossiping time interval.

We can also say that by tuning the gossip overlay properties (gossiping fan-out, gossiping interval and overall timeouts) we can improve the data dissemination, and therefore monitoring data can be exchanged faster between wireless nodes. Thus, gossiping fan-out and interval can be modified in accordance with the number of nodes there are in the network, and the characteristics of the network, in order to enhance the performance to data dissemination for CNs. The gossip overlay properties are then required to be tuned according to the position of the nodes in the network and its wireless capabilities.

Furthermore, the option to have a centralized monitoring system, while gathering decentralized logging is an approach that can be successful within CNs and low-capacity devices. However, further study on decentralizing the monitoring system (including processing and storage) is required to pursue an optimal solution in respect to sharing network knowledge, such as hierarchically, grouped or cluster gathering and storing of data.

4.1.6 Summary

This section analyses the monitoring provisioning that is required for edge cloud computing environments and envisions a monitoring platform to be used for edge cloud computing. Data dissemination was done with the introduction of gossip-enabled network, interconnecting the nodes of the wireless mesh network, as is the case of CNs. The monitoring data is then gathered through the gossip-enabled network to be shared across the entire network. This is done to give users knowledge about their services, and the community cloud environment, while expanding the knowledge on how management of such clouds is possible.

In our experiments with node emulation and network simulation we show that the dissemination of data over a gossip-enabled network is done quickly within minutes of the start, and is optimal for non-critical shared information. We also observe that under high latency situations and with low-capacity devices the use of a gossip-enabled network is a best practice to overcome the harsh conditions, while removing the need to flood the network with information, or to know the structure of the whole infrastructure.

The creation and deployment of a monitoring tool that is tailored for CN micro-clouds, grants us the knowledge on service performance, usage and quality perceive. Therefore, we can ask how do services perform within CN micro-clouds? What is the service quality perceived by the users?

4.2 Analysis and Optimization of Services in CN Micro-Clouds

In this section we look at the service level, where we asked how the services perform, what is the service quality perceived by users and how can services be improved without having to change the entire environment. Therefore, we analyze a type of service (live video streaming), which gives a broader view of the behavior of the network, and serve as a representative for other services.

In this work, we used Cloudy, which fosters the service deployment and automation in CN micro-clouds. We present two ways provisioned by Cloudy to integrate the services and improve the users QoS in these clouds. First, we present a distributed service discovery mechanism that helps users with service quality metrics to choose the best service from a pool of instances. Second, we experiment with a live video streaming service deployed in CN environments, using more than 50 real CN nodes across Europe for the evaluation. Our analysis shows that tuning the vital parameters of this service as neighborhood

peer selection strategies, and source node dispersion strategy, improves the video streaming QoS in the CNs. Our results indicate that both ways help the user to experience improved service performance. Automated service selection, needed once the number of micro service providers becomes larger, is the next step that can be built upon our results.

4.2.1 Overview

The deployment of services in CN micro-clouds allows to offer resources and applications, which are of value for the users and meet their particular needs and interests. Among the services that are very appealing, P2P live streaming is an important candidate, as can be seen by the growing success and usage of commercial systems such as PPLive, SopCast. P2P live streaming systems allow to watch live streams such as events or television channels over a network, granting anyone to become a content provider.

To enable these types of services within CN nodes is very challenging, since community networks are diverse and dynamic networks with limited capacity of wireless links and often low-resource and cheap devices. Streaming applications, however, have high demands of bandwidth, they require low and stable latency and only withstand low packet loss.

Our motivation begins with the integration of a cloud-like system in community networks which gives users the opportunity to use services (e.g. video streaming) in their constraint devices (home gateways), without relying on the commercial clouds. Furthermore, we extend our motivation towards providing the service ease of usage and optimization of QoS on the challenging environment of CNs.

The contributions of this section are the following:

- Integration of a P2P live streaming service in the Cloudy distribution, and enable the automation and provision of this service in CN micro-clouds.
- Implementation of a search service based on SERF that allows the P2P live streaming service to be published and discovered by users in the community network cloud. Furthermore, we add a QoS-aware service selection algorithm that allows users to choose the best service from a pool of instances, according to network metrics.
- Evaluation of the performance of PeerStreamer as a P2P live streaming service deployed over 55 geographically distributed real community network nodes. Study the effects of different parameters of PeerStreamer on its performance in the community network environment, in order to understand how other services can behave in CN micro-clouds.

4.2.2 Community Networks Micro-Clouds

Our proposition is to deploy the PeerStreamer service on CN micro-clouds. The resources are therefore heterogeneous, geographically distributed, and often with resource constraints. Home gateways located in user homes can become cloud resources and they are integrated as Community Home Gateways (CHGs). From an administrative perspective, these CHGs are peer-to-peer infrastructures. The community network cloud we envisioned consists therefore of user-contributed infrastructures, such as home gateways, connected to the cloud in a peer-to-peer fashion, used for the collective provision of services that are of interest for the community.

Our model fits to the general cloud computing deployment categories. Besides public, private and hybrid models of cloud computing, a community network cloud differs from the others in that it is designed with a specific community

in mind, and where costs and responsibilities are shared among the community members. Such community network cloud model assumes that cloud users can be classified into communities, where each community of users has specific needs in terms of services. We identify in CN such a community as a *micro-cloud*. A micro-cloud has a reduced number of nodes which are close as in Figure 4.6. This closeness in the context of CNs can be of technical and social nature. Cloud nodes within a micro-cloud announce their services and discover other nodes within the micro-cloud they belong to.

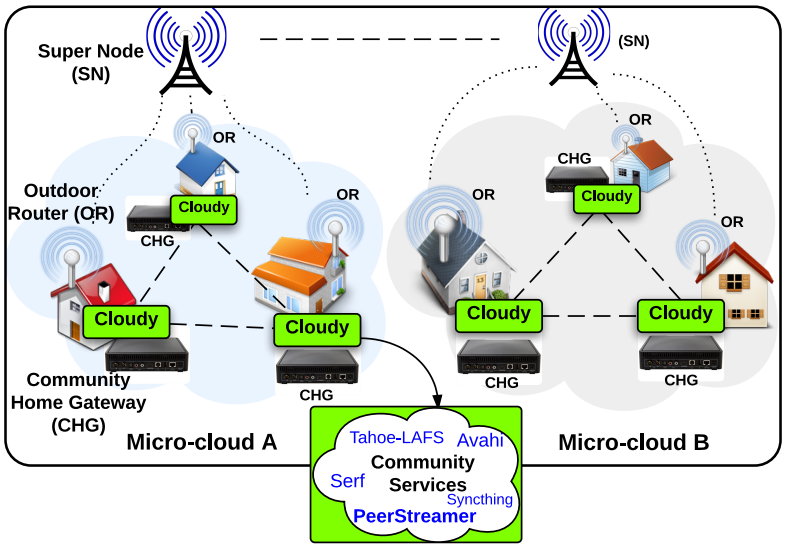


Figure 4.6: Community network cloud nodes, grouped into micro-clouds. The nodes of micro-clouds are spread on different locations inside the CNs, forming a meta cloud environment (community network cloud).

4.2.3 Live Streaming Service

PeerStreamer is an open source live P2P video streaming service, and mainly used in our Cloudy distribution as the live streaming service example. This service is built on a chunk-based stream diffusion, where peers offer a selection

of the chunks that they own to some peers in their neighbourhood. The receiving peer acknowledges the chunks it is interested in, thus minimizing multiple transmissions of the same chunk to the same peer. Chunks consists of parts of the video to be streamed (by default, this is one frame of the video). At the beginning of the streaming process, these chunks are all from the same peer (since only one peer is the source), then the source sends m copies of the chunks to random peers ($m = 3$ by default), creating an overlay topology with all peers [54] in order to exchange chunks between them. The whole architecture and vision of PeerStreamer is described in detail in [7].

4.2.3.1 PeerStreamer Assumptions and Notation

We call the community network the *underlay* to distinguish it from the *overlay* network which is built by PeerStreamer. The underlay network is supposed to be connected and we assume each node knows whether other nodes can be reached (next hop is known). We can model the underlay graph as:

$$G^{ug} = (S, L^{ug}) \tag{4.1}$$

where S is the set of super nodes present in community network and L^{ug} is the set of wireless links that connect them. This is the global level.

In the micro-cloud level we have a set of outdoor routers (OR) that are connected to each other in the same micro-cloud as shown in Figure 4.6,

$$G^{um} = (OR, L^{um}) \tag{4.2}$$

where OR is the set of outdoor routers present in the micro-clouds of the CNs and L^{um} is the set of wireless links that connects them.

The nodes of the underlay (connected to super nodes through outdoor routers) run an instance of the PeerStreamer and are called *peers*. Each peer P_i at time t chooses a subset of the other peers as a set of neighbours that are called $N_i(t)$. The peer P_i exchange video frames (chunks) only with peers in $N_i(t)$, and the union of all the $N_i(t)$ and the related links defines the network topology of the application, also represented as graph and called *overlay*. The overlay built by PeerStreamer is a directed graph:

$$G^{og}(t) = (P_{set}, L^{og}(t)) \tag{4.3}$$

where P_{set} is the set of peers and

$$L^{og}(t) = (P_i, P_j) : P_j \in N_i(t) \tag{4.4}$$

is the set of edges that connect a peer to its neighbours. The main difference between the overlay and the underlay is that the underlay is determined by the network topology, on which PeerStreamer does not have control, while the overlay is generated by PeerStreamer.

4.2.3.2 PeerStreamer integration in Cloudy

The version of PeerStreamer that is bundled with Cloudy, only features UDP streaming for video input, which is an acceptable transport protocol for video streaming. Therefore, we need to consider this fact in our stream provision. Either an online stream can be used (with the help of other applications) or a local video streamed to a local port is used. However, most of the video streams in the Internet do not use directly the network-level UDP protocol, instead it is more common to use an application-level protocol, such as RTSP/RTP[5]. In order to include PeerStreamer in Cloudy we choose the

[5]http://www.ietf.org/rfc/rfc2326.txt

lightweight PeerStreamer version since we have low-resource machines in our community cloud deployment.

4.2.4 Service Discovery

Cloud services in the context of CNs are built and operated in a decentralized way, and need a common place for both providers and users respectively, to publish their services and learn about their availability. In Guifi.net, the available network services are normally declared on the web page, by manually submitting the details (type of service and specific characteristics, location, IP address, terms of usage, etc.). The lack of automated methods for publishing services, and also for conveniently finding out which are the ones closest to the user, has led to a couple of drawbacks: not all the services are declared on the website (although they are announced by other means, like users mailing lists) and when a service is temporarily or permanently unavailable, it still appears on the website as *online* until it is manually removed from the list. In this section we show how we implement and use the automatic service discovery based on Serf to discover services such as PeerStreamer in Cloudy instances.

4.2.4.1 Serf Implementation

The distributed announcement and discovery of services (DADS) operates in parallel at both the global community network cloud level and at the micro-cloud level. On each of these two levels a different technological approach is used. Cloudy includes a tool to announce and discover services in the CN clouds based on Serf, a decentralized solution for cluster membership, failure detection, and orchestration. Serf relies on an efficient and lightweight gossip protocol to communicate with other nodes that periodically exchange messages between each other. This protocol is, in practice, a very fast and extremely efficient way to share small pieces of information. An additional byproduct

is the possibility of evaluating the quality of the point-to-point connection between different Cloudy instances. This way, Cloudy users can decide which service provider to choose based on network metrics like RTT, number of hops or packet loss (Algorithm 1). The second level of DADS occurs in the micro-cloud, where a number of Cloudy instances are federated and share a common, private Layer 2 over Layer 3 network built with Getinconf[6]. At that level, Avahi[7] is used for announcement and discovery. Originally this solution was to be applied to the whole CN but as more Cloudy instances started to appear it became clear that the solution would not scale further than the tens of nodes as we explain in [55]. However, in the context of an orchestrated micro-cloud, it can be used not only for publishing cloud services but also other resources like network folder shares, etc.

When Serf finds different services (including services of the same type) we need to provide a QoS-aware service selection approach that will help users to choose the best quality of service among all instances. It is worth noting that a service with consistently good QoS performance is typically more desirable than a service with a large variance on its QoS performance. This would allow users to choose the best service available ranked according to some important community network parameters.

When a Cloudy client issues a find service request, Serf obtains the service list available and related service availability degree. Service availability may include many aspects to service i as S_i, we denote as A_{i1}, A_{i2}, A_{i3},...A_{ij},..., A_{im}, where m is the attribute number of each service. The services can have attributes as RTT, packet loss, throughput etc. We use W_{ij} to denote the importance weight of every attribute of service i, where j=1, 2, 3 ...,m and

[6]https://github.com/Clommunity/getinconf/
[7]https://avahi.org

ϵ as a preference weight of the user for a given type of service. Taking into account this, the service availability can be described as A_i, in Equation 4.5. We specify also a service availability threshold λ, which denotes that if a service with A_i is greater than specified λ, then the service is available and it is added to the available service list set.

$$A_i = \sum_{j=1..m} (W_{ij} A_{ij}) - \lambda \tag{4.5}$$

Algorithm 1 ServiceSelection(S_i, W_{ij}, A_{ij})

```
1:  // S_i ← service in the cloud, A_ij ← the jth attribute value of service i,
    W_ij ← the weight of importance degree, ε ← user preference weight, λ ←
    the availability threshold;
2:  procedure S–SELECTION
3:      AvSet={};
4:      for each S_i in the Community Cloud do
5:          if S_i is in Micro-Cloud then
6:              W_i = W_i * ε where ε > 0 ;
7:          end if
8:          calculate A_i with equation (4.5);
9:          if A_i ≥ 0 then AvSet = AvSet U {(S_i,A_i)}
10:     end for
11:     sort(AvSet) order by descending;
12: end procedure
```

By default, Serf is used in Cloudy in order to simplify the process of service discovery for the users by utilizing the QoS-aware service selection algorithm (Algorithm 1).

4.2.5 Experiment Setup

For the experimental research, our main configuration includes geographically distributed CN nodes from Guifi.net in Spain, AWMN in Greece and Ninux

in Italy. These nodes are co-located in either users homes (as home gateways, set-top-boxes etc) or within other infrastructures around each city. Nodes are deployed to use the wireless links of each community network that operate in the ISM frequency bands at 2.4 GHz and 5 GHz. The connectivity between CN nodes varies significantly. Two CNs (Guifi.net and AWMN) are connected on the IP layer via the FEDERICA[8] (Federated E-infrastructure Dedicated to European Researchers) infrastructure, enabling network federation. The nodes of Ninux CN in Italy are not connected to FEDERICA, therefore we experiment with them separately (without including other CN nodes). In our experiments the nodes from UPC (Technical University of Catalonia) are a subset of Guifi.net CN nodes which are distributed in our UPC campus in Barcelona. We use these nodes as a baseline in order to be able to better understand the effects of the network given by the statistical data gathered from the community networks.

In order to deploy the PeerStreamer application in a realistic community network cloud setting, we use the Community-Lab [17] infrastructure which is a distributed infrastructure provided by the CONFINE project, where researchers can deploy experimental services and perform experiments in a real and production CN.

Our experimental evaluation is comprised of 55 physical nodes distributed across Europe, among the working nodes available from the three CNs. Most of the nodes are built with an Intel Jetway device equipped with an Intel Atom N2600 CPU (2 cores), 4GB of RAM and 120GB SSD and running a custom Linux OS (based on OpenWRT), which makes them resource constrained devices at the edges of the network. Table 4.1 shows the number of nodes used in three community networks, their location and type of devices deployed.

[8] http://www.fp7-federica.eu/

In our experiments we connect a live streaming camera (maximum 512 kbps bitrate, 30 fps) to a local PeerStreamer instance which acts as the *source* for the P2P streaming. We choose as a source a stable node with good connectivity and bandwidth to the camera in order to minimize the video frame loss from the networked camera. The source is responsible for converting the video stream into chunk data that is sent to the peers. In the default configurations of PeerStreamer a single chunk is comprised of one frame of the streaming video. Also, the source PeerStreamer node sends three copies ($m = 3$) of the same chunk to the peers, meaning that only three peers receive the chunks directly from the source at a given time. Thus, each peer that receives the chunks exchange with other peers in order to form the P2P exchange network.

The evaluation metrics presented were chosen in order to understand the network behavior, quality of service and quality of experience. Thus, for network behavior section 4.2.6.1 explains in details the network measurements obtained. On the quality of service side, we measure the number of chunks that are received by peers and the chunk loss percentage in order to understand the impact of the network on the reliable operation of this type of service. On the quality of experience, we gather statistical data from the chunks that are played out locally by each of the peers to understand the quality of the images that the edges show to the users. These metrics, show the impact of such networks when using streaming services while also guaranteeing the image quality that each node can display on average. Regarding the network interference issues of other users' concurrent activity which can impact the results of the experiments, we reference to [56] and is out of the scope of this work.

Table 4.1: Nodes in the cluster and their location

Nr. of nodes	Cat.	Location	Type
23	UPC	Barcelona, Spain	Physical nodes and VMs
8	Guifi.net	Catalonia, Spain	Physical nodes
12	AWMN	Athens, Greece	Physical nodes
12	Ninux	Rome, Italy	Physical nodes

4.2.5.1 Scenarios

To assess the applicability of PeerStreamer in CNs, the following describes a chosen scenario that reflects a use case of live video streaming in CNs. Also, we augment our findings with a scenario reflecting different parameters of PeerStreamer usage, in order to understand possible improvements of the service level created by the PeerStreamer instances. The parameters used in the scenarios are summarized in Table 4.2.

