UNIVERSITAT POLITÈCNICA
DE CATALUNYA
BARCELONATECH

TÉCNICO
LISBOA

On the Service Placement in Community Network Micro-Clouds

Mennan Selimi

UNIVERSITAT POLITÈCNICA DE CATALUNYA
Department of Computer Architecture
Barcelona, 2017

On the Service Placement in Community Network Micro-Clouds

Mennan Selimi

advisors

Dr. Felix Freitag

Dr. Luís Antunes Veiga

Computer Networks and Distributed Systems Group

Department of Computer Architecture

Universitat Politècnica de Catalunya

Barcelona

Spain

Thesis submitted for the degree of Doctor of Philosophy

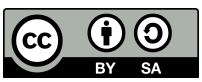
at the Universitat Politècnica de Catalunya

March, 2017

On the Service Placement in Community Network Micro-Clouds. *March 2017*.

Mennan Selimi
mselimi@ac.upc.edu

Computer Networks and Distributed Systems Group
Universitat Politècnica de Catalunya
Jordi Girona, 1-3
08014 - Barcelona, Spain



Acknowledgments

The completion of my doctoral dissertation was possible with the support of several people. I would like to express my sincere gratitude to all of them.

First of all, I am extremely grateful to my first advisor Felix Freitag (UPC), for his valuable guidance and consistent encouragement I received throughout the research work. Felix broadened my research by letting me work in various topics, at the same time continuing to contribute valuable feedback, advice, and encouragement. He made it possible for me to achieve the maximum I could, in the shortest time span. I am very grateful for his patience, motivation and valuable scholarly inputs.

I wish to express my sincerest gratitude to Leandro Navarro (UPC). Leandro was not only my program coordinator but my mentor and friend. His scientific acumen has made him as a continuous source of ideas which inspired and enriched my growth as a student and as a researcher.

I am deeply indebted to my second advisor Luís Antunes Veiga (IST) for his fundamental role in my doctoral work. He has been actively interested in my work and has always been available to advise me. He has been motivating, encouraging and enlightening.

My gratitude is also extended to Llorenç Cerdà-Alabern (UPC), for his assistance and expertise that I needed during my last year of doctoral work. His support and immense knowledge in the area of networking, taken together, make him a great mentor and a collaborator.

Some Guifi.net and Pangea.org members have been very kind to extend their help at various phases of this research, whenever I approached them, and I do hereby acknowledge all of them. I would like to give a heartfelt, special thanks to Roger Pueyo Centelles, Agustí Moll, Roger Baig Viñas, Ivan Vilata, Santiago Lamora and Jorge L. Florit.

I am also thankful to my colleagues at Distributed Systems Groups, DSG at UPC and GSD at IST/INESC-ID. Special thanks to Emmanouil Dimogerontakis (UPC), Vamis Xhagjika (UPC), Amin Khan (IST), Roshan Shedar (UPC), João Neto (UPC), Leila Sharifi (IST), Nuno Apolonia (UPC), Farnoosh Farokhmanesh (UPC), Rasha Khoury (UPC), Navaneeth Rameshan (UPC) and Khulan Batbayar (UPC). I would also like to thank my cousin Besim Bilalli (UPC) and my friend Rana Faisal Munir (UPC) for their help, support and our philosophical discussions on many topics.

During the course of my research, I got a chance to interact with many great minds, and I appreciate their feedback and insights, notably, Arjuna Sathiaseelan (University of Cambridge), Liang Wang (University of Cambridge), Fatos Xhafa (UPC), Luís Rodrigues (IST), Joan Manuel Marquès (UOC), Fernando Mira da Silva (IST), Besim Bilalli (UPC), Marc Sánchez-Artigas (Rovira i Virgili University), Rana Faisal Munir (UPC), Nesrine Khouzami (Barcelona Supercomputing Centre), Leonardo Maccari (University of Trento) and Davide Vega (Uppsala University).