Table 4.2: Summary of our Scenario Parameters

Scenario	1 and 2
Total number of nodes	55
Groups of nodes	UPC, Guifi.net, AWMN, Ninux
Tests time-frame	T1 = 30m — T2 = 1h — T3 = 2h
Source 1 Send Rate (chps)	T1 = 31 — T2 = 32 — T3 = 31
Source 2 Send Rate (chps)	T1 = 55 — T2 = 55 — T3 = 49
Metrics	Peer Receive Ratio, Chunk Loss Chunk Playout, Neighborhood Size

For the first scenario we choose the default parameters of PeerStreamer and run in the challenging environment of CNs. One of the nodes, which has the best connectivity to the camera stream is chosen to be the source peer, while the rest of the available nodes will initially contact the source in order to enter

the P2P network for chunk exchange. Since the Ninux group of nodes do not have connectivity in IPv4 to other CNs (they are not part of FEDERICA), we deliberately executed the experiment apart from the other CNs, in order to understand different CNs network behaviors. The experiment ran on this group was different because of the non-connectivity to the camera stream, therefore another solution was devised. We introduced a live TV streaming channel as the streaming source, transcoded to 512 kbps bitrate, 30 fps on average similar to the camera stream. However, this stream also included audio, which made the exchange of data between peers higher than the peers of other CNs. Each experiment is composed of 20 runs, where each run has 10 repetitions, and averaged over all the successful runs (90% of the runs were successful). In the 10% of the runs the source was not able to get the stream from the camera, so peers did not receive the data. The measurements we present consists of 3 weeks of experiments, with roughly 300 hours of actual live video distribution and several MBytes of logged data.

We then establish three experiments shown for 30 minutes, 1 hour and 2 hours of continuous live streaming from the PeerStreamer source. This was done in order to gather statistical information within different time-frames and to use as initial step towards live events coverage on CNs. Other nodes were started at the same moment in time, 10 seconds after the source started, in order for the source to gather enough data to be able to exchange with the peers. This also allows the randomization of the nodes that the source PeerStreamer will first push the chunks to, and thus on all experiments the peers that begin receiving chunks from the source will be different (PeerStreamer overlay topology changes in every run of experiments). In all experiments we try to guarantee the number of nodes to remain constant. However, since we are dealing with a very dynamic and challenging environment, there is an issue of churn rate of nodes. This happens in the CNs because most of the nodes are

connected wirelessly and their connectivity depends on many factors (such as weather, electric failures, router connectivity, among others). PeerStreamer for its own overlay performs operations to manage the peer churn rate by constantly updating each peer neighborhood, an important feature for the potentially unstable and dynamic nodes that we find in community networks.

For our second scenario, the evaluation performed includes the findings of different configuration parameters of PeerStreamer, which results in better quality streaming. This was done in order to understand the different behavior of the PeerStreamer algorithms which helps to optimize its instances. The different parameters chosen include sending different amount of copies of the chunks from the PeerStreamer source ($m = 5$, $m = 1$); keeping the best peers in the neighborhood in between topology updates of the overlay that PeerStreamer creates ($TopoKeepBest$); and the addition of the peers that can be selected to the neighborhood by extending the default RTT (10 ms) of the peer selection metric [7] to 20 ms .

4.2.6 Results

4.2.6.1 Characterizing the Network Performance

Typically, CN users have an outdoor router (OR) with a Wi-Fi interface on the roof, connected through Ethernet to an indoor AP (access point) as a premises network. In Guifi.net where nodes are located, OSPF, BGP, BMX6 [57] or combination of them is used as a routing protocol. In AWMN and Ninux BGP and OSPF are mainly used between outdoor routers. Most of the super nodes (the ones routing the traffic between the different zones) are working in AP mode. The nodes (home gateways) where the PeerStreamer application is running are connected to these super nodes through their outdoor routers. A few super nodes are placed strategically on third party

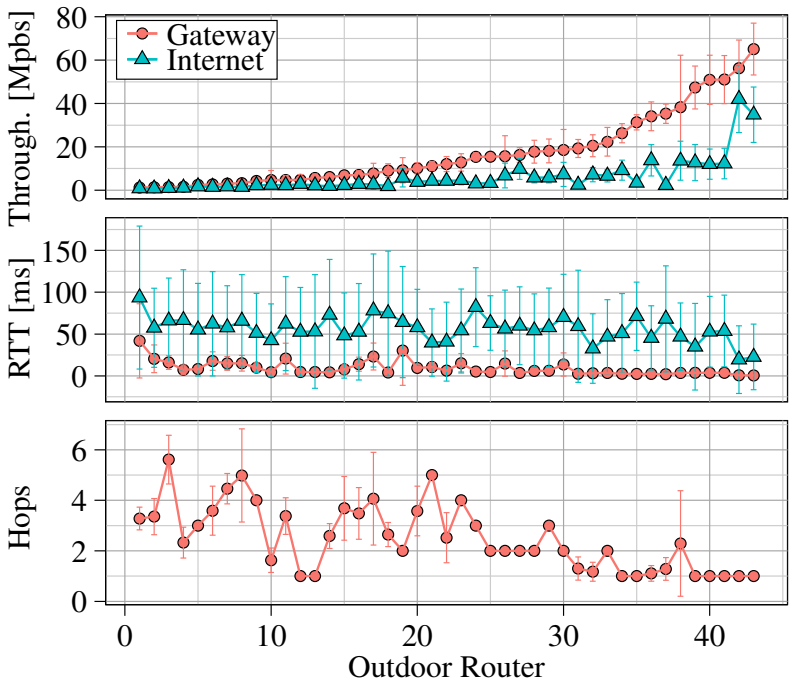


Figure 4.7: Average throughput and RTT to the gateway/Internet and number of hops to the gateway.

locations, e.g. telecommunication installations of municipalities, to improve the community network's backbone. In order to gain insight for network behavior in community networks we monitored the network for a period of 30 days.

Figure 4.7 shows the average throughput and RTT to the gateway (proxy) and the Internet, and the number of hops to the gateway obtained for every OR. The values are sorted by the throughput to the gateway. Standard deviation error bars are also given. Internet values are measured using a server located outside of Guifi.net. The figure also reveals that the throughput to the Internet and the gateway are not linearly correlated. The average throughput

to gateway is 17.4 Mbps and to the Internet 6.3 Mbps. This is because one of the gateways in CNs has a better connection to the Internet. Thus, even if the throughput to the gateway is high, those nodes using the second gateway in other parts of the network have a low throughput to the Internet. Furthermore, it demonstrates that the RTT has a stronger correlation with the number of hops than the throughput (average RTT to the gateway is 9.26 ms and to the Internet 56.3 ms). Error bars reveals that some nodes have an average number of hops with noticeable deviations. This variability has two causes: change in the routes, and selection of a different gateway.

4.2.6.2 Scenario Results

Figure 4.8 depicts the amount of chunks on average the peers receive. Knowing that Source 1, sends out to the peers around 31 chunks per second (chps), we notice that the distant groups (Guifi.net and AWMN) in relation to the source, receive less chunks than the closer group (UPC), in relation to the source. This is because of the network impact on the delivery time of the chunks. Thus, more chunks arrive out of the time allotted, the farther the chunks have to travel. We also notice that the number of chunks received on average increases with longer time-frames, this occurs because the peers can gather more statistical information about each other and therefore update their neighboring peers accordingly, while securing a subset of peers in which they can rely on to receive the chunks in the time allotted to be displayed. We also show that on Ninux side the amount of chunks received tends to be higher that of the other CNs. This is due to the fact that we use a different stream (Live TV channel stream), in which Source 2 sends around 55 chps instead (accounting with the added audio part of the stream). We also notice a drop of receiving chunks for longer times, because of the inherited instability

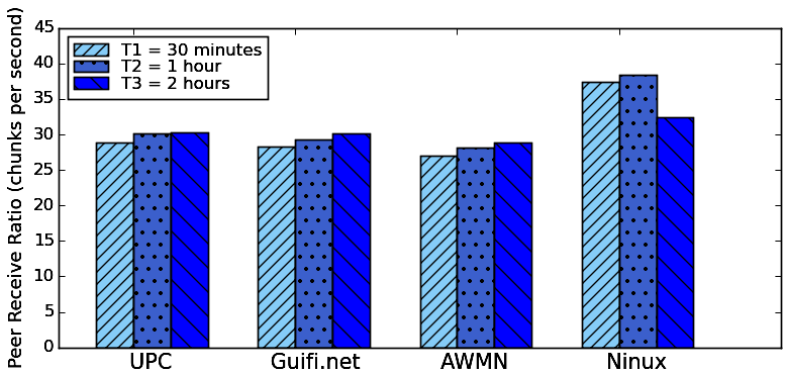


Figure 4.8: Average Peer Receive Ratio

of this group of nodes, where the loss of data is more constant/visible when dealing with longer times.

Figure 4.9 shows the average chunk loss for each group of peers. We can see that the loss is greater for shorter time-frames (loss in UPC 7%, Guifi.net 9% and AWMN 13%) and are amortized for longer time-frames (loss in UPC 2%, Guifi.net 3% and AWMN 7%). We also notice that distant groups (distant from the source stream) are more affected by the diminished rate of chunks received, which demonstrates the influence the network has to the amount of data that is lost (either by losses on the network or by not arriving on time to be displayed). As for the Ninux group, as previously mentioned, the network behavior is more volatile since there is a higher packet loss. Therefore, we notice that since Source 2 sends more chunks per second (around 55) than Source 1, the loss of chunks in the peers is greater than in other groups and in longer time-frames the network instability has a higher impact on the data exchanged (34% loss).

Figure 4.10 illustrates the quality (chunks played) of video offered on the peers side. The closer groups display more chunks, because the loss between farther

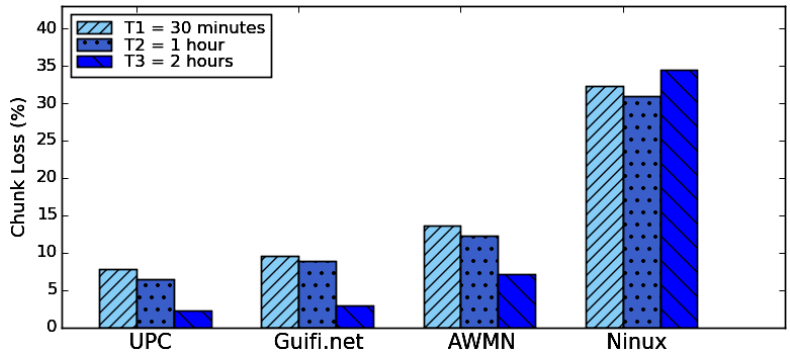


Figure 4.9: Average Chunk Loss

nodes is greater than closer nodes and since the network plays a big role on the delivery of chunks. We also notice that the longer time-frames have on average a better chunk playout because more chunks arrive on time to be displayed (UPC 98%, Guifi.net 98%, AWMN 92%). For the Ninux group we see a more stable chunk playout for each of the time-frames, which means that since the network instability occurs during the whole evaluation the same amount of chunks (on average 71%) arrive to be displayed, also meaning that the network bandwidth/throughput between nodes (on average) is lower than on other CNs and remains constant over time.

Figure 4.11 demonstrates the chunk loss gathered during 30 minutes experiment, with different parameters given to the PeerStreamer. The parameters shown ($TopoKeepBest$, $RTT = 20ms$ and $m = 5$) have been selected in order to predict the behavior and improvements that PeerStreamer can have when executed in CNs. We notice that increasing the RTT for the overlay topology gives the peers higher probability to receive chunks in time and therefore decreasing the chunk loss in each of the groups. The other parameters have a higher impact on losing chunks, especially when the source only sends one copy ($m = 1$) of the chunks to peers (not shown in the figure). We also notice

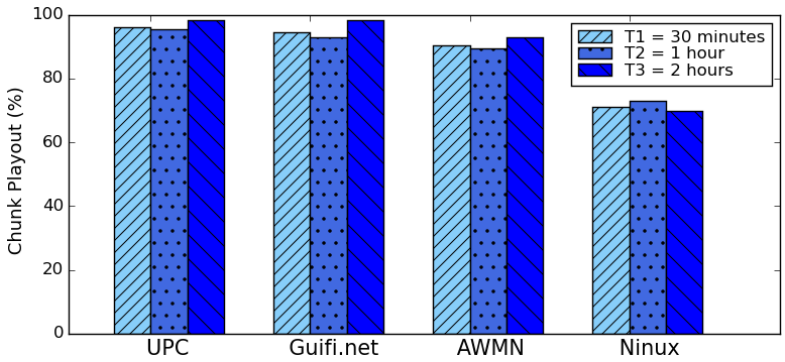


Figure 4.10: Average Chunk Playout

that keeping the best neighbors on topology overlay updates, lowers groups loss chunks (as in UPC case) that have nodes closer to each other, in which the selection of peers for exchanging chunks will have higher probability to choose the best nodes from previous topology updates. For the Ninux group we notice that when keeping the best nodes on topology updates there is a greater improvement (23% in loss, comparing with default parameters where we got 32% loss), because the probability of choosing the best nodes will be higher, since the nodes on this CN have worst connectivity. Also for Ninux, giving a RTT of 20 ms has mostly the same average as the previous experiments (with default parameters) since the nodes are farther apart (in RTT terms), meaning that there will be no significant changes in the neighborhood created for these peers. We also show that there is improvement when changing the number of chunk copies Source 2 sends to peers ($m = 5$). This is because of the resources that Source 2 has at its disposal, which makes it able to send more copies without losing bandwidth and computation time (against Source 1 as a low-power device); and also, since the network has more packet loss than in other CNs, flooding the network with more copies makes a higher probability for peers to be able to receive more chunks on time.

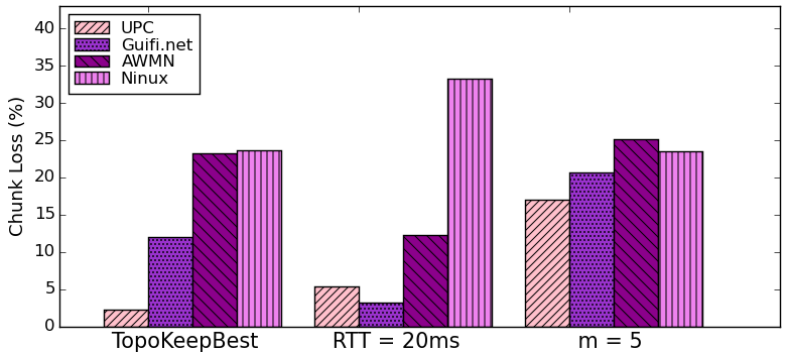


Figure 4.11: Average Chunk Loss with different parameters

4.2.7 Discussion

We started our evaluation by demonstrating the performance PeerStreamer has on CNs, with the default parameters, in order to understand what improvements can be achieved in CNs. We found that PeerStreamer neighborhood selection lacks accountability for network instability and therefore PeerStreamer can perform poorly in CNs. The metric for randomly selecting a subset of peers for the neighborhood reduces the probability to receive chunks in time, since peers can select the worst neighbors. This metric can be good for reducing time spent on initial costs however over time the CN selected peers need to be within either the best or with a greater range in RTT, as shown with different parameters scenario in Figure 4.11. We also found that while modifying the number of chunk copies that the source sends, can have beneficial results, guaranteeing that the chunks will travel to more nodes and be available to be traded in the P2P network over more peers. However, since the wireless links of CNs have a high diversity in bandwidth, this issue can arise and should be studied more thoroughly. Regarding the amount of data exchanged between peers we consider that in current wireless CNs and using P2P networks the high quality video streams (i.e., 1080p) are affected by the

performance of the network links since more data or sizable data needs to pass through the network to the peers, and may even congest it. While using standard quality video streams, as shown in our evaluation the amount of loss is lower and more efficiently exchanged between peers in CNs.

Furthermore, by enhancing the performance of live video streaming, with the opportunity for users to choose the preferable services for them (based on the services' attributes such as RTT) can augment the probability for optimizing the QoE/S in these environments and therefore the combination of our contributions can achieve higher quality of service than an ad-hoc solution.

4.2.8 Summary

An important aspect for the ease of usage of CN micro-clouds is the automatic announcement of services, such as PeerStreamer, and their discovery by other cloud nodes. A service announcement mechanism based on Serf was used to allow end users to discover active PeerStreamer instances in the cloud and join a live streaming event. Furthermore, we design an algorithm to help users choose the service with the better QoS available to them. This was our contribution done on the users perspective, which improves the underlay network. The service discovery and the ease of usage that the Cloudy environment provides for end users, is considered an important element that envisages the users to participate in the streaming service.

On the service level our goal was to have a feasible system that can utilize the resources scattered on CNs in order to achieve a live video streaming service. The part of PeerStreamer that can be modified is the communication between different instances. Our evaluation shows that using PeerStreamer with the default settings can achieve lower rating in terms of QoS in the CN environment where the network instability is prominent. We show that in

different CNs the results obtained in terms of loss of data between peers is distinctive. For this reason, we augment our findings by running PeerStreamer with different configuration of parameters so that we can understand the best behaviour that PeerStreamer can provide to its users. Our evaluation shows that modifying the number of chunk copies that the source sends to peers and modifying the neighbourhood selection policies such as metrics for peer selection as RTT and keep best peers ($TopoKeepBest$) can have beneficial results for live video streaming in the high diversity environment of a CN.

4.3 Conclusion

In this section we approach a monitoring capability for services in micro-clouds in order to gather information that can be applied to optimize the services. The approach for monitoring under CN micro-clouds can be applied to services in order to understand their behavior, and make necessary changes enhancing such services. However, do services have different behavior from data centers or data center cloud infrastructure?

In review, services such as Peerstreamer, can be modified to operate within CN micro-clouds and wireless edge cloud environments, by introducing different configurations. The services deployed in CN micro-clouds also behave differently than in data centers. Thus, their configuration is a necessary step in order to have cloud services running in CN environments.

However, can modifications made on the middleware level, such as the overlay networks, further enhance services in the global network? Does the introduction of social information be an asset to the already gathered information from resources and services?

Chapter 5

Socially-enhanced Overlay Networks in CN Micro-Clouds

In this chapter we investigate the middleware layer, and introduce into overlay networks, as the main communication system, social information. The impact on the routing and message dissemination can be enhanced for services that rely on user interaction, because the knowledge introduced in the overlay network creation aggregates nodes that have more in common, in this case nodes that require the same messages (or data).

In section 5.1 we begin by introducing a publish/subscribe system and reflect the utilization of social information into the positioning of nodes within the overlay network. Moreover, the use of a socially-enhanced overlay network enhances communication and message dissemination, although we exclusively build upon a publish/subscribe system, such a technique can be used for CN micro-cloud services, since services require communication between instances and use overlay networks to achieve its own message routing. In fact, in chapter 4 we discuss about service discovery using SERF, which builds an overlay network to disseminate information about services that are operating within the micro-clouds.

In section 5.2, we introduce the approach for gathering social information in Community networks, and use to build an overlay network that optimizes services message dissemination. The community network does not have a social network such as Facebook, Twitter, therefore the use of community of practices can be made to be similar to social networks. The fact that CoPs share similarities with social networks, indicate that social information can be obtain in a similar way towards understanding the relations of the users, which can be centered in the user cooperation relationship instead of friendship as with Facebook and other social networks.

The use of overlay networks to empower services is a well known technique to optimize service communication, however the use of social information to guarantee nodes or peers that share common interest (i.e. friendship, relationships, proximity to the content) is still an open topic.

5.1 Socially-enhanced Overlay Networks

Publish/subscribe (pub/sub) mechanisms constitute an attractive communication paradigm in the design of large-scale notification systems for Online Social Networks (OSNs). To accommodate the large-scale workloads of notifications produced by OSNs, pub/sub mechanisms require thousands of servers distributed on different data centers all over the world, incurring large overheads. To eliminate the pub/sub resources used, we propose SELECT - a distributed pub/sub social notification system over peer-to-peer (P2P) networks. SELECT organizes the peers on a ring topology and provides an adaptive P2P connection establishment algorithm where each peer identifies the number of connections required, based on the social structure and user availability. This allows to propagate messages to the social friends of the users using a reduced number of hops. The presented algorithm is an

efficient heuristic to an NP-hard problem which maps workload graphs to structured P2P overlays inducing overall, close to theoretical, minimal number of messages. Experiments show that SELECT reduces the number of relay nodes up to 89% versus the state-of-the-art pub/sub notification systems. Additionally, we demonstrate the advantage of SELECT against socially-aware P2P overlay networks and show that the communication between two socially connected peers is reduced on average by at least 64% hops, while achieving 100% communication availability even under high churn.

5.1.1 Overview

One of the fundamental services for Online Social Networks (OSNs) is the real-time delivery of notifications due to users' social interactions. Notifications constitute one of the primary ways social users first learn about the activity of their social friends or their preferable sources (e.g. groups, pages). Twitter users generate on average 8,000 tweets per second[1] which amounts to 500 million of notifications per day from more than 300 million of active users. Thus, large-scale notification systems require to be scalable.

In the case of scalable pub/sub systems, such as Google Cloud Pub/Sub[2], typically massive corporate resources are required to accommodate large-scale workloads of social notifications. Likewise, IBM deploys over a thousand servers on geographically distributed data centers [40] to provide a high-quality pub/sub system. Moreover, with the advent of the Internet of Things (IoT) the number of devices is estimated to reach 20 billion[3] by 2020 [58]. Also, the integration of the IoT with social networks [59, 60] will increase

[1] `http://www.statista.com/`
[2] `https://cloud.google.com/pubsub`
[3] `https://gartner.com/newsroom/id/3598917`

the required computational resources, further motivating research for more advanced pub/sub overlay designs.

The above motivations attracted the attention of both academia [12, 11] and industry [61] to decentralized OSNs (DOSNs) [62] and provide pub/sub systems for OSNs [40] using Peer-to-Peer (P2P) networks [63, 64]. In DOSNs, social users are connected in a P2P network and interact with their social friends using the P2P routing mechanism. However, designing a scalable P2P pub/sub notification system for DOSNs requires four main challenges to be addressed, as follows:

- ***Relay Nodes***: A key characteristic of P2P pub/sub systems [38, 12] is that they leverage a generic overlay network (e.g. DHT, tree, full-mesh) without projecting the social graph in the P2P overlay network. Since social users are not always directly connected in the P2P overlay network, the message dissemination in P2P pub/sub systems relies on peers (also known as *relay nodes*) that may or may not be interested for the message.

- ***High Traffic***: Recent pub/sub systems [40] try to simplify the design of the routing tree and focus on the construction of the P2P overlay network in order to improve the efficiency of message dissemination. Each peer has a bounded number of connections that can be maintained, the selection of which is accomplished without leveraging the social graph and the social interactions. Hence, the generated P2P overlay network presents load balancing problems, where a portion of the peers has high traffic overhead against the rest of the peers, due to the high social interactions and the absence of social integration to the design of the overlay.

- ***Dissemination Latency***: Each peer in the P2P overlay network presents different upload and download bandwidth characteristics. Since each peer has a bounded number of connections, retaining a P2P connection with a poor bandwidth rate increases the dissemination latency that affects the overall performance of the P2P pub/sub system.
- ***Dynamic environment***: It is essential for the success of the OSN to provide a failure resilient P2P pub/sub system with minimum disruption to the communication between social friends. The design of a churn-resistant P2P pub/sub system has been studied in OMen [11]; however, OMen falls short of identifying the online activity of each social user, which poses an additional latency overhead as the establishment of a P2P connection requires a Multi-Path TCP connection [65].

In synopsis, the major issue that arises from current pub/sub implementations is that they are oblivious to specific workloads which result from specific social structure and interactions, as well as, connectivity restrictions (NATs, firewalls, and others).

To address the above challenges, we introduce a fully decentralized pub/sub system for DOSNs, called SELECT. The proposed system organizes peers on a ring topology and exploits both the social graph and the online activity of each social user to establish connections between peers. The intuition behind this is the construction of a P2P overlay network that acts as a substrate for the pub/sub system with the minimum number of *relay nodes* and the minimum communication interruption between social users. Therefore, by harnessing the social network graph we are able to build an overlay in which messages propagate towards the subscribers with minimum relay nodes in between (while relay nodes may also be subscribers). SELECT is the first approach which exploits the small-world properties by embedding it into the overlay

networks' ID space. Thus, peers are placed in the same area based on their social proximity, establishing a bounded and adaptive number of connections to peers. We consider our approach as solving an NP-hard problem, where a P2P overlay is induced from a workload social graph embedded into the identifier (ID) space. Thus, decentralized greedy routing is not only possible but also very efficient and equivalent to routing in navigable small-world networks [66]. In summary, we define the contributions this section as:

- A proposal for a full decentralization of a pub/sub system for DOSNs that exploits both social graphs and online activities of the users. We use Locality Sensitive Hashing (LSH) [67] and Cumulative Moving Average (CMA)[4] to identify which connections allow message propagation with minimum hops, as well as, which peers potentially present better online behaviour over time.

- The description and evaluation of an ID re-assignment process which projects the social graph on a P2P overlay network, minimizing the distance on the overlay networks' ID space.

- The SELECT algorithm, which creates a global overlay network that allows for message propagation with minimum number of hops, taking advantage of peers that present better online behaviour over time instead of relying on random peers. SELECT is adaptive to dynamic environments with the use of novel recovery mechanisms, applying re-routing when it is required.

- An evaluation and analysis by means of simulation and experimentation with real-world data sets, in order to understand the value of each step of the proposed approach and the performance gain in the pub/sub system.

[4] `https://en.wikipedia.org/wiki/Moving_average`

To prove the efficiency of SELECT on pub/sub systems we designed and developed a browser-based P2P pub/sub system[5] using the free and open-source W3C standardized protocol, WebRTC[6]. Based on our implementation we emulated the social behaviour using real-world data sets of Facebook [68], Slashdot [69] and Google Plus [69]. To prove the scalability of our proposed system, we also conducted simulations on a large-scale data set with millions of users collected by Twitter [69]. We show experimentally that this social graph exploitation reduces the number of hops required for dissemination over 64% and the number of relay nodes over 89% against state-of-the-art approaches. Moreover, SELECT maintains 100% communication availability by establishing connections on peers that present better online behavior than other peers. Finally, the peers in the proposed overlay network converge to a stable state in 75% fewer iterations than the state-of-the-art approaches.

5.1.2 Background and Problem Statement

In this subsection, we provide an analysis of P2P networks and overlay construction, relating to the node relay minimization problem and dissemination of information.

5.1.2.1 Peer-to-Peer Networks

A P2P overlay network consists of a set of N peers $\mathcal{P}(|\mathcal{P}| = N)$. The identifiers of the peers are assigned from an ID space $\mathcal{I}$, on the unit interval $\mathcal{I} \in [0 \dots 1)$, using a uniform mapping function (SHA-1). There exists a distance function $d_{\mathcal{I}}(u, v)$ which indicates the distance between peer $u \in \mathcal{P}$ and peer $v \in \mathcal{P}$ in the ID space $\mathcal{I}$. Each peer $p \in \mathcal{P}$ maintains *short-range* links $\mathcal{R}_p^s \subset \mathcal{P}$, that are peers with the minimum distance $d_{\mathcal{I}}(u, v)$ in the ID space $\mathcal{I}$, and *long-range*

[5] `https://github.com/stefanosantaris/SelectDemo`
[6] `https://webrtc.org/`

links $\mathcal{R}_p^l \subset \mathcal{P}$ have been established with a probability inversely proportional to the distance between the peers. The *short-range* links $\mathcal{R}_p^s$ and *long-range* links $\mathcal{R}_p^l$ of each peer p form its routing table $\mathcal{R}_p = \mathcal{R}_p^s + \mathcal{R}_p^l$, where $|\mathcal{R}_p| \ll N$ (usually $|\mathcal{R}_p| = logN$); this is the case in latest P2P models [70] where the optimization stands in minimizing connections instead of establishing new connections to peers.

The lookup query from a peer p to a peer u is routed in a greedy fashion, i.e. peer p selects the neighbor $w \in \mathcal{R}_p$, that minimizes the distance $d_{\mathcal{I}(w,u)}$ in the ID space $\mathcal{I}$, to forward the query. The lookup process forms an h-hop path $p \rightarrow^+ u$ with $h = |p \rightarrow^+ u| \geq 1$. Based on the selection process of the links $\mathcal{R}_p$, P2P overlay networks can guarantee that the h-hop path is bounded in $O(logN)$.

Moreover, peers are heterogeneous in terms of their connectivity characteristics. Different peers present different bandwidth capabilities reflecting in different latency l between peers, and affecting the rate at which a peer can send or receive packets. Therefore, the propagation of messages from peer p to peer u can be given by $l(p,u) = \sum_{i=1}^{h} l_i$.

5.1.2.2 Publish/Subscribe for OSNs

A pub/sub system for OSNs consists of four basic entities: i) a social graph $G = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of social users and $\mathcal{E}$ is the set of social connections; ii) a *Publisher* set $\mathcal{B} \subseteq \mathcal{V}$ of social users that produces the data; iii) a *Subscriber* set $\mathcal{S} \subseteq \mathcal{V}$ that comprises the publishers' social friends and consumes the data; and iv) an interest function $f : \mathcal{S} \times \mathcal{B} \rightarrow \{true, false\}$. A subscriber $s \in \mathcal{S}$ expresses his interest only on the messages that a user $b \in \mathcal{B}$ produces if $f(b,s) = true \wedge (b,s) \in \mathcal{E}$. When a publisher $b \in \mathcal{B}$ posts a new

message, this message needs to be delivered *only* to the interested subscribers $\mathcal{S}_b = \{s \in \mathcal{S} | f(s,b) = true\}$.

A routing tree RT_b is constructed in order to disseminate the message to all the subscribers $\mathcal{S}_b$. Thus, the edges of the routing tree connect social friends' peers that receive and forward the message until all subscribers receive it. However, an edge in the routing tree RT_b does not necessarily correspond to an existing connection in the P2P overlay network. Therefore, the dissemination path latency from a publisher b to all his subscribers $\mathcal{S}_b$ is calculated as follows:

$$l(b, \mathcal{S}_b) = \max_{s \in \mathcal{S}_b} l(b, s) \tag{5.1}$$

Furthermore, the relay nodes $r \in RT_b$ are viewed as the edges of the routing tree RT_b in the path between a publisher and its subscribers. Therefore, depending on the social graph, relay nodes can be subscribers themselves.

5.1.2.3 Problem Definition

The focus of our work is to minimize the number of relay nodes used to disseminate messages on pub/sub systems, as well as to reduce the dissemination latency, even under node churn.

Although the routing tree RT guarantees the dissemination of messages to all the social friends, it suffers from high number of relay nodes. This happens because social users that are connected in the routing tree RT are not necessarily directly connected in the P2P overlay network. Hence, each edge in the routing tree consists of $O(logN)$ relay nodes. Replacing the connections of the routing table $\mathcal{R}_p$ with the subscribers $s \in \mathcal{S}_p$ does not provide guarantees that the h-hop path would be bounded in $O(logN)$, while also not guaranteeing communication between any two arbitrary peers in the P2P overlay network.

We need to ensure that the propagated messages will be reached by all the social user's friends regardless of the structure of the social network.

Therefore, a P2P substrate that minimizes the number of relay nodes in the routing tree RT is required. We define the problem of relay nodes minimization as follows:

Given a publisher b and a set of subscribers S_b, each peer $p \in \mathcal{P}$ aims to establish links in its routing table $\mathcal{R}_p$ such that the routing tree among the publisher b and the subscriber $s \in \mathcal{S}_b$ contains the minimum number of relay nodes $\mathcal{S}_b = \{s \in \mathcal{S} | f(s,b) = false\}$, granting a near theoretical optima minimal solution.

5.1.3 The SELECT System

SELECT aims to construct a global P2P overlay network that establishes connections between peers that host social friends. Moreover, SELECT seeks to organize socially-connected peers in close distance in the overlay network, in order to reduce the number of hops required for the routing process. The intuition behind this is to provide a P2P substrate that reduces the number of hops between two socially-connected peers as well as to maintain the minimum number of relay nodes of the routing tree RT_b for each publisher $b \in \mathcal{B}$. Finally, the goal of SELECT is to provide a pub/sub service that has a low latency impact.

5.1.3.1 System Model

Our system model consists of a set of peers $\mathcal{P}$ and a set of social users $\mathcal{V}$. Social users join the social network either by invitation or they subscribe independently. Each social user $u \in \mathcal{V}$ is mapped onto only one peer $p \in \mathcal{P}$.

Table 5.1: A peer's p local state, listing of local variables for a given peer.

D_p	the peer's p identifier
$\mathcal{R}_p$	a set of peers' identifiers that the peer p is connected
$\mathcal{C}_p$	a set of identifiers of the peers that host the peer's p social friends
$\mathcal{L}_p$	a set of connections that the peer $v \in \mathcal{R}_p$ maintains

We assume that peers communicate with each other over reliable channels (e.g. TCP connections) that bound the number of connections each peer maintains.

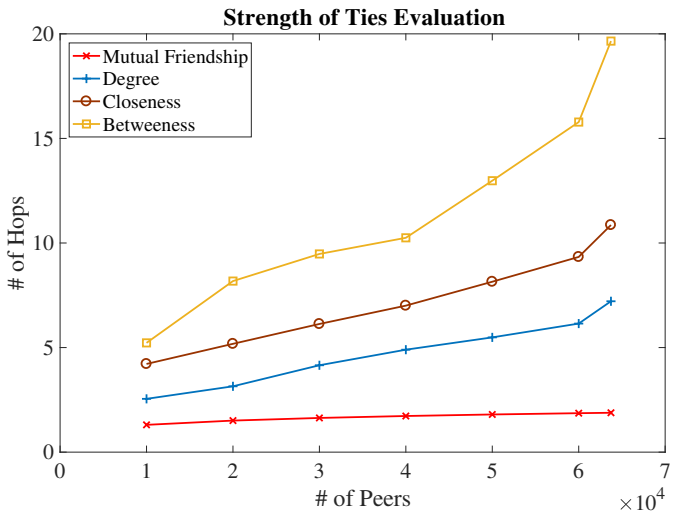


Figure 5.1: Evaluation of the strength of ties between individuals using different centrality measures.

Each peer maintains a set of four local variables listed in Table 5.1. The first variable, D_p is the identifier of the peer p and defines the position of the peer p in the ID space $\mathcal{I} \in [0 \dots 1)$. SELECT seeks to organize the socially-connected peers in close distance in the overlay network. Thus, each peer p modifies its identifier D_p in order to minimize the distance $d_{\mathcal{I}(p,u)}$ to its most important peer u. We measure the importance between two peers and use it as a distance

factor between social users, the strength of ties between two users in the social graph, by using the number of common friends that the two nodes share in the social graph. In Fig. 5.1 we compared how the effect of different centrality measures affect the results of SELECT, however these results can depend on the social network, nevertheless the use of mutual friendship is chosen in respect to the lower estimation (from other centrality measures) for the number of hops between nodes. Therefore, we define the social strength between two peers p and u as follows:

$$s(p,u) = \frac{|\mathcal{C}_p \cap \mathcal{C}_u|}{\mathcal{C}_p} \quad \text{where p,u } \in \mathcal{V} \tag{5.2}$$

The second variable, set $\mathcal{R}_p$, constitutes the routing table of the peer p. The third variable, set $\mathcal{C}_p$, comprises the identifiers of the peers that host its friends in the social graph. The number of connections that each peer establishes is usually lower than the number of friends that each social user maintains in the social network. This is due to the fact that most of the social friends peers have either equal connections to the same friends or a lower number of friends. Thus, in most cases, $|\mathcal{R}_p| \ll |\mathcal{C}_p|$, given that in social networks the number of friends is much higher than the connections that are required. Also, the set $\mathcal{C}_p$ contains only the identifiers of the peers which enhance the lookup process without establishing direct connections to all the peers of the set $\mathcal{C}_p$. The main goal of SELECT is to establish connections with the maximum number of each social users' neighborhood, while minimizing communication with the rest of the social friends by maintaining a minimum number of hops.