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

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

Mennan Selimi
8th of March 2017
Barcelona, Spain

* * *

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

Abstract

Community networks (CNs) have gained momentum in the last few years in response to the growing demand for network connectivity in rural and urban areas. These networks, owned and managed by volunteers, offer various services to their members. While Internet access is the most popular service offered to their members, the provision of services of local interest within the network is enabled by the emerging technology of CN *micro-clouds*. By putting services closer to users, CN *micro-clouds* pursue not only an improved service performance, but also a low entry barrier for the deployment of alternatives to mainstream Internet services within the CN. Unfortunately, the provisioning of the services is not so simple. Due to the large and irregular topology, high software and hardware diversity and different service requirements in CNs, a "careful" placement of *micro-cloud* services over the network is required.

First, in order to understand the *micro-cloud* service requirements for a successful operation in CNs, we perform deployment, feasibility analysis and in-depth performance assessment of popular CN *micro-cloud* services such as distributed storage, live video-streaming and service discovery. We characterize and define workload upper bounds for successful operation of such services and perform cross-layer analysis and optimizations to improve the service performance. This deployment experience supports the feasibility of CN *micro-clouds* and our measurements contribute to understand the performance of services and applications in this challenging environment.

Then, in order to improve the performance of the services on the network level over which a service host provides a service to client nodes, it is necessary to adapt the logical network topology to both external (e.g., wireless connectivity, node availability) and internal (e.g., service copies, service demand) factors. To achieve this, we propose to leverage state information about the network to inform service placement decisions, and to do so through an i) exploratory algorithm *PASP* (Policy-aware Ser-

vice Placement) that minimizes the service overlay diameter, while fulfilling service specific criteria and ii) through a fast and low-complexity service placement heuristic *BASP* (Bandwidth and Availability-aware Service Placement), which maximizes bandwidth between nodes and improves user QoS.

Our results show that *PASP* and *BASP* consistently outperform the existing in-place strategies in the `Guifi.net` CN, with respect to bandwidth, availability and latency when used with real CN *micro-cloud* services. Since this improvement translates in the QoE (Quality of Experience) perceived by the user, our results are relevant for contributing to higher QoE, a crucial parameter for using services from volunteer-based systems.

Keywords

community networks; community network micro-clouds; service placement; cloud computing; edge computing;

Resumen

Las redes comunitarias (*Community Networks* - CNs) han cobrado impulso en los últimos años en respuesta a la creciente demanda de conectividad de red en zonas rurales y urbanas. Estas redes, desplegadas y gestionadas por voluntarios, ofrecen diversos servicios a sus miembros. Si bien el acceso a Internet es el servicio más popular ofrecido a sus participantes, la provisión de servicios de interés local dentro de la red está siendo possible gracias a la tecnología emergente de *micro-clouds* en CNs. Al acercar los servicios a los usuarios, los *micro-clouds* persiguen no sólo una mejor experiencia de uso del servicio, sino también bajar la barrera de entrada para el despliegue de alternativas a los servicios de Internet convencionales dentro de la CN. Lamentablemente, la provisión de servicios no es tan simple. Debido a una topología grande e irregular, a la alta diversidad de software y hardware y a los diferentes requisitos de servicio en las CNs, es necesaria una colocación ”cuidadosa” de los servicios de *micro-cloud* a través de la red.

Primero, para comprender los requisitos del *micro-cloud* para su operación exitosa en CNs, se realiza el despliegue, el análisis de factibilidad y la evaluación en profundidad del rendimiento de servicios populares de *micro-cloud* tales como almacenamiento distribuido, retransmisión de vídeo y descubrimiento de servicios. Se caracterizan y definen los límites superiores de la carga de trabajo para una operación exitosa de tales servicios y se realizan análisis y optimizaciones entre capas para mejorar el rendimiento del servicio. Esta experiencia de implementación apoya la factibilidad de los *micro-clouds* comunitarios, y las mediciones mediciones obtenidas contribuyen a comprender el rendimiento de los servicios y aplicaciones en este desafiante entorno.