The fourth variable, set $\mathcal{L}_p$, contains the identifiers of the peers that each peer $u \in \mathcal{R}_p$ maintains. The existence of this variable is similar to the *lookahead* process of Symphony [64]. This lookahead set enhances the routing process as

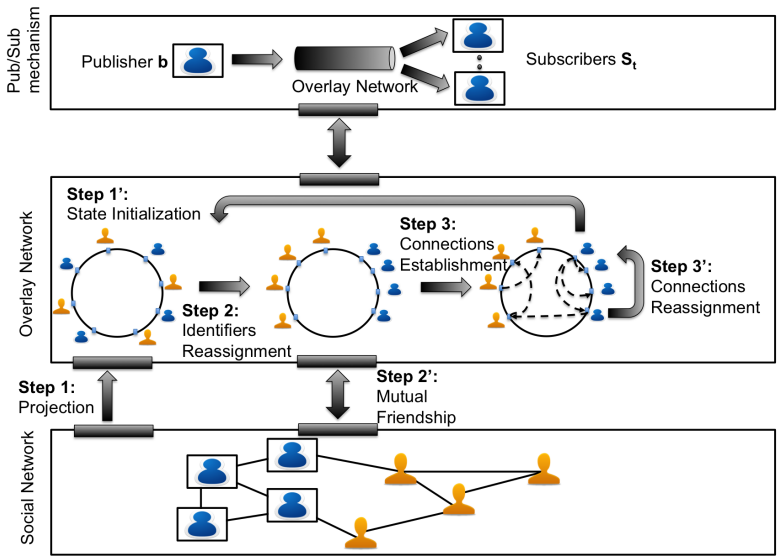


Figure 5.2: The three-layer architecture of SELECT.

it forwards the message to the neighbor that affirms the connection with the targeted peer.

In our model, each peer establishes its own connections to other peers, according to the social networks' friendship mapping. In overall, all peers will lay in a ring topology shared to all peers in order to gain routing performance across the whole network, by minimizing the relay nodes and establishing social routing instead of plain network routing.

5.1.3.2 SELECT System Overview

We implement SELECT using a three-layer architecture, as shown in Figure 5.2, where the bottom layer provides the social network; the middle layer provides a topology construction mechanism, based on each peer's position and its social neighbourhood, that creates the global overlay network; and the top

layer refers to the topology construction protocol where pub/sub mechanism is executed.

The SELECT system consists of three main processes:

- ***Projection***: SELECT associates the position of each peer that hosts a social user in the overlay network (**Step 1** in Figure 5.2). The position of each peer is used to define the distance between two socially connected peers in the ID space $\mathcal{I}$. When a peer joins the overlay network, the local variables of Table 5.1 are initialized (**Step 1'**).
- ***Identifiers Reassignment***: SELECT evaluates the peers position in the ID space and reassigns the identifiers on a round-based basis. Specifically, each peer leverages the social neighborhood information and modifies its identifier in order to reduce its distance in the overlay network with its social friends (**Step 2**).
- ***Peer Connections Establishment and Reassignment***:
  While SELECT organizes the socially connected peers in the same area in the ID space $\mathcal{I}$, each peer establishes direct connections to peers that are also connected in the social network (**Step 3**). A link reassignment process is performed (**Step 3'**) to ensure that each social user communicates with the maximum number of his social friends in the minimum required hops, as well as with the minimum dissemination latency.

Both peers' identifier and connection reassignment processes use a gossip-based peer-sampling methodology to evaluate the topology defined. When the overlay network is constructed, SELECT applies the pub/sub mechanism to construct the routing tree RT_b for the publisher $b \in \mathcal{B}$.

5.1.3.3 Projection and Identifier Reassignment

The projection process (**Step 1** in Figure 5.2) determines the peer's initial position that is perceived by the underlying overlay network. Since social users join the social network either by invitation or by subscription, this directly impacts the peer's initial position (**Step 1'** in Figure 5.2).

As shown in Algorithm 2, the projection of the social user in the overlay network is specified based on the subscription type in the overlay network (line 1). When a social user is subscribed by invitation, his assigned identifier D_p reduces the distance (line 3) between the peer u and the peer p that hosts the invited social user. Otherwise, a random identifier is assigned to the peer p using a uniform hash function (line 5).

As the social network grows, social users create new friendships and the social strength in Equation 5.2 between two users is modified. SELECT strives to reduce the distance in the ID space $\mathcal{I}$ between social friends. In particular, each peer modifies its identifier in order to minimize the distance in the overlay network, to be near the peer's ID which hosts the social friend that has the highest social strength (**Step 2** in Figure 5.2).

The new position choice is the centroid of all its social friends position. However, this does not work in social users with high degree, in which the social strength between friends may significantly differ. Thus, social friends can be located in a totally different position in the ID space $\mathcal{I}$. To address this, we use the centroid between the two social friends that maintain the highest social strength value, as presented in Algorithm 3.

The social strength of each user is calculated using a gossip-based peer-sampling protocol, as shown in Algorithm 4 and Algorithm 5. Every peer p periodically (e.g. every 10 seconds) selects a random social friend u and sends its social

neighborhood set $\mathcal{C}_p$ (line 3 in Algorithm 4). The peer u compares the received neighborhood set $\mathcal{C}_p$ with its neighborhood set $\mathcal{C}_u$ (line 4 in Algorithm 5) and returns the number of mutual friends to the peer p (line 6 in Algorithm 5).

Complexity Analysis : For each peer $p \in \mathcal{P}$, the initial position in the overlay network is calculated in $O(1)$, since the identifier is assigned either uniformly or based on the invited peer's identifier. Thus, the initial projection of the social graph in the P2P topology requires $O(N)$ complexity. The reassignment of the peers' identifiers based on the peer-sampling protocol requires a $O(|\mathcal{C}_p|)$ complexity for each peer, where $|\mathcal{C}_p| \ll N$. In modern social networks usually $|\mathcal{C}_p|$ is on the range of hundreds of social friends, while the size of the network N is billions of users [69]. The total complexity of the Projection and Identifier Reassignment algorithm is

$$O(N \cdot |\mathcal{C}_p|) \tag{5.3}$$

Procedure 2 Peer Identifier Assignment

Input: $v \in \mathcal{V}$ newly registered social user
Output: D_p peer's identifier
1: **if** $\mathcal{C}_v \neq \emptyset$ **then**
2: | u $\leftarrow$ the peer of the social user that invited v
3: | $D_p \leftarrow min_D d_{\mathcal{I}}(u, v)$
4: **else**
5: | $D_p \leftarrow$ UNIFORMHASH(v)
6: **end if**
7: **Return** D_p

5.1.3.4 Peer Connections Establishment and Reassignment

SELECT utilizes a gossip-based peer-sampling service to construct the topology in the overlay network. Each peer periodically acquires its social neighbor's connections in the overlay network and evaluate its current established con-

Procedure 3 Peer Identifier Reassignment

Procedure EVALUATEPOSITION()
    $u \leftarrow$ peer with the highest social strength in $\mathcal{C}_p$;
    $v \leftarrow$ peer with the second highest social strength in $\mathcal{C}_p$;
    $D_p \leftarrow \frac{|d_{\mathcal{I}(u,v)}|}{2}$;
end Procedure

Procedure 4 Peer-sampling - Active Thread

Procedure EXCHANGERT()
    socialFriend $\leftarrow$ getRandomSocialFriendPeer();
    Send $< \mathcal{C}_p, \mathcal{R}_p >$ **to** socialFriend;
    Receive $< nMutual, M >$ **from** socialFriend;
    $socialFriend$.nMutual = nMutual;
    $socialFriend$.M = M;
    $D_p \leftarrow$ evaluatePosition();
    $\mathcal{R}_p \leftarrow$ createLinks();
end Procedure

Procedure 5 Peer-sampling - Passive Thread

Procedure RESPONSEEXCHANGERT()
    Receive $< \mathcal{C}_u, \mathcal{R}_u >$ **from** socialFriend;
    nMutual $\leftarrow |\mathcal{C}_u$.merge$(\mathcal{C}_p)|$;
    $socialFriend$.nMutual $\leftarrow$ nMutual;
    $M \leftarrow \mathcal{C}_u$.constructFriendshipBitmap$(\mathcal{R}_p)$;
    Send $< nMutual, M >$ **to** socialFriend;
    $M' \leftarrow \mathcal{C}_p$.constructFriendshipBitmap$(\mathcal{R}_u)$;
    $socialFriend$.bitMap $= M'$;
    $D_p \leftarrow$ evaluatePosition();
    $\mathcal{R}_p \leftarrow$ createLinks();
end Procedure

nections (**Step 3** in Figure 5.2). Specifically, each peer p seeks to establish direct connections with the maximum number of its social neighborhood $\mathcal{C}_p$.

Each peer is allowed to accept only K incoming links, while maintaining two *short range* outgoing links $\mathcal{R}_p^s$ with his successor and predecessor in the overlay network in order to create a ring topology and K *long range* outgoing links $\mathcal{R}_p^l$ with its social friends. The intuition behind the K incoming links is to avoid having peers that have too many connections, because other peers seek to connect to them, and consequently present more traffic than others. When the K incoming links are established, the peer accepts a new incoming connection if the new connection has better bandwidth capability than the already existing connections. The K outgoing *long range* links are selected by applying the Locality Sensitive Hashing (LSH) technique to the social neighbor's connections retrieved from the peer-sampling service (lines 3-6 and 2-8 in Algorithms 4 and 5, respectively). The LSH technique is used to choose the long range links from different zones of the overlay and avoid link overlap in the overlay network. We consider that the LSH family technique to be reliable in maintaining long range connectivity for the overlay network.

The connection establishment mechanism is shown in Algorithm 6. We begin by indexing the bitmaps of the social neighborhood in $\mathcal{H}$ buckets in the LSH index (lines 2 - 4). In our algorithm, we consider that the number of buckets is equal to the number of *long range* links defined ($|\mathcal{H}| = K$). The reason for selecting $|\mathcal{H}| = K$ buckets in the LSH index is to simplify the selection process of the direct connections. Peers, whose connections are similar, will be indexed in the same bucket. This results in selecting only one peer in each bucket, establishing at most K long range links.

The bitmap of $u \in \mathcal{C}_p$ is an array of size $|\mathcal{C}_p|$, the values of which define the link existence in $\mathcal{R}_u$ between two socially connected peers $u \in \mathcal{C}_p$ and $v \in \mathcal{C}_p$, where $u \neq v$, as follows:

$$bitmap(u,v) = \begin{cases} 1 & \text{if } (u,v) \in \mathcal{R}_u \\ 0 & \text{if } (u,v) \notin \mathcal{R}_u \end{cases}$$

While the bitmaps are indexed in the $\mathcal{H}$ buckets of LSH, in lines 5 - 18, we aim to select one peer of each bucket $h \in \mathcal{H}$ to establish connection. However, not all buckets may contain only one peer, as social friends tend to converge to similar connections. In order to establish connections with the maximum number of the peer's p social neighborhood $\mathcal{C}_p$, but also the minimum dissemination latency, we select the peer that attempts to establish a connection using a picker (line 8), as shown in Algorithm 7. In doing so, we select the peer that achieves the maximum number of social connections and presents better bandwidth capability in order to propagate the message with higher rate. Moreover, we drop an already established connection $(p,u) \in \mathcal{R}_p$ with a peer u that presents similar connections with newly established connection $(p,v) \in \mathcal{R}_p$ (lines 12-16) in Algorithm 5.

Complexity Analysis : The complexity analysis of the *Connections Establishment and Reassignment* algorithm is analogous to the number of social friends $|\mathcal{C}_p|$ that each user maintains and the number of buckets $|\mathcal{H}|$ assigned on the LSH index. The peer-sampling protocol aggregates the bitmaps in $O(|\mathcal{C}_p|)$ complexity. The index of the bitmaps in $|\mathcal{H}| = K$ buckets requires $O(|\mathcal{C}_p| \cdot log(|\mathcal{C}_p|) \cdot K)$ complexity. The selection of the K *long range* links using the LSH index is performed in $O(K)$ cost. Summarizing, the total complexity of the *Connections Establishment and Reassignment* algorithm for each peer $p \in \mathcal{P}$ is

$$O(|\mathcal{C}_p|^2 \cdot log(|\mathcal{C}_p|) \cdot K^2) \tag{5.4}$$

Procedure 6 Peer Links Reassignment

```
1: Procedure createLinks()
2:     for u ∈ C_p do
3:         LSHIndex(u.bitMap);
4:     end for
5:     for h ∈ H do
6:         P_h ← peers assigned in the same bucket h;
7:         if P_h ≠ ∅ then
8:             u ← picker(P_h);
9:             if (p, u) ∉ R_p then
10:                R_p ← (p, u)
11:            end if
12:            for v ∈ P_h, v ≠ u do
13:                if (v, p) ∈ R_p then
14:                    R_p.remove(v);
15:                end if
16:            end for
17:        end if
18:    end for
19: end Procedure
```

Procedure 7 Picker Peer Connection

```
1: Procedure picker(P_h)
2:     PS_h ← sortPeers(P_h)
3:     if (|PS_h| > 0) && (PS_h(0).bw < PS_h(1).bw) then
4:         return PS_h(1)
5:     end if
6:     return PS_h(0)
7: end Procedure
```

5.1.3.5 Pub/Sub system

The pub/sub system utilizes the generated overlay network to create the routing tree RT_b for each social user $b \in \mathcal{B}$ and guarantees the delivery of the published messages to all of his social friends $\mathcal{S}_b$.

Following the lookahead technique of [64], each peer p maintains a lookahead set $\mathcal{L}_p$ of connections that each peer $u \in \mathcal{C}_p$ maintains. Peer p uses the lookahead set $\mathcal{L}_p$ to create the routing tree RT_b and forward the message during the routing process. Each peer monitors its routing table RT_b and the lookahead set $\mathcal{L}_p$ and forwards the message to the peer that guarantees the delivery of the message within 1 or 2 hops. If the peer $u \in \mathcal{S}_b$ is not included in the routing table $\mathcal{R}_p$ and the lookahead set $\mathcal{L}_p$, the peer $v \in \mathcal{R}_p$ that minimizes the distance $d_{\mathcal{I}}(v, u)$ is selected.

5.1.3.6 Recovery Mechanism

Peers join and depart the overlay network at unexpected rate (churn). Also, to maintain the pub/sub reliability property for message delivery, we need to manage a routing table that efficiently recovers from peers departure. In doing so, peers periodically request each social friend of their routing table for their state. The availability of each peer is recorded and their online behavior is calculated using the Cumulative Moving Average (CMA). The intuition of using CMA is to identify the average online behavior of a social user during the period of the last few days and ensure that the social user is a good candidate for establishing a connection. Thus, the peer identifies if a connection is unresponsive because a social user is mostly offline or if it is a temporal connection failure. In doing so, a peer p decides to keep an unresponsive connection to the peer u in order not to create a chain of connections reassignment to the peers that are connected to the peer p. In contrast, when a peer is unresponsive and its CMA value is low, we replace the unresponsive peer with another peer from the same bucket of the LSH index (see Section 5.1.3.4). Using this approach, SELECT maintains direct connections with peers that host social friends and publish a message on non-relay nodes while also being adaptive to dynamic environments.

Table 5.2: Four real-world data sets of social networks that include users information, such as social connections and average degree.

Data Set	Users	Connections	Average Degree
Facebook	63,731	817,090	25.642
Twitter	3,990,418	294,865,207	73.89
Slashdot	82,168	948,463	11.543
GooglePlus	107,614	13,673,453	127

5.1.4 Evaluation

For our evaluation, we considered two types of experiments, one as a simulation and another as a realistic environment. For the simulation experiments, we used the Gelly Graph API[7] which runs over the Apache Flink[8] distributed data processing framework. We ran our experiments on a Flink cluster with 20 nodes in order to provide a distributed discrete event simulator suitable to conduct large-scale experiments with millions of peers. For the realistic experiments, we used WebRTC to create the peers as browser-dependent and deployed on a cloud infrastructure several VMs (in total 18 VMs are used). The VMs contained several peers spread among each, hosting all the users in each data set. The communication between peers was done through the network interface, which allowed to emulate the latency between nodes and achieve a realistic environment.

The implementation of SELECT is performed using the vertex-centric iterative model [71]. Specifically, in synchronized iteration steps, each peer produces messages to other peers and updates their identifiers and their connections in the overlay network using the SELECT algorithms.

[7] `https://ci.apache.org/projects/flink/flink-docs-master/dev/libs/gelly/index.html`

[8] `http://flink.apache.org/`

Experiments are performed in evolving networks, where users join the overlay network at different phases. We initiate our experiments by selecting a social user $u \in \mathcal{V}$ from the data set at random. Thereafter, we insert into the social network a portion of the user u's social friends, following the model of [72]. Based on [72], social users establish friendship connections at high rate in the beginning of the join process, and this rate decreases exponentially over time. Therefore, at each iteration step, we select a registered social user and insert into the social graph a number of her social friends that preserves the exponentially decreasing rate of the model.