A continuación, con el fin de mejorar el rendimiento de los servicios a nivel de red sobre la que un servidor de servicios proporciona un recurso a los nodos cliente,

es necesario adaptar la topología de red lógica tanto a factores externos (por ejemplo, conectividad inalámbrica, disponibilidad de nodo) como internos (por ejemplo, copias de servicio, demanda de servicios). Para lograr esto, se propone aprovechar la información de estado de la red para informar de las decisiones de colocación de servicios, y hacerlo a través de un i)algoritmo exploratorio *PASP* (Policy-aware Service Placement) que minimiza el diámetro del overlay de servicio, cumpliendo Y ii) a través de una heurística de colocación de servicios rápida y de baja complejidad *BASP* (Bandwidth and Availability-aware Service Placement) que maximiza el ancho de banda entre nodos y mejora la QoS del usuario.

Los resultados obtenidos muestran que *PASP* y *BASP* consistentemente superan las estrategias existentes en la CN `Guifi.net`, respecto al ancho de banda, disponibilidad y latencia cuando se usan con servicios de *micro-cloud*. Dado que esta mejora se traduce en la QoE (calidad de experiencia) percibida por el usuario, los resultados hallados son relevantes para contribuir a una mejor QoE, un parámetro crucial para el uso de servicios de sistemas basados en voluntarios.

Palabras Clave

redes comunitarias; nube comunitaria; colocación de servicios; computación en la nube; computación perimetral;

Resumo

As redes comunitárias (*Community Networks* - CNs) têm ganho popularidade nos últimos anos em resposta à crescente procura por conectividade de rede em áreas rurais e urbanas. Estas redes, propriedade de e geridas por voluntários, oferecem diversos serviços aos seus membros. Apesar do acesso à Internet ser o serviço mais popular, a provisão de serviços de interesse local dentro da própria rede é habilitada pela tecnologia emergente de *micro-clouds* em CNs. Ao colocar os serviços mais perto dos seus utilizadores, as *micro-clouds* não procuram apenas melhorar o desempenho dos serviços, mas também uma redução da barreira no que toca à implantação de outros serviços convencionais de Internet dentro da CN. Infelizmente, o provisionamento de serviços não é simples. Devido à grande e irregular topologia, à grande diversidade tanto de software como de hardware, bem como à diversidade de requisitos em CNs, é necessária uma colocação "cuidada" de serviços em *micro-clouds* na rede.

Primeiro, de forma a compreender os requisitos dos serviços em *micro-clouds* para que funcionem corretamente em CNs, analisamos a viabilidade, implementamos e avaliamos o desempenho de alguns serviços populares em *micro-clouds* tais como armazenamento distribuído, streaming de vídeo em direto, e serviços de descoberta de recursos e de outros serviços. Caracterizamos e definimos limites de carga para que as operações de tais serviços seja bem sucedida, e analisamos e optimizamos em várias camadas de forma a melhorar o desempenho do serviço. Esta experiência de implementação suporta a viabilidade das *micro-clouds* comunitárias, e as nossas experiências contribuem para o discernimento do desempenho de serviços e aplicações neste ambiente desafiante.

Depois, de forma a melhorar o desempenho dos serviços ao nível da rede sobre o qual um fornecedor disponibiliza um serviço a clientes, é necessário adaptar a topologia lógica da rede a factores tanto externos (e.g. conectividade sem fios, dispon-

ibilidade de nós) como internos (e.g. réplicas de serviço, procura). Para isso, propomos basear as decisões de colocação de serviços em informação sobre o estado da rede, e fazê-lo através de i) um algorítmo exploratório *PASP* (Policy-aware Service Placement) que minimiza o diâmetro do overlay do serviço, cumprindo critérios específicos do serviço e ii) uma heurística de colocação de serviços rápida e de reduzida complexidade *BASP* (Bandwidth and Availability-aware Service Placement), a qual maximiza a largura de banda entre os nós e melhora a qualidade de serviço (QoS) dos utilizadores.