Additionally, we introduced the churn rate of each peer in the overlay network, following the model of [73]. Specifically, at each iteration step, we select a number of peers based on a log-normal distribution to be excluded from the overlay network. When the iteration step is completed, the removed peers are recovered in the overlay network.

When the overlay network is constructed, we perform simulations of the pub/sub mechanism to measure the number of relay nodes that exist on each routing tree. In order to realistically simulate a real-time notification system in the social network, each publisher posts messages at exponential rate following the model of [74].

The realistic experiments follow the same pattern as the simulation experiments, however since each node has different bandwidth capabilities, different latency is applied for each node and accounted in the analysis. Also, in the pub/sub system, packets of 1.2MB are sent from the publishers to the subscribers.

5.1.4.1 Data sets

Our evaluation is performed with four real-world data sets, listed in Table 5.2. These data sets cover a wide range of social graph features, from less connected

graphs (Slashdot [69], Facebook [68]) to highly connected graphs (Twitter, Google Plus [69]), that enhance the evaluation of our proposed approach on several graph types. Moreover, we conduct experiments on the large-scale data set of Twitter in order to demonstrate the scalability of our algorithm. The characteristics of the data sets are presented in Table 5.2.

5.1.4.2 Metrics

In order to measure the efficiency of SELECT, we use the following metrics:

- **Number of Hops:** The average number of overlay hops within the path between two peers.
- **Number of relay nodes:** The average number of relay nodes that exists in the pub/sub routing tree.
- **Number of iterations:** The average number of iterations required to organize the peers in the overlay network.
- **Percentage of messages:** The percentage of messages that each peer forwards in the dissemination tree.
- **Latency:** The average latency of communication between peers in the overlay network, counting the latency between intermediate peers in a given path. Only used for the realistic experiments, since simulations do not account with latency.

To validate our analysis, for each metric we report the average result out of 100 independent trials to decrease the risk of statistical error. We consider these metrics to be important to understand the behaviour of SELECT and the pub/sub system. Thus, be able to compare the end results with other

works while also giving feedback on the use of SELECT for the domain of pub/sub systems.

5.1.4.3 Simulation Experiments

We compared SELECT with several existing pub/sub systems of different categories: i) a pub/sub system over the *Symphony* P2P overlay network without any further modification on the P2P topology; ii) *Bayeux*, a pub/sub system that organizes peers into a DHT in a P2P overlay and builds a spanning tree for each topic to propagate the messages; iii) *Vitis*, a gossip-based pub/sub system that organizes the subscribers into clusters; and iv) *OMen*, that constructs TCOs to disseminate information on each topic.

As the number of direct connections increases, we observe a substantial reduction, over 90%, on the average number of hops required for the communication between two socially-connected peers. However, as the number of links used overcomes the logarithmic number of peers in the overlay network, no further improvement is performed. Based on the above observation, for the rest of the experiments, we assign $log_2 N$ direct connections on each peer in order to construct a P2P topology.

Figure 5.3 presents the average number of hops required for a publisher to propagate information to each one of his subscribers. As the network grows, the average number of hops increases logarithmically. However, SELECT performs with 76%, 83%, 75% and 85% fewer hops compared to the pub/sub mechanism built over the Symphony overlay network and for the Facebook, Twitter, Google Plus and Slashdot data sets, respectively. This occurs due to the fact that Symphony's construction of long range links is completely oblivious to the social graph and the publication workload. In contrast, SELECT establishes connections between socially-connected peers, and as

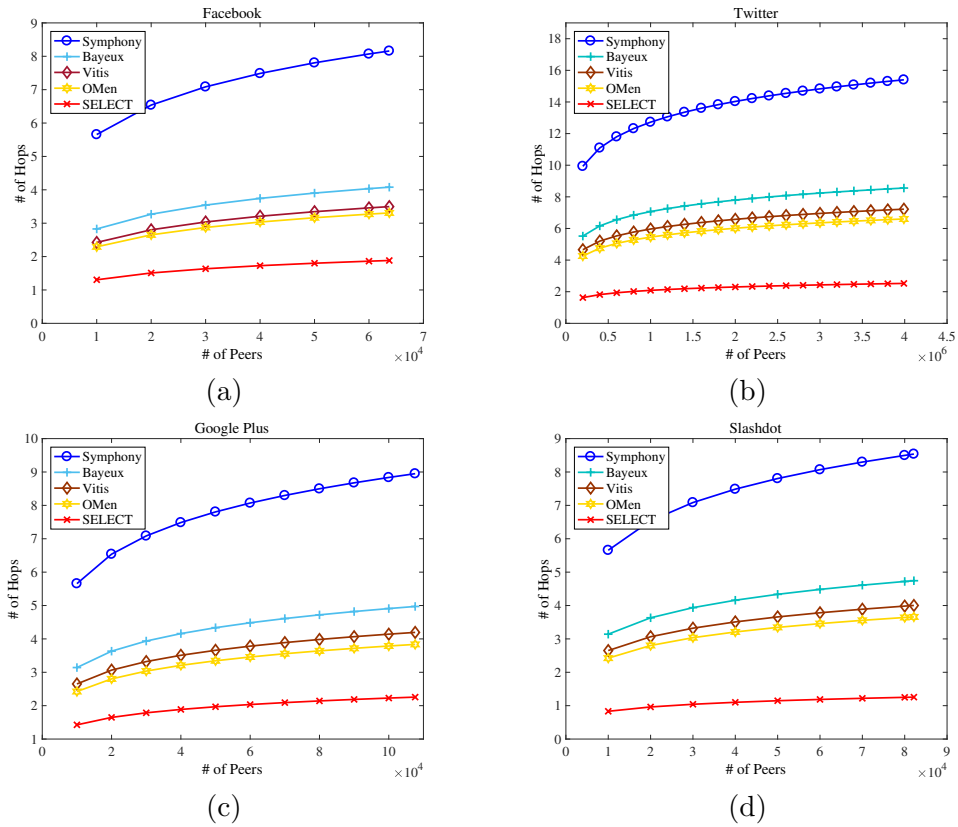


Figure 5.3: Number of hops per social lookup for the (a) Facebook, (b) Twitter, (c) Google Plus and (d) Slashdot data sets.

such subscribers are 1 or 2 hops away from the publisher. Compared to the state-of-the-art pub/sub approaches, SELECT achieves more than 43%, 61%, 41% and 65% reduction for the Facebook, Twitter, Google Plus and Slashdot data sets, respectively. This happens because peer identifiers on SELECT are mutable and socially-connected peers are clustered in the same region in the ID space. Hence, a small-world network is accomplished on SELECT, in

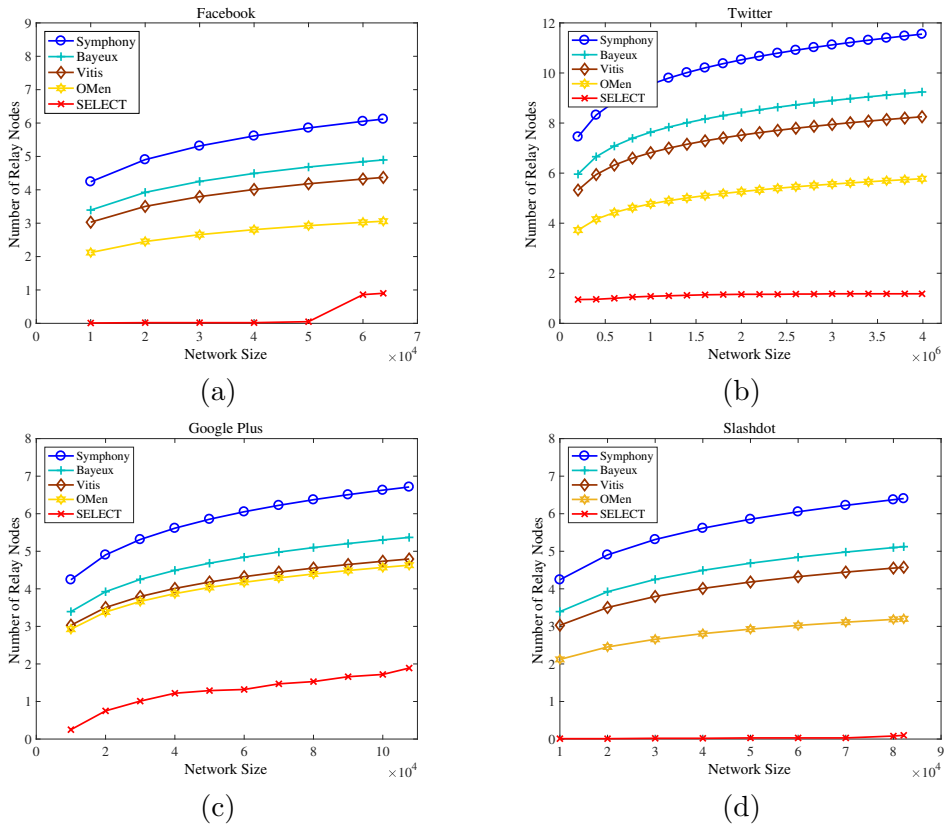


Figure 5.4: Number of relay nodes per pub/sub routing path for the (a) Facebook, (b) Twitter, (c) Google Plus and (d) Slashdot data sets.

contrast to the presented approaches where an immutable identifier policy is applied.

Figure 5.4 presents the impact of SELECT on the number of relay nodes that exist in the routing path between publisher and subscriber. SELECT presents over 98% reduction on the number of relay nodes for all data sets, in comparison to the Symphony, Bayeux, Vitis and OMen approaches. This happens because in Symphony, Bayeux, Vitis and OMen the probability of

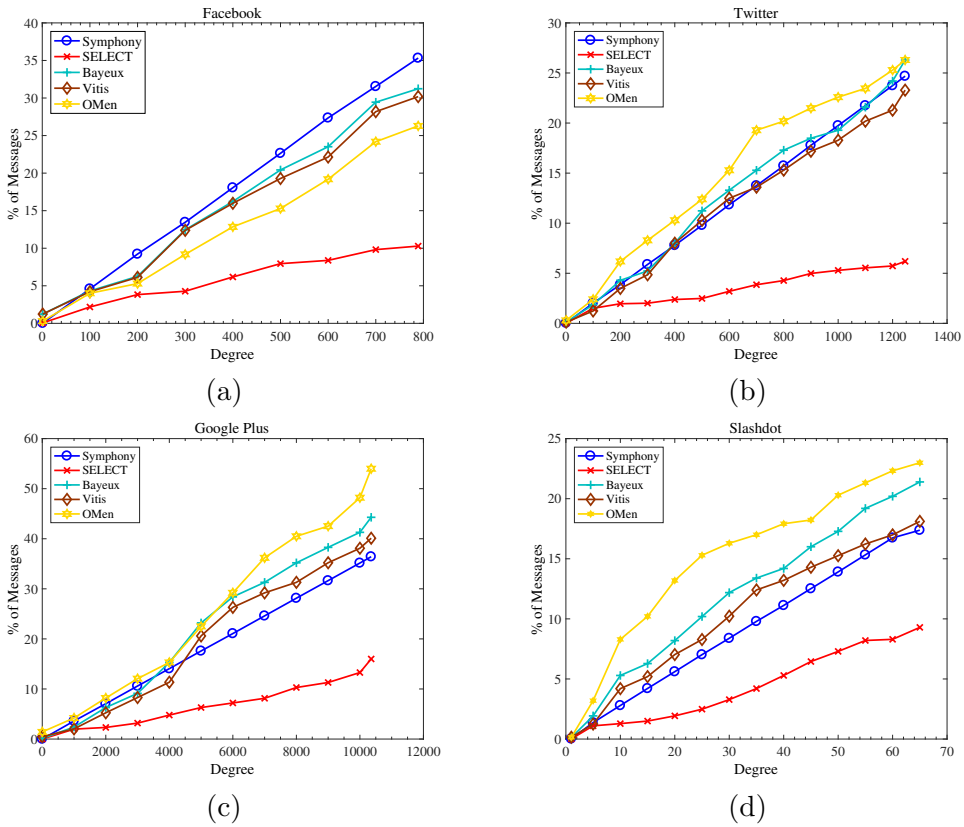


Figure 5.5: Messages forwarded per social degree in a pub/sub routing tree for the (a) Facebook, (b) Twitter, (c) Google Plus and (d) Slashdot data sets.

two socially-connected users to be also connected in the overlay network is extremely low. In contrast, SELECT leverages the social graph and establishes connections between socially-connected peers that reduces the number of relay nodes in the routing path between publisher and subscriber.

In Figure 5.5, we investigate the balance of the load that each peer presents, by measuring the percentage of messages that each peer forwards in the routing tree against the degree of the peer. Figure 5.5 indicates that SELECT

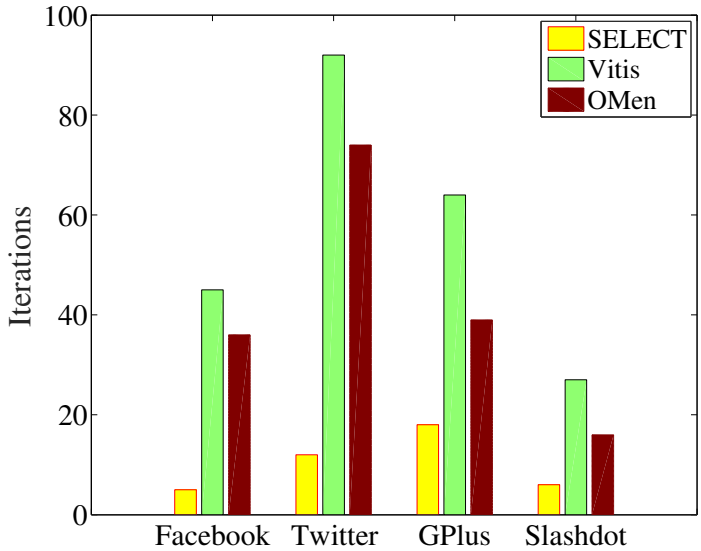


Figure 5.6: Number of iterations required to construct the overlay. Symphony and Bayeux approaches are excluded as they provide no iterative connection establishment process.

provides better load balancing than Symphony, Bayeux, Vitis and OMen approaches. This happens because Symphony and Bayeux are agnostic to the social network dynamics, and thus information propagation converges to the peers that present high social degree. In contrast, Vitis and OMen leverage the social network dynamics but the peer connection strategy that they follow emphasize on connecting peers with high social degree. SELECT presents more than 60%, 73%, 56% and 46% improvement against Symphony, Bayeux, Vitis and OMen approaches for the Facebook, Twitter, Google Plus and Slashdot data sets, respectively.

The total number of iterations required to establish the connections between peers, are presented in Figure 5.6. Symphony and Bayeux are excluded from

this set of experiments as they provide no iterative algorithms. Based on the reporting results in Figure 5.6 we observe that SELECT converges in significantly lower number of iterations than Vitis and OMen. This observation is due to the fact that Vitis and OMen initially organize the peers following a standard DHT-based overlay network and optimise the connections when the overlay network is formed. Thus, connections are established between non socially-connected peers and the gossip algorithm applied requires more iterations in order to identify the socially-connected peers. In contrast, SELECT establishes immediately the connections between peers that are socially-connected and thereafter optimises the connections in order to improve the information propagation. This results in a lower number of iterations to organize the peers since most of the peers' connections are already to a socially-connected peer.

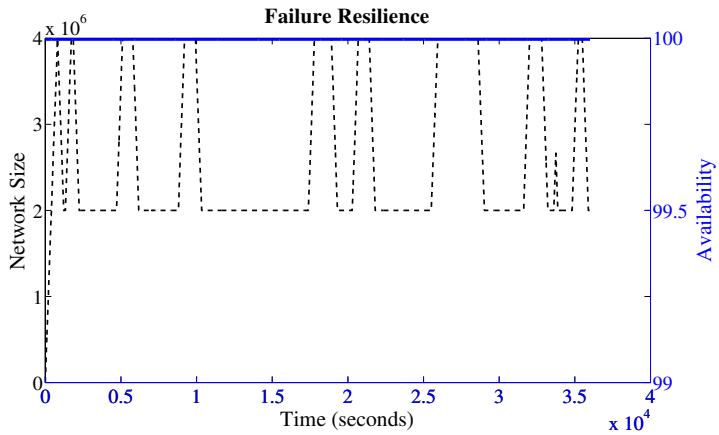


Figure 5.7: The impact of churn in the data availability during the information propagation. Dash line represents the node churn and continuous line the availability.

Finally, in Figure 5.7, we present the impact of the unexpected join and leave of peers in the relay nodes between two socially-connected peers. In this set

of experiments we ran a simulation for over ten hours, where in each second a random number of peers depart or join the network. The total number of peers that are available in the P2P overlay network cannot be less than half of the overall social network. Based on this experiment, we observe that each peer efficiently replaces the unresponsive connection with another peer that presents similar connections based on the LSH index. Thus, the routing process maintains 100% data availability in all data sets.

5.1.4.4 Realistic Experiments

In the realistic experiments, we perform the comparison with other pub/sub systems, as Symphony, Bayeux, Vitis and Omen, as described in the previous experiments.

Towards understanding the behaviour of our algorithm when latency is applied, we start by introducing an initial experiment on simultaneous connectivity. The peers join a network and connect to a central peer, without applying any selection algorithm. Thus, the central peer is connected to all others. Afterwards, the central peer creates a data fragment of 1.2MB (average image size) and sends to all its connections simultaneously. In our findings when increasing the number of connections there is a linear increase in the total time for transfers. Therefore, we can establish that an issue is not the number of connections to be established, but the simultaneous transfers to peers.

Figure 5.8 presents the latency for message dissemination between the publishers and their subscribers. At first, without selection algorithm (random), for each of the data sets we find that the peers connectivity can grow exponentially making the dissemination process costly in terms of timing. When applying SELECT, the overlay becomes latency aware and therefore the dissemination

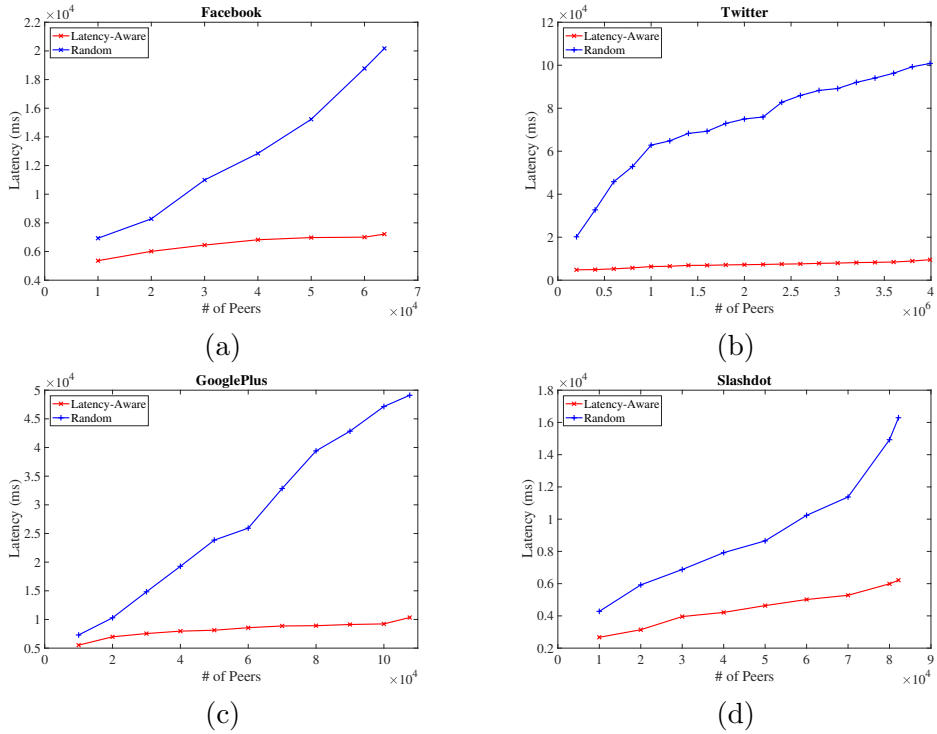


Figure 5.8: Average latency of data dissemination in the pub/sub routing tree for the (a) Facebook, (b) Twitter, (c) Google Plus and (d) Slashdot data sets.

latency has a small *linear growth* accommodating more peers in the overlay without sacrificing dissemination time.

Figure 5.9 presents the distribution of the identifiers after applying SELECT, for each of the data sets. We determine that SELECT rearranges the overlay in such a way that the nodes distances are maintained as low as possible while still being able to reach all of the network. In fact, we can observe that small groups of nodes are within the same regions, which aggregate the socially-connected nodes without losing connectivity between regions.

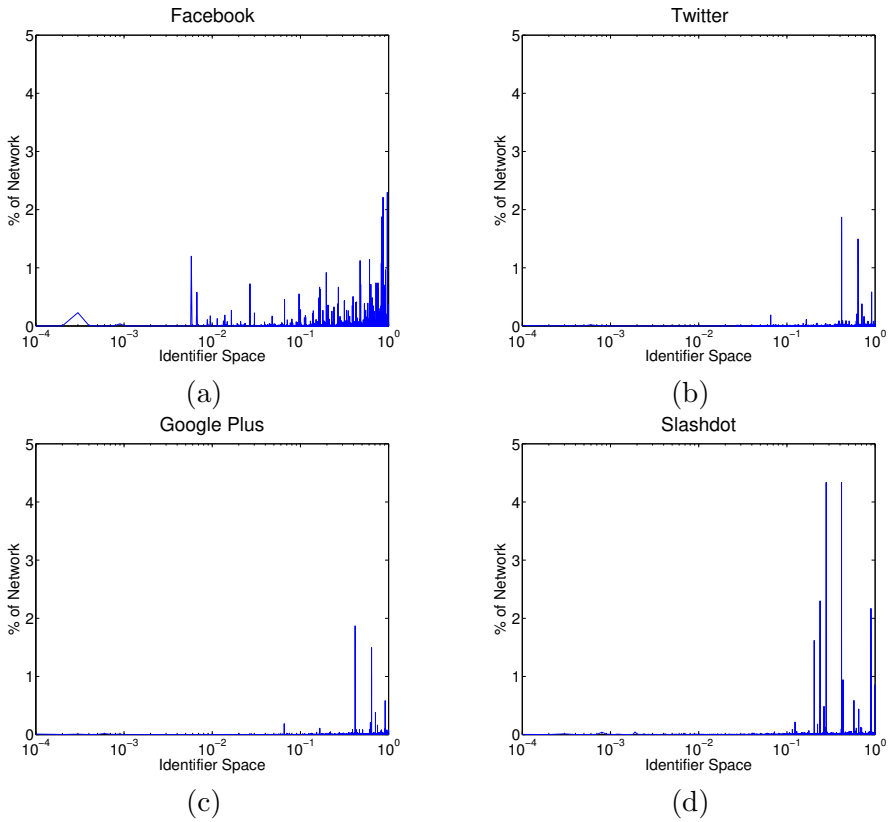


Figure 5.9: Identifiers distribution among the network for the (a) Facebook, (b) Twitter, (c) Google Plus and (d) Slashdot data sets.

5.1.5 Discussion

Our approach to disseminate data in pub/sub systems relies on the social network connections. Due to the fact that state-of-the-art approaches rely on different aspects of the network for their optimization, it is hard to provide a fair cost comparison between them and the SELECT algorithm. We clearly see from our experimental results that the actual costs come from the added

social information which is necessary to create the friendship graph in order to augment the global overlay.

We assure the correctness of our approach by grounding it in a ring topology, since it gives us the ability to continue sending messages to all peers and guarantee that all nodes are able to receive them. Other topologies, such as mesh, tend to create isolated communities of nodes. Therefore, the use of other topologies may not guarantee the same results when applying different social networks.

We also show that the issue of simultaneous data transfers may degrade the performance of a peer when disseminating concurrent messages. This issue can be optimized by having more than one paths to the subscribers in order to guarantee the transmission; however, it is unlikely to find paths of the same length and latency stability.

Finally, we can observe that SELECT achieves its uni-dimensional network construction in real world environments very successfully and without compromising any of the required large-scale pub/sub properties. This proves that SELECT is fully applicable on OSNs in real world settings, although a geographically distribution study would augment our findings.

5.1.6 Summary

In this section, we address the problem of relay nodes in a pub/sub system for social notifications and our solution comprises the creation of SELECT - a distributed pub/sub system. We design a P2P overlay network that exploits the social graph to organize the peers in an overlay network and establish connections between socially-connected peers. Using a gossip-based peer sampling service, SELECT reduces the number of hops required to communicate two socially-connected peers.

Additionally, the constructed routing trees in the pub/sub system exhibit the minimum number of relay nodes. We evaluate SELECT in simulated and realistic environments using four real-world data sets and highlight the performance of SELECT against state-of-the-art approaches. Modern social networks, such as Facebook, Twitter and Spotify, have to propagate a vast amount of notifications. Consequently, to account for the fact that such notification systems need to offload processing from their dedicated resources it is worth to consider the implementation of SELECT that reduces the number of relay nodes, while maintaining 100% communication availability.

However, how can we integrate SELECT in CN micro-clouds? Are there explicit relations between people that allows to have the same information as with social networks?

5.2 Socially-aware Micro-Cloud Services in Community Networks

Community networks are a growing network cooperation effort by citizens to build and maintain an Internet infrastructure in regions that are not available. Adding that, to bring cloud services towards community networks, micro-clouds were started as an edge cloud computing model where members cooperate with resources. Therefore, enhancing routing for service communication in CNs is an attractive paradigm which benefits the infrastructure. The problem faced is a growing consumption of resources for dissemination of messages in the community network environments. This is due to the fact that services build their overlay networks oblivious to the underlying workload patterns which arise from social cooperation in community networks. Furthermore, CNs do not have an explicit social network or social interactions. Therefore, the use of Community of Practice (CoPs) as the social information would grant the

required social information to understand the cooperation that exists within CN micro-clouds.

In this section we induce SELECT with the information that comes from CoPs which enhances the creation of overlay networks for CN micro-cloud services. Social information is based on the cooperation within community networks, by exploiting the community of practice social aspects.

Our work, organizes the peers in a ring topology and provide an adaptive P2P connection establishment algorithm where each peer identifies the number of connection needed, based on the social structure and user availability. Experiments show that SELECT reduces the number of relay nodes up to 89% using the CoP information versus state-of-the-art pub/sub notification systems given as baseline, using social networks information.

5.2.1 Overview

Community networks can be viewed as community of practice (CoPs), where users collaborate to fulfill common goals. In large networks, such as Guifi.net with more than 35.000 nodes, collaboration is done mostly within areas, or groups of people instead of the entire network. Also, these networks present challenges to the members, in which members contribute towards collaborative goals, such as adding new devices (antennas, routers) to increase network capacity, or by adding new services to the network (Internet proxies, FTP, camera videos)[9].

Community of practice (CoP) is a common form of people gathering and completing tasks towards a common goal [75]. The perspective of collaboration is an important step in developing ideas or infrastructures that support said

[9] `http://guifi.net/node/3671/view/services`

goals. In this respect, community networks are then viewed as CoP applications when it comes to setup network devices, augmenting the network abilities or even supporting new cloud services at micro-cloud levels. CoP can also give a perspective of social interaction between members of the network, which can help towards optimizing the network routing and infrastructure built by the members.

The idea of edge cloud computing brings forward the micro-clouds in community networks, in such that, each area of the network collaborate to minimize services requirement to outside resources. To further attempt to optimize such solutions, we see an adaptation for service deployment according to network properties [76].

To complement the CN micro-clouds, we need to take into account the social behaviour of the members, in which the CoP takes its role as the compulsory environment towards understanding how people behave in the network and the expected behaviour of the services in respect to the social properties provided. The level of interaction, and collaboration between people need to be respected and transpose towards the service behaviour in order to have a fair system and to motivate further interest in the services.

We find that building CN micro-clouds can be a solution to minimize the dependency to the outside network (Internet) within CNs, which is the majority of the traffic. In fact, with the broader use of micro-clouds such networks can minimize the Internet interaction, favouring services that are already inside, diminishing the traffic towards the Internet. Issues also arise as the CN micro-cloud services come into focus, such as their performance with constrained devices; the communication latency in both networks; or the motivation towards using newly created services over well-established Internet services. We can also establish that service routing is an important step towards

optimizing services, while making them attractive to members. However, such routing is only made by looking towards the network itself, disregarding the social interactions.

This work comes to aid in the optimization of overlays within CN micro-clouds, by utilizing CoP information to build and optimize the relay nodes when dissemination of information is done. Services within the CNs micro-clouds are mainly focus on a P2P dissemination and run with constrained devices [77, 21]. Therefore, with the use of CoP information as the social support we aim to build overlay networks that prioritizes information according to how users contribute in the network. Nevertheless, all nodes should be guaranteed that they can receive their fair share of the services. The resulting overlay created through SELECT in CN makes the dissemination of data through the nodes that contribute more, without losing those that contribute less.

By applying SELECT in CN on services in micro-clouds, we need to consider that each node (or user) contributes to the community in varied ways, such as, with network links, devices, or capital to construct area antennas or faster links. However, we see that each user may not be concerned in utilizing all services available in the micro-clouds in the same way, therefore, we can establish contribution as a proponent towards data dissemination of each service.

In summary, we define our contributions as:

- A proposal for an optimization of service overlays within CN micro-clouds, that exploits both social graphs and cooperation between members, by making use of community of practice information as the main source for relationships between users.
- The description and evaluation of the SELECT algorithm, that projects the social graph on P2P overlay network, minimizing the distance on the

overlay networks' ID space, when using CoP information as alternative to social networks.

- An evaluation and analysis by means of simulation environment, in order to understand the value and viability of using CoP information for enhancing the service overlays in micro-clouds.

To prove the viability and scalability of our proposed system, we used as baseline the results from large-scale data sets with thousands to millions of users collected by Facebook and Twitter used in section 5.1.4. We show experimentally that this social graph exploitation reduces the number of hops required for dissemination over 64% and the number of relay nodes over 89% against state-of-the-art approaches. Moreover, we compare our previous results with experiments with CoP information gathered from CNs, where we see that the results are in the same range as with social network information, and thus adding its viability to use within CN micro-clouds.

5.2.2 SELECT in CN System

SELECT in CN aims to construct a global P2P overlay network that establishes connections between peers that host social connections, to be used within CN micro-clouds. Moreover, SELECT in CN seeks to organize socially-connected peers in close distance in the overlay network, in order to reduce the number of hops required for the routing process. The intuition behind this is to provide a P2P substrate that reduces the number of hops between two socially-connected peers as well as to maintain the minimum number of relay nodes of the routing tree RT_b for dissemination of data. Finally, we aim to improve service overlays networks within CN micro-clouds while having a low latency impact.

5.2.2.1 Community of practice

The information gathered from CoP differs from usual social networks in which instead of relationships between users (as friends, friends of friends) it is used the concept of interactions between people (cooperation between members). We exploit the mailing lists (as the alternative to social networks) to establish interactions between people and understand how people cooperate. Therefore, by establishing the relations as the cooperation of the users, it is an easier process to identify main users, users that cooperate more or less with others.

The mailing list information was gathered from the Guifi.net mailing lists[10]. We assume that the mailing lists are used exclusively for cooperation within the network and network enhancement. Thus, users interact with each other in a variety of cooperation projects, in examples we find the installation of new routers/devices, and services to be used by the community as topics on the mailing lists. From such lists we identified the users (by email) and crossed referenced all the posts in order to establish common links between users. Therefore, establishing the strength of cooperation for each user, when they appear in different threads. Also, the lists are mainly separated for geographically separate locations, and thus we can add users that collaborate between different locations, and those that only cooperate within the same region. Other types of CoP should be usable as long as relations can be established between members, such is the case of real-life meetings, in which members come together to discuss and even deploy devices in the field.

The number of users found are a small percentage of the total users of the network, since only some of the people cooperate with others by using mailing lists. We find that most people tend to install their devices without much

[10] `https://llistes.guifi.net/sympa`

guidance and let the network itself automatically configure routing and cooperation with other devices. Parts of the network also use real life meetings for cooperation between members.

However, we can say that CoP over CNs follow the small world properties, where clustering coeficient is not small while the distance between nodes grows logarithmically.

The detailed explanation of the Select algorithm is in subsection 5.1.3. The modifications made on the system were exclusively in the social information gathered. Thus, instead of using the strength of ties between users, we focus on the cooperation between members of the CNs.

By using the mailing lists of community networks, we gather each pair of members that communicates with each other, strengthening their relation when found in multiple threads $\mathcal{N}_t$. Therefore, we define the social strength between two peers p and u as follows:

$$s(p,u) = \frac{|\mathcal{C}_p \cap \mathcal{C}_u| * \mathcal{N}_t}{\mathcal{C}_p}, p, u \in \mathcal{V} \tag{5.5}$$

As CoP grows, social users interact more often and the social strength in Equation 5.5 between two users is modified. The goal is to reduce the distance in the ID space $\mathcal{I}$ between social users. As such each peer modifies its identifier and moves closer to the peer that hosts a collaboration (social peer) with higher social strength.

5.2.3 Evaluation

For our evaluation, we consider previous experiments done with SELECT and social networks data sets, such as Facebook and Twitter, as baseline and the experiments with Guifi.net mailing lists as the community of practice information. The simulation experiments, were done with the use of Gelly

Graph API[11] running over the Apache Flink[12] distributed data processing framework. The experiments were run on a Flink cluster with 20 nodes in order to provide a distributed discrete event simulator suitable to conduct large-scale experiments with thousands to millions of peers.

Experiments are performed in evolving networks, where users join the overlay network at different phases. We initiate our experiments by selecting a social peer $u \in \mathcal{V}$ from the data set at random. Thereafter, we insert into the social network a portion of the user u's relationships, following the model of [72]. Therefore, at each iteration step, we select a registered social user and insert into the social graph a number of her social peers that preserves the exponentially decreasing rate of the model. The use of CoP graph, is done in the same manner as with social networks, in which relationships are added to the users with each iteration.

5.2.3.1 Datasets

Our evaluation is performed with three real-world data sets, listed in Table 5.3. These data sets cover a wide range of social graph features, from less connected graphs as Facebook [68] to high connected graphs as Twitter[69], which enhance the evaluation of our proposed approach on several graph types. Moreover, we conduct experiments on CoP-Guifi data set, based on the mailing lists information gathered. The characteristics of the data sets are presented in Table 5.3.

[11] `https://ci.apache.org/projects/flink/flink-docs-master/libs/gelly_guide.html`

[12] `http://flink.apache.org/`

Table 5.3: Four real-world data sets of social networks, that includes users information such as social connections and average social degree.

Data Set	Users	Connections	Average Degree
Facebook	63,731	817,090	25.642
Twitter	3,990,418	294,865,207	73.89
CoP Guifi	3016	16,471	10.9

5.2.3.2 Metrics

In order to measure the efficiency of SELECT in CN, we use the following metrics:

- **Number of Hops:** The average number of overlay hops within the path between two peers.
- **Number of relay nodes:** The average number of relay nodes that exists in the pub/sub routing tree.

To validate our analysis, for each metric we report the average result out of 100 independent trials to decrease the risk of statistical error. We consider these metrics to be important to understand the behaviour of SELECT in CN when different social information is used, and be able to compare the end results with other works while also giving feedback on the use of SELECT in CN for the domain of community networks. The metrics reflect the efficiency of the overlay network, however it does not account with the underlay network effect.