Os nossos resultados mostram que o *PASP* e a *BASP* superam consistentemente as estratégias já em uso na rede comunitária `Guifi.net`, em relação à largura de banda, disponibilidade e latência quando usadas com serviços *micro-cloud* reais. Como esta melhoria traduz-se numa melhoria da qualidade de experiência (QoE) para o utilizador, os nossos resultados são relevantes para contribuir para uma melhoria da QoE, um parâmetro crucial para utilizar serviços baseados em recursos fornecidos por voluntários.

Palavras Chave

redes comunitárias; nuvem comunitária; colocação de serviços; computação em nuvem; edge computing;

List of Publications

The results presented in this dissertation have led to the following publications:

Journal Articles

[P1] **Mennan Selimi**, Amin M Khan, Emmanouil Dimogerontakis, Felix Freitag, and Roger Pueyo Centelles. "Cloud services in the Guifi.net community network". In: *Computer Networks* 93.P2 (Dec. 2015). (JCR IF: 1.446, Q2), pp. 373–388.

[P2] **Mennan Selimi**, Felix Freitag, Llorenç Cerdà-Alabern, and Luís Veiga. "Performance evaluation of a distributed storage service in community network clouds". In: *Concurrency and Computation: Practice and Experience* 28.11 (2015). (JCR IF: 0.942, Q3), pp. 3131–3148.

Conference Proceedings

[P3] **Mennan Selimi**, Llorenç Cerdà-Alabern, Marc Sanchez-Artigas, Felix Freitag, and Luís Veiga. "Practical Service Placement Approach for Microservices Architecture". In: *17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid 2017)*. (CORE Rank A). Madrid, Spain, May 2017.

[P4] **Mennan Selimi**, Davide Vega, Felix Freitag, and Luís Veiga. "Towards Network-Aware Service Placement in Community Network Micro-Clouds". In: *22nd International Conference on Parallel and Distributed Computing (Euro-Par 2016)*. (CORE Rank A). Grenoble, France, Aug. 2016, pp. 376–388.

[P5] **Mennan Selimi**, Llorenç Cerdà-Alabern, Liang Wang, Arjuna Sathiaseelan, Luís Veiga, and Felix Freitag. "Bandwidth-aware Service Placement in Community Network Micro-Clouds". In: *41st IEEE Conference on Local Computer Networks (LCN 2016)*. (CORE Rank A) (Short paper). Dubai, UAE, Nov. 2016, pp. 220–223.

[P6] **Mennan Selimi**, Nuno Apolónia, Ferran Olid, Felix Freitag, Leandro Navarro, Agusti Moll, Roger Pueyo Centelles, and Luís Veiga. "Integration of an Assisted P2P Live Streaming Service in Community Network Clouds". In: *7th IEEE International Conference on Cloud Computing Technology and Science (CLOUDCOM 2015)*. (CORE Rank C). Vancouver, Canada, Nov. 2015, pp. 202–209.

[P7] **Mennan Selimi**, Felix Freitag, Roger Pueyo Centelles, Agusti 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)*. (CORE Rank B). Gwangju, Korea, Mar. 2015, pp. 642–649.

[P8] **Mennan Selimi**, Felix Freitag, Roger Pueyo Centelles, and Agusti Moll. "Distributed Storage and Service Discovery for Heterogeneous Community Network Clouds". In: *7th IEEE/ACM International Conference on Utility and Cloud Computing (UCC 2014)*. London, UK, Dec. 2014, pp. 204–212.