5.2.3.3 Simulation Experiments

We compared our results with several existing pub/sub systems of different categories: i) a pub/sub system over the *Symphony* P2P overlay network

without any further modification on the P2P topology; ii) *Bayeux*, a pub/sub system that organizes peers into a DHT P2P overlay and build a spanning tree for each topic to propagate the messages; iii) *Vitis*, a gossip-based pub/sub system that organizes the subscribers into clusters; and iv) *OMen*, that constructs TCOs to disseminate information on each topic.

As the number of direct connections increases, we observe a substantial reduction, over 90%, on the average number of hops required for the communication between two socially-connected peers. However, as the number of links used overcomes the logarithmic number of peers in the overlay network, no further improvement is performed. Based on the above observation, for the rest of experimentation, we assign $log_2 N$ direct connections on each peer in order to construct a P2P topology.

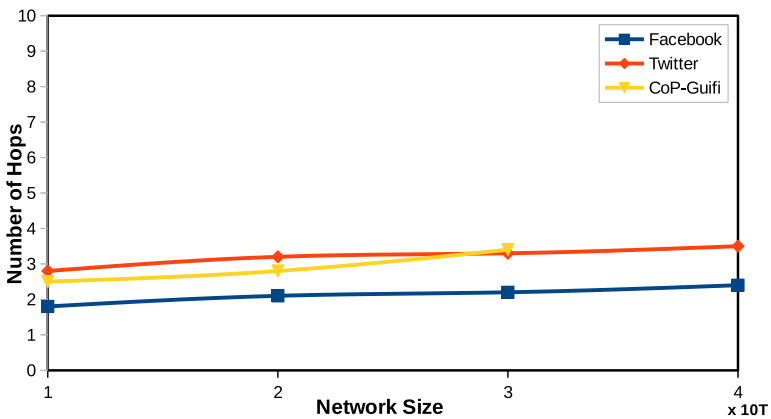


Figure 5.10: Comparison of Number of hops per social lookup obtained with the use of Facebook, Twitter data sets and CoP-Guifi information.

The results taken as baseline in Figure 5.3(a) and (b), explained in more detail in subsection 5.1.4 presents the average number of hops required for a publisher to propagate information to each one of his subscribers. As the network grows,

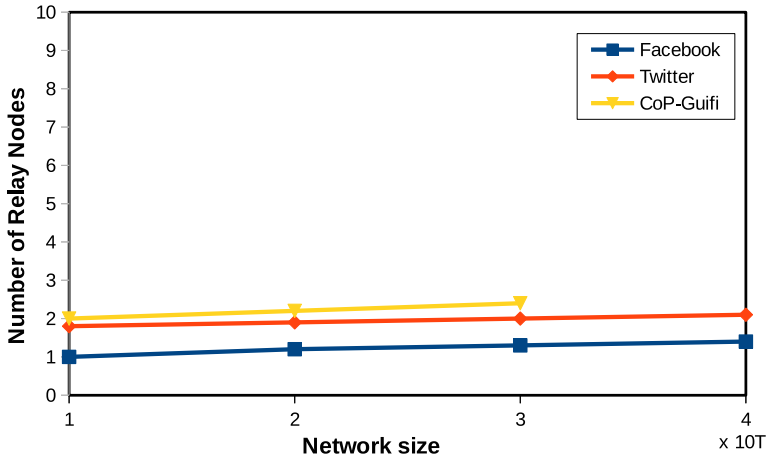


Figure 5.11: Comparison of Number of relay nodes per pub/sub routing path obtained with the use of Facebook, Twitter data sets and CoP-Guifi information.

the average number of hops increases logarithmically and we see a decrease of 76% and 83% less hops compared to the pub/sub mechanism built over the Symphony overlay network and for the Facebook and Twitter data sets, respectively. This occurs due to the fact that Symphony's construction of long range links is completely oblivious to the social graph and the publication workload.

In contrast, SELECT establishes connections between socially connected peers, and as such subscribers are 1 or 2 hops away from the publisher. Compared to the state-of-the-art pub/sub approaches, SELECT achieves more than 43% and 61% reduction for the Facebook and Twitter data sets, respectively. This happens because peer identifiers on SELECT are mutable and socially connected peers are clustered in the same region in the ID space. Hence, a

small-world network is accomplished on SELECT, in contrast to the presented approaches where an immutable identifier policy is applied.

Figure 5.10 presents the use of CoP information against using social networks, and the we show that the average number of hops is within the same values as in social networks. The use of CoP information as an alternative to social network information, does not impact the construction of the overlay network, in respect to using cooperation as the relation between users. Furthermore, although the number of users within the data set is smaller in comparison with larger social networks, the tendency of the results is very close (within a 1 hop on average) to previous results using social network data sets.

The results taken as baseline in Figure 5.4(a) and (b), explained in more detail in subsection 5.1.4 presents the impact of SELECT on the number of relay nodes that exist in the routing path between publisher and subscriber, and we see a reduction of over 98% on the number of relay nodes for all data sets, in comparison to the Symphony, Bayeux, Vitis and OMen approaches. This happens because in Symphony, Bayeux, Vitis and OMen the probability of two socially connected users to be also connected in the overlay network is extremely low. In contrast, SELECT leverages the social graph and establishes connections between socially-connected peers that reduces the number of relay nodes in the routing path between publisher and subscriber.

Figure 5.11 presents the comparison between Facebook and Twitter against using the CoP information, and in the results obtained we see that the average number of relay nodes in the overlay network maintains within the same values as our baseline using social networks. Thus, using CoP information can be used with the same results as with social networks, however, the users that are not connected through the CoP are not represented within the information which can make the number of relays and hops higher on average.

5.2.4 Discussion

Our approach to P2P overlay optimization in community networks relies on the CoP connections provided by SELECT algorithm. Although it is not possible to provide a fair cost comparison between state-of-the-art approaches and SELECT algorithm due to the fact that all of them rely on different aspects of the network, we however clearly see from our experimental results that the actual costs are the added social information necessary to create the relationship graph in order to augment the global overlay.

In respect to community of practice information, the difference of number of users is because of the network itself having less members than Online Social networks. Therefore, in our experiments the comparison is as close as possible to the maximum amount of users in CoP graph. However, we can definitely say that our approach has growth potential even when using CoP information. Furthermore, an extended study on different types of collaborative users is necessary when including all members of community networks, in such that all social peers can minimize their own routing towards other peers, without being randomized within the overlay.

The use of SELECT, with micro-cloud services in community networks is an approach that should be further studied, since our experiments were to understand the viability to apply CoP information on overlays. Also, the social peers maintain their underlying network connectivity, even though an overlay may optimize the path to which peers should data flow. Therefore, the network connectivity should be included as a metric on the creation of the overlay, in order to account for network status and social status. Although the algorithm was design with pub/sub systems in mind, it can work towards micro-clouds services, since these services use pub/sub methods for disseminating data.

Thus, we can say that applying SELECT in micro-clouds environments would guarantee optimization for dissemination of data, when coupled with network information such as latency, bandwidth capacity and resources. Moreover, by adding network and social information in an evolving environment we can enhance data dissemination for services.

We assure the correctness of our approach by grounding it in the ring topology, since it gives us the ability to continue sending messages to all peers and guarantee that all nodes are able to receive it. Other topologies, such as mesh, tend to create isolated communities of nodes. Therefore, the use of other topologies may not guarantee the same results when applying different social networks or CoP. Also, since in CoP there are users that do not communicate by mailing lists or forum as in our experiments, pockets of users without connections would be a possibility with other topologies and thus avoided when using a ring topology.

A more extensive study on CoP would be necessary to include other members, and information that is not available through the mailing lists and forums, such as meetings (where members meet face-to-face without the use of emails). Also, CoP information and relationships need to be explored in order to account the over time aspect of collaboration.

5.2.5 Summary

In this work, we approach an optimization for service overlays within CN micro-clouds by proposing the use of SELECT induced with the information from community of practices. We design a P2P overlay network that exploits the social graph of community of practice in CNs to organize the peers in an overlay network and establish connections between socially-connected peers. Using a gossip-based peer sampling service, SELECT reduces the number

of hops required to communicate two socially-connected peers. Additionally, the constructed routing trees exhibit the minimum number of relay nodes even when using different types of relationships. We evaluate SELECT in a simulated environment using three real-world data sets and highlight the performance of SELECT with state-of-the-art approaches as baseline against the use of CoP information.

In respect to our evaluation, we posit that using CoP information is a way to handle social information in community networks to reduce the number of relay nodes in the overlay network, which will be beneficial to enhance service performance in CN micro-clouds.

5.3 Conclusion

The work presented in this chapter accounts with the relationships that can be exploited from social networks, such as Facebook, Twitter. We explain how enhancing overlay networks with social information can be beneficial towards minimizing relay nodes, which in return will optimize communication by routing messages through nodes that are related to the content or the users. However, in community networks one of the main relationships is cooperation among users, therefore can social information be extracted from community networks and used to optimize overlay network as before? Does enhancing overlay networks optimize services in CN micro-clouds?

In this chapter, we present an approach to include social information in overlay networks to optimize services in CN micro-clouds. The use of social information can minimize the relay nodes in the message dissemination process. Services use communication between instances, in order to provide the content to the users, e.g. live video streaming, distributed storage or service discovery. Thus, the optimization within the middleware level, can further benefit services

communication. However a question arises, can overlay networks be further optimized by combining different properties from each perspective (resource, service and user information)? Does near-optimal routing solutions be enough in order to bring cloud services into micro-cloud environments?

Chapter 6

Discussion

In this thesis we presented the work brought forth by the required infrastructure of CN micro-clouds. The opportunity to migrate certain services from data centers closer to the users, and utilizing resources within the community networks, is very attractive. In fact, such a solution can minimize the dependency on data center cloud services and the excessive Internet connectivity in CNs. However, cloud services are not prepared for an heterogeneous environment with varying connectivity, thus, optimization on different levels, such as resources, middleware and services, should be applied.

In this work we broke down our problem in three levels, in order to improve service performance/quality in each level and gather information that would help the other levels succeed.

In the resource level, we constructed tools to augment the capacity of low-resource devices, thus utilizing virtualization in order to guarantee a fair use of resources between community and owners. Further motivating the community to share their devices, in order to enhance the computation power of the micro-clouds. The utilization of virtualization techniques does depend on the devices and resources available, i.e. shared devices in CNs may not have

the capabilities to house virtualization. Therefore, the current solution of deploying services in bare-metal, would have a better performance/quality than using virtualization technologies.

The study done with virtualization technologies also did not account with the individual use of memory, CPU and storage. In our studies, we compared our results with other works as an aggregated result instead of individually for each property, and our deployment was in an attempt to have excessive use of resources. However, CN micro-clouds are built with an heterogeneous environment, therefore individual use of particular resources may lead to a different service deployment, which is dealt by the CN micro-cloud system, when users deploy their services towards specific devices. Thus, our results were to reflect the excessive use of the resources.

In the service level, we gather information on how the services are composed, their ability to be configured for other environments and their optimization (performance, quality perceived) in CN micro-clouds. Services can be configured to withstand adverse conditions to what it was expected on data centers, e.g. in live video streaming the manipulation of data transmission rate towards different adjacent nodes, can have an immediate impact on the video quality.

The use of gossip-enabled networks, in order to understand and enhance service performance is also an option to consider. In such networks, where the environment is heterogeneous and varies its infrastructure, the gossip technique gives stability to nodes data dissemination, by relying in different dissemination paths that are according to node's neighborhood. Moreover, by applying the gossip technique, the network can adapt to node churn without losing data.

In the deployment of the monitoring tool, we do not address the fact that information may be required for long periods of time, and therefore requires to be saved, or to be re-introduced in the network when is necessary. The impact on such solution would depend on the utilization of the monitoring tool and information gathered. Solutions to this impact can range from saving information on a shared data base (potentially using the distributed storage service in CN micro-clouds), or sending older data upon request by any node. Therefore, monitoring can still be achieved with minimum interference to the services, and be available to all members of the CN micro-clouds.

The study done with actual CN micro-cloud services demonstrate how they can run in CN environments. In our results, there was no explicit methods that analyze the interference that other users or services may have had in the results obtained. However, in our service deployment we implicitly account with the network interference, meaning that the adaptation of the services to the environment accounts with what can potentially happen in the network, i.e. download/uploads by other services or users which has an impact on the bandwidth/latency of the network or the use of the devices.

In the middleware level, we look into the creation of the overlay networks that serve as communication system. Services use the overlay networks to communicate between instances and users and therefore improving this level guarantees an enhanced service communication, performance and overall quality.

The use of social information, to construct overlay networks in which we optimize paths according to the social pattern of each user, is an improved solution for publish/subscribe systems. In fact, such solution minimizes the relay nodes that are required to transfer messages. The same practice can be used on services in CN micro-clouds, in order to minimize dissemination

towards only the nodes that require such information. However, in this case several issues are to be addressed, the fact that the network infrastructure is heterogeneous and latency varies along different paths, and each service can have specific properties for data, computation or network.

In our solution we do not account with the impact in the underlay network, however such a case would only be necessary for production systems. In our experiments, the use of fewer relay nodes translates into the use of less resources in the underlay, and thus our latency results would account with the underlay system. However, the CN infrastructure is more complex, in terms of latency, bandwidth and resources, therefore an adaptation (that includes information of the underlay) is necessary for production systems.

Furthermore, the use of CoPs information as an alternative to social networks, when creating overlay networks for CN micro-clouds, is an approach that utilizes the cooperation as the social relationship within CNs. CoPs may not be applicable to all the CNs, or specific zones of the network. Other types of relations can be applied as an alternative, which will benefit the services from what happens with their usage or infrastructure. Also, the information provided by the mailing lists may not reflect a global usage of the services, rather as a local or to specific groups. Therefore, the implementation of our solution for CN micro-clouds could start at a local level, and grow towards the entire CNs when social information would become available.

The use of a global overlay network that includes each level (resources, services and users), is in respect to the quality of services, an optimal solution to minimize resource usage, optimizing data dissemination through the micro-clouds. How the combination of different factors (resources, services and social) can be applied to the overlay network and their impacts on the micro-cloud services and resources? The use of socially aware networks, coupled with

information of the infrastructure can determine the optimal paths for services to disseminate data towards nodes that require such data, while using the minimal resources of the micro-cloud.

Different properties of the network such as resources, services and social, can be combined in overlay networks separately. However, each property can be required for different aspects when using the services. Therefore, a global overlay network that includes each aspect can be beneficial, optimizing each aspect of the services. In that respect, by including information of resources, services and social properties into overlay networks it is possible to find near-optimal paths for message dissemination without loosing conditions, e.g. latency, bandwidth, trust, time consumption, high computational resources. The introduction of each property into the overlay networks can be done by differentiating each property in their own levels and giving it as input towards the best candidates for routing, as relay nodes.

Moreover, in this thesis we consider the optimization of CN micro-cloud services in each level, in order to reach for an aggregated solution that encompass each improvement, within the overlay network for data dissemination. However, does our solution improve enough the CN micro-clouds to become optimal? Our intent in this thesis is to provide a path for improvement of service performance and quality, which would provide CN members an alternative to Internet services that would be perceived with the same performance/quality as in data centers. However, with the advancements of technology it is possible that further study and improvements would be necessary to match the future of data center cloud services. Nonetheless, by breaking our problem into three levels we are able to provide improvement on service performance/quality that takes into consideration the nature (heterogeneous devices, latency, bandwidth, service deployment, service usage) of current CN micro-clouds.

Chapter 7

Conclusion

The work in this thesis demonstrates the feasibility to enhance services in micro-clouds within community networks. The use of services in micro-clouds is dependent on the users' perceived quality of service and experience according to how they perceive cloud services in the Internet. Thus, by optimizing services in the community networks environment it is possible that more users in the community choose to use the services available within the micro-clouds instead of going to the Internet.

The optimization solution provided in this thesis comprises three levels: at the resource level, we began by introducing virtualization techniques and a multi-purpose environment that empowers owners to share their devices to the community and prepare CN micro-clouds environments which potentially fosters more services in CNs. At the service level, we created a monitoring tool tailored for CN micro-clouds, that helps with understanding of how services behave within CN micro-clouds. The analysis done on services then granted us the knowledge to adapt service configuration in order to improve its performance and quality within CN micro-cloud environments. Finally, at the middleware level, the use of overlay networks that serve as the communication system, brings services a tool for message dissemination that is close to

optimal, in respect to message route/path. By adding social information in the construction of overlay networks, we contributed to reduce the number of relay nodes in the overlay network and enhanced message dissemination concerning the impact social behaviour has on the services. Since CNs do not have an explicit social network, we used community of practices as an alternative to social networks, where the predominant relation between users is cooperation within the network. This allows us to bring the concept into the CN micro-clouds, and provide an improvement to the communication system of CN micro-clouds.

7.1 Future Work

The organization of community networks allows the use of a combined factor of resources, service and user information to infer which routing paths can become more prevalent for optimization of service performance across the CN micro-clouds. Therefore, the combination of each factor within overlay networks is an option to consider when dealing with wireless and heterogeneous environments. The next steps on such a solution are the automation of the system and the ability to periodically evolve the overlay network in order to be aware of the ever changing nature of community networks.