Other Publications

The background research of this dissertation has led to the following publications:

Conference Proceedings

[P9] **Mennan Selimi** and Felix Freitag. "Tahoe-LAFS Distributed Storage Service in Community Network Clouds". In: *4th IEEE International Conference on Big Data and Cloud Computing (BDCLOUD 2014)*. Sydney, Australia, Dec. 2014, pp. 17–24.

[P10] **Mennan Selimi** and Felix Freitag. "Towards Application Deployment in Community Network Clouds". In: *14th International Conference on Computational Science and Its Applications (ICCSA 2014)*. (CORE Rank C). Guimarães, Portugal, June 2014, pp. 614–627.

[P11] Amin M Khan, **Mennan Selimi**, and Felix Freitag. "Towards Distributed Architecture for Collaborative Cloud Services in Community Networks". In: *6th IEEE International Conference on Intelligent Networking and Collaborative Systems (INCoS 2014)*. Salerno, Italy, Sept. 2014, pp. 1–9.

Abstracts, Demos & Posters

[P12] Roger Baig, Rodrigo Carbajales, Pau Escrich Garcia, Jorge L. Florit, Felix Freitag, Agusti Moll, Leandro Navarro, Ermanno Pietrosemoli, Roger Pueyo Centelles, **Mennan Selimi**, Vladimir Vlassov, and Marco Zennaro. "The Cloudy Distribution in Community Network Clouds in Guifi.net". In: *14th IFIP/IEEEInternational Symposium on Integrated Network Management, (IM 2015)*. (CORE Rank A). Ottawa, Canada, May 2015, pp. 1161–1162.

[P13] **Mennan Selimi**, Felix Freitag, Daniel Marti, Roger Pueyo Centelles, Pau Escrich Garcia, and Roger Baig. "Experiences with Distributed Heterogeneous Clouds Over Community Networks". In: *Proceedings of the ACM SIGCOMM workshop on Distributed Cloud Computing (SIGCOMM DCC 2014)*. Chicago, USA, Aug. 2014, pp. 39–40.

[P14] **Mennan Selimi**, Jorge L Florit, Davide Vega, Roc Meseguer, Ester Lopez, Amin M Khan, Axel Neumann, Felix Freitag, Leandro Navarro, Roger Baig, Pau Escrich, Agusti Moll, Roger Pueyo Centelles, Ivan Vilata, Marc Aymerich, and Santiago Lamora. "Cloud-Based Extension for Community-Lab". In: *22nd IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS 2014)*. (CORE Rank A). Paris, France, Sept. 2014, pp. 502–505.

Contents

Abstract **vii**

List of Publications **xv**

List of Figures **xxii**

List of Tables **xxiii**

List of Acronyms **xxiii**

1 Introduction **1**

1.1 Motivation 2
1.2 Problem Statement 4
1.3 Research Questions 7
1.4 Contributions 9
1.5 Scope 10
1.6 Outline of the Thesis 10

2 Background and Related Work **13**

2.1 Community Networks 13
2.1.1 Guifi.net 14
2.1.2 qMp: an Urban Community Network 16
2.1.3 Methodology and Data Collection 16
2.1.4 Topology 17
2.1.5 Micro-Clouds in Community Networks 18
2.1.6 Cloudy: Micro-Cloud-in-a-Box 19
2.2 Definitions 22

2.3 Assumptions . . . 24
2.4 Community Clouds . . . 25
2.5 Service Performance Evaluation . . . 26
2.5.1 Distributed Storage Service . . . 27
2.5.2 Live-video Streaming Service . . . 28
2.5.3 Service Discovery . . . 29
2.6 Service Placement . . . 30
2.6.1 Service Placement in Data Centre Environment . . . 31
2.6.2 Service Placement in Distributed Data Centres . . . 31
2.6.3 Service Placement in Wireless Networks . . . 32
2.7 Discussion . . . 34