Bibliography

[1] Minsung Jang et al. Personal clouds: Sharing and integrating networked resources to enhance end user experiences. In *INFOCOM, 2014 Proceedings IEEE*, pages 2220–2228, April 2014. 1, 30

[2] Flavio Bonomi et al. Fog computing: A platform for internet of things and analytics. In Nik Bessis and Ciprian Dobre, editors, *Big Data and Internet of Things: A Roadmap for Smart Environments*, volume 546 of *Studies in Computational Intelligence*, pages 169–186. Springer Int. Publishing, 2014. 1, 31

[3] Karthik Kumar et al. A survey of computation offloading for mobile systems. *Mob. Netw. Appl.*, 18(1):129–140, February 2013. 1, 30

[4] Mennan Selimi, Amin M. Khan, Emmanouil Dimogerontakis, Felix Freitag, and Roger Pueyo Centelles. Cloud services in the guifi.net community network. *Computer Networks*, 93, Part 2:373 – 388, 2015. Community Networks. 2, 53, 66, 77

[5] Javi Jimenez et al. Supporting Cloud Deployment in the Guifi.net Community Network. In *5th Global Information Infrastructure and Networking Symposium (GIIS 2013)*, Trento, Italy, October 2013. 2, 35

[6] Hoang T Dinh, Chonho Lee, Dusit Niyato, and Ping Wang. A survey of mobile cloud computing: architecture, applications, and approaches. *Wireless communications and mobile computing*, 13(18):1587–1611, 2013. 2

[7] R. Birke, E. Leonardi, M. Mellia, et al. Architecture of a network-aware p2p-tv application: the napa-wine approach. *Communications Magazine, IEEE*, 49(6):154–163, June 2011. 3, 95, 104

[8] Zooko Wilcox-O'Hearn and Brian Warner. Tahoe: The least-authority filesystem. In *Proceedings of the 4th ACM Int. Workshop on Storage Security and Survivability*, StorageSS '08, pages 21–26, New York, NY, USA, 2008. ACM. 3, 43

[9] Anne-Marie Kermarrec and Maarten van Steen. Gossiping in distributed systems. *SIGOPS Oper. Syst. Rev.*, 41(5):2–7, October 2007. 5

[10] Feng Xia, Li Liu, Jie Li, Jianhua Ma, and Athanasios V Vasilakos. Socially aware networking: A survey. *IEEE Systems Journal*, 9(3):904–921, 2015. 6

[11] Chen Chen, Roman Vitenberg, and Hans-Arno Jacobsen. OMen: Overlay Mending for Topic-based Publish/Subscribe Systems Under Churn. In *Proceedings of the 10th ACM International Conference on Distributed and Event-Based Systems (DEBS 2016)*, June 2016. 6, 37, 116, 117

[12] F. Rahimian et al. Vitis: A gossip-based hybrid overlay for internet-scale publish/subscribe enabling rendezvous routing in unstructured overlay networks. In *IEEE International Conference on Parallel Distributed Processing Symposium (IPDPS)*, 2011. 6, 36, 37, 116

[13] Fabrício A. Silva, Azzedine Boukerche, Thais R. M. Braga Silva, Linnyer B. Ruiz, Eduardo Cerqueira, and Antonio A. F. Loureiro. Vehicular networks: A new challenge for content-delivery-based applications. *ACM Comput. Surv.*, 49(1):11:1–11:29, June 2016. 19

[14] Davide Vega, Roger Baig, Llorenç Cerdà-Alabern, Esunly Medina, Roc Meseguer, and Leandro Navarro. A technological overview of the guifi.net community network. *Computer Networks*, 93, Part 2:260–278, December 2015. 19

[15] Axel Neumann, Ester López, and Leandro Navarro. Evaluation of mesh routing protocols for wireless community networks. *Computer Networks*, 93, Part 2(P2):308–323, December 2015. 19

[16] M. Selimi et al. Cloud-based extension for community-lab. In *22nd Int. Symposium on Modelling, Analysis Simulation of Computer and Telecommunication Systems, IEEE*, pages 502–505, Sept 2014. 21, 24

[17] Bart Braem et al. A case for research with and on community networks. *SIGCOMM Comput. Commun. Rev.*, 43(3):68–73, July 2013. 24, 100

[18] Leandro Navarro Roger Baig and Felix Freitag. On the sustainability of community clouds in guifi.net. In *Proceedings of the 13th International Conference on Economics of Grids, Clouds, Systems and Services (GECON15)*. Springer, September 2015. 27

[19] Roy Friedman, Daniela Gavidia, Luis Rodrigues, Aline Carneiro Viana, and Spyros Voulgaris. Gossiping on manets: the beauty and the beast. *ACM SIGOPS Operating Systems Review*, 41(5):67–74, 2007. 29

[20] Roger Baig, Felix Freitag, and Leandro Navarro. On the sustainability of community clouds in guifi.net. In Jörn Altmann, Gheorghe Cosmin

Silaghi, and Omer F Rana, editors, *Economics of Grids, Clouds, Systems, and Services*, Lecture Notes in Computer Science, pages 265–278. Springer International Publishing, 15 September 2015. 29

[21] Boonyarith Saovapakhiran and Michael Devetsikiotis. Enhancing Computing Power by Exploiting Underutilized Resources in the Community Cloud. In *IEEE Int. Conference on Communications (ICC 2011)*, Kyoto, Japan, June 2011. 31, 150

[22] Dale F. Willis, Arkodeb Dasgupta, and Suman Banerjee. Paradrop: A multi-tenant platform for dynamically installed third party services on home gateways. In *Proceedings of the 2014 ACM SIGCOMM Workshop on Distributed Cloud Computing*, DCC '14, pages 43–44, New York, NY, USA, 2014. ACM. 31

[23] Stephen Soltesz et al. Container-based Operating System Virtualization: A Scalable, High-performance Alternative to Hypervisors. In *EuroSys'07, the European Chapter of SIGOPS*, Portugal, March 2007. ACM SIGOPS. 31

[24] Dirk Merkel. Docker: Lightweight linux containers for consistent development and deployment. *Linux J.*, 2014(239), March 2014. 32

[25] D. Bernstein. Containers and cloud: From lxc to docker to kubernetes. *IEEE Cloud Computing*, 1(3):81–84, Sept 2014. 32

[26] Doosik Park, Sang-Moon Lee, and Changsung Lee. The cluster monitoring & controlling method with scalable communication framework. In *Proceedings of the Eighth International Conference on High-Performance Computing in Asia-Pacific Region*, HPCASIA '05, pages 387–, Washington, DC, USA, 2005. IEEE Computer Society. 32

[27] Zhengyu Liang, Yundong Sun, and Cho-Li Wang. Clusterprobe: An open, flexible and scalable cluster monitoring tool. In *Proceedings of the 1st IEEE Computer Society International Workshop on Cluster Computing*, IWCC '99, pages 261–, Washington, DC, USA, 1999. IEEE Computer Society. 33

[28] J. S. Ward and A. Barker. Monitoring Large-Scale Cloud Systems with Layered Gossip Protocols. *ArXiv e-prints*, May 2013. 33

[29] Da Cunha Rodrigues and et al. Monitoring of cloud computing environments: Concepts, solutions, trends, and future directions. In *Proceedings of the 31st Annual ACM Symposium on Applied Computing*, SAC '16, pages 378–383, New York, NY, USA, 2016. ACM. 34

[30] Rajagopal Subramaniyan, Pirabhu Raman, Alan D. George, and Matthew Radlinski. Gems: Gossip-enabled monitoring service for scalable heterogeneous distributed systems. *Cluster Computing*, 9(1):101–120, January 2006. 34

[31] L. Baldesi, L. Maccari, and R. Lo Cigno. Live p2p streaming in communitylab: Experience and insights. In *13th Annual Mediterranean Ad Hoc Networking Workshop (MED-HOC-NET)*, June 2014. 35

[32] Luca Baldesi, Leonardo Maccari, and Renato Lo Cigno. Improving P2P Streaming in Community-Lab Through Local Strategies. In *10th IEEE Int. Conference on Wireless and Mobile Computing, Networking and Communications*, pages 33–39, Larnaca, Cyprus, October 2014. 35, 43, 77

[33] Amr Alasaad. Content sharing and distribution in wireless community networks. In *Doctoral dissertation*. Univeristy of British Columbia, PhD Thesis, 2013. 35

[34] S. Traverso et al. Experimental comparison of neighborhood filtering strategies in unstructured p2p-tv systems. In *IEEE 12th International Conference on Peer-to-Peer Computing (P2P)*, pages 13–24, Sept 2012. 36

[35] A. Russo and R.L. Cigno. Pullcast: Peer-assisted video multicasting for wireless mesh networks. In *10th Annual Conference on Wireless On-demand Network Systems and Services (WONS)*, March 2013. 36

[36] Jo ao Oliveira et al. Can peer-to-peer live streaming systems coexist with free riders? In *P2P'13*, pages 1–5, 2013. 36

[37] A.P. Couto da Silva, E. Leonardi, M. Mellia, and M MEO. Exploiting Heterogeneity in P2P Video Streaming. *IEEE Transactions on Computers*, 60:667–679, May 2011. 36

[38] Shelley Q. Zhuang et al. Bayeux: An architecture for scalable and fault-tolerant wide-area data dissemination. In *Proceedings of the 11th International Workshop on Network and Operating Systems Support for Digital Audio and Video*, 2001. 36, 116

[39] Gregory Chockler, Roie Melamed, Yoav Tock, and Roman Vitenberg. Constructing scalable overlays for pub-sub with many topics. In *Proceedings of the Annual ACM Symposium on Principles of Distributed Computing*, 2007. 36, 37

[40] Chen Chen and Yoav Tock. Design of Routing Protocols and Overlay Topologies for Topic-based Publish/Subscribe on Small-World Networks. In *Proceedings of the Industry Track of the 16th ACM/IFIP/USENIX Middleware conference (Middleware Industry 2015)*, Dec 2015. 36, 37, 115, 116

[41] Fatemeh Rahimian, Thinh Le Nguyen Huu, and Sarunas Girdzijauskas. Locality-awareness in a peer-to-peer publish/subscribe network. In *IFIP International Conference on Distributed Applications and Interoperable Systems*, pages 45–58. Springer, 2012. 37

[42] Chen Chen, Hans-Arno Jacobsen, and Roman Vitenberg. Algorithms based on Divide and Conquer for Topic-based Publish/Subscribe Overlay Design. *ACM/IEEE Transactions on Networking*, 2014. 37

[43] Gregory Chockler, Roie Melamed, Yoav Tock, and Roman Vitenberg. Spidercast: A scalable interest-aware overlay for topic-based pub/sub communication. In *Proceedings of the 2007 Inaugural International Conference on Distributed Event-based Systems*, DEBS '07, pages 14–25, New York, NY, USA, 2007. ACM. 37

[44] Vinay Setty, Maarten van Steen, Roman Vitenberg, and Spyros Voulgaris. Poldercast: Fast, robust, and scalable architecture for p2p topic-based pub/sub. In *Proceedings of the 13th International Middleware Conference*, Middleware '12, pages 271–291, New York, NY, USA, 2012. Springer-Verlag New York, Inc. 37

[45] Gregory V. Chockler, Roie Melamed, Yoav Tock, and Roman Vitenberg. Spidercast: a scalable interest-aware overlay for topic-based pub/sub communication. In *DEBS*, 2007. 38

[46] Vinay Setty, Maarten van Steen, Roman Vitenberg, and Spyros Voulgaris. Poldercast: Fast, robust, and scalable architecture for p2p topic-based pub/sub. In Priya Narasimhan and Peter Triantafillou, editors, *Middleware 2012*, pages 271–291, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg. 38

[47] Maria De Marsico, Carla Limongelli, Filippo Sciarrone, Andrea Sterbini, and Marco Temperini. *Social Network Analysis and Evaluation of Communities of Practice of Teachers: A Case Study*, pages 3–12. Springer International Publishing, Cham, 2014. 38

[48] Mennan Selimi and Felix Freitag. Tahoe-LAFS Distributed Storage Service in Community Network Clouds. In *4th IEEE Int. Conference on Big Data and Cloud Computing 2014*, Sydney, Australia, December 2014. IEEE. 43

[49] Robert Birke et al. Architecture of a network-aware p2p-tv application: The napa-wine approach. *IEEE Communications Magazine*, 49:154–163, 06/2011 2011. 43, 77

[50] Mennan Selimi, Nuno Apolonia, Ferran Olid, Felix Freitag, Leandro Navarro, Agustí Moll, Roger Pueyo, and Luís Veiga. Integration of assisted p2p live streaming service in community network clouds. In *Proceedings of the IEEE 7th International Conference on Cloud Computing Technology and Science (CloudCom 2015)*. IEEE, November 2015. 53, 65, 85

[51] William D Norcott and Don Capps. Iozone filesystem benchmark. *URL: www. iozone. org*, 55, 2003. 56

[52] Bob Lantz, Brandon Heller, and Nick McKeown. A network in a laptop: rapid prototyping for software-defined networks. In *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, page 19. ACM, 2010. 84

[53] Ramon R Fontes, Samira Afzal, Samuel HB Brito, Mateus AS Santos, and Christian Esteve Rothenberg. Mininet-wifi: Emulating software-defined wireless networks. In *Network and Service Management (CNSM), 2015 11th International Conference on*, pages 384–389. IEEE, 2015. 84

[54] Robert Birke et al. A delay-based aggregate rate control for p2p streaming systems. *Computer Communications*, 35(18):2237 – 2244, 2012. 95

[55] Mennan Selimi, Felix Freitag, Roger Pueyo Centelles, Agustí Moll, and Luís Veiga. TROBADOR: Service discovery for distributed community network micro-clouds. In *29th IEEE International Conference on Advanced Information Networking and Applications (AINA 2015)*, March 2015. 98

[56] Llorenç Cerdà-Alabern, Axel Neumann, and Pau Escrich. Experimental evaluation of a wireless community mesh network. In *Proceedings of the 16th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM '13)*, pages 23–30, New York, NY, USA, 2013. ACM. 101

[57] A. Neumann, E. Lopez, and L. Navarro. An evaluation of bmx6 for community wireless networks. In *IEEE 8th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, pages 651–658, Oct 2012. 104

[58] Roy Want, Bill N Schilit, and Scott Jenson. Enabling the internet of things. *IEEE Computer*, 48(1):28–35, 2015. 115

[59] M. Nitti, R. Girau, and L. Atzori. Trustworthiness management in the social internet of things. *IEEE Transactions on Knowledge and Data Engineering*, 26(5):1253–1266, May 2014. 115

[60] A.M. Ortiz, D. Hussein, Soochang Park, S.N. Han, and N. Crespi. The cluster between internet of things and social networks: Review and research challenges. *IEEE Internet of Things Journal*, 1(3):206–215, June 2014. 115

[61] Vinay Setty et al. The hidden pub/sub of spotify: (industry article). In *Proceedings of the ACM International Conference on Distributed Event-based Systems*, 2013. 116

[62] Anwitaman Datta, Sonja Buchegger, Le-Hung Vu, Thorsten Strufe, and Krzysztof Rzadca. *Decentralized Online Social Networks*, pages 349–378. Springer, 2010. 116

[63] Márk Jelasity, Alberto Montresor, and Ozalp Babaoglu. T-man: Gossip-based fast overlay topology construction. *Computer Networks*, 53(13):2321–2339, August 2009. 116

[64] Gurmeet Singh Manku, Mayank Bawa, and Prabhakar Raghavan. Symphony: Distributed hashing in a small world. In *Proceedings of the Conference on USENIX Symposium on Internet Technologies and Systems*, 2003. 116, 124, 133

[65] Benjamin Hesmans and Olivier Bonaventure. Tracing multipath tcp connections. In *Proceedings of the 2014 ACM Conference on SIGCOMM*, 2014. 117

[66] Sarunas Girdzijauskas. Designing peer-to-peer overlays: a small-world perspective. *EPFL thesis no. 4327, advisor: Karl Aberer*, 154, 2009. 118

[67] Aristides Gionis, Piotr Indyk, Rajeev Motwani, et al. Similarity search in high dimensions via hashing. In *VLDB*, 1999. 118

[68] Bimal Viswanath et al. On the evolution of user interaction in facebook. In *Proceedings of the ACM Workshop on Online Social Networks*, 2009. 119, 136, 154

[69] Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford large network dataset collection. `http://snap.stanford.edu/data`, June 2014. 119, 128, 136, 154

[70] Krishna Dhara, Yang Guo, Mario Kolberg, and Xiaotao Wu. *Overview of Structured Peer-to-Peer Overlay Algorithms*, pages 223–256. Springer US, Boston, MA, 2010. 120

[71] Robert Ryan McCune, Tim Weninger, and Greg Madey. Thinking like a vertex: A survey of vertex-centric frameworks for large-scale distributed graph processing. *ACM Comput. Surv.*, 48(2), 2015. 134

[72] Konglin Zhu, Wenzhong Li, and Xiaoming Fu. Modeling population growth in online social networks. *Complex Adaptive Systems Modeling*, 1(1), 2013. 135, 154

[73] A. Berta, V. Bilicki, and M. Jelasity. Defining and understanding smartphone churn over the internet: A measurement study. In *IEEE International Conference on Peer-to-Peer Computing*, 2014. 135

[74] Jing Jiang et al. Understanding latent interactions in online social networks. In *Proceedings of the ACM SIGCOMM Conference on Internet Measurement*, 2010. 135

[75] Olivier Serrat. *Building Communities of Practice*, pages 581–588. Springer Singapore, Singapore, 2017. 148

[76] Mennan Selimi, Llorenç Cerdà-Alabern, Marc Sánchez-Artigas, Felix Freitag, and Luís Veiga. Practical service placement approach for microservices architecture. In *Proceedings of the 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, CCGrid '17, pages 401–410, Piscataway, NJ, USA, 2017. IEEE Press. 149

[77] Nuno Apolónia, Roshan Sedar, Felix Freitag, and Leandro Navarro. Leveraging low-power devices for cloud services in community networks. In *Future Internet of Things and Cloud (FiCloud), 2015 3rd International Conference on*, pages 363–370. IEEE, 2015. 150