3 Performance Assessment of Micro-Cloud Services **37**
3.1 Current State of Service Deployment in Guifi.net . . . 38
3.2 Experimental Environment . . . 41
3.3 Distributed Storage . . . 42
3.3.1 Tahoe-LAFS . . . 43
3.3.2 Experiment Setup . . . 44
3.3.3 Experimental Results in Community Networks . . . 46
3.3.4 Experimental Results in Microsoft Azure Cloud . . . 49
3.4 Live-video Streaming . . . 49
3.4.1 PeerStreamer . . . 50
3.4.2 PeerStreamer Assumptions and Notation . . . 51
3.4.3 Experiment Setup . . . 52
3.4.4 Scenarios . . . 53
3.4.5 Experimental Results . . . 56
3.5 Service Discovery . . . 58
3.5.1 Experiment setup . . . 59
3.5.2 The Studied Scenarios . . . 61
3.5.3 Experimental Results . . . 62
3.6 Discussion . . . 64

3.7 Summary 66

4 Topology-aware Service Placement **69**
4.1 Network Structure 70
4.2 Allocation Model and Architecture 71
4.3 Service Quality Parameters 72
4.4 PASP: Policy-Aware Service Placement 73
4.5 Experimental Results 75
4.5.1 Network Behaviour and Algorithmic Performance 76
4.5.2 Deployment in a Real Production Community Network . . . 78
4.6 Discussion 80
4.7 Summary 81

5 Service Placement Heuristic **83**
5.1 Network Characterization 84
5.1.1 Node Availability 84
5.1.2 Bandwidth Characterization 86
5.1.3 Observations 87
5.2 Context and Problem 89
5.2.1 Network Graph in qMp 89
5.2.2 Service Graph in qMp 89
5.2.3 Service Placement Problem 91
5.2.4 BASP: Bandwidth and Availability-aware Service Placement . 92
5.3 Experiment Setup 93
5.4 Comparison 95
5.5 Results 95
5.6 Evaluation in a Real Production Community Network 98
5.6.1 Live-video Streaming Service 99
5.6.2 Web 2.0 Service 100
5.7 Discussion 103
5.8 Summary 104

6 Conclusion **107**
6.1 Future Work 108
6.1.1 Composite Service Placement 108
6.1.2 Distributed Decision Making 108
6.1.3 Service Migration 109
6.1.4 Security 109

Bibliography **111**

List of Figures

1.1 Guifi.net inbound and outbound traffic from/to the Internet (Dec. 2014 - Dec. 2016) . . . 3
1.2 Guifi.net bandwidth distribution . . . 5
1.3 Rackspace bandwidth distribution . . . 5
1.4 Outline of the thesis . . . 12

2.1 Community network micro-clouds . . . 14
2.2 Micro-cloud devices . . . 14
2.3 Guifi.net topology structure . . . 18
2.4 qMp network topology . . . 18
2.5 Cloudy architecture . . . 20
2.6 Service placement problem . . . 24
2.7 Classification by applicability environment . . . 34

3.1 Tahoe-LAFS deployed in the Community-Lab testbed . . . 44
3.2 ECDF of the average throughput to the gateway/Internet. . . . 45
3.3 Write performance in CNs . . . 47
3.4 Read performance in CNs . . . 47
3.5 Summary of all storage benchmark operations in CNs . . . 48
3.6 Write performance in Azure cloud . . . 50
3.7 Read performance in Azure cloud . . . 50
3.8 Summary of all storage benchmark operations in the Azure cloud . . . 51
3.9 Average peer receive ratio . . . 56
3.10 Average chunk loss . . . 57
3.11 Average chunk playout . . . 57
3.12 Average chunk loss with different parameters . . . 58

3.13 Throughput of the nodes 60
3.14 Single service discovery time (Scenario 1) 62
3.15 Responsiveness of service discovery (Scenario 2) 63
3.16 Number of Cloudy services discovered by clients (Scenario 3) 64

4.1 Guifi.net node availability 76
4.2 Guifi.net node latency 76
4.3 PASP-Availability 77
4.4 PASP-Latency 77
4.5 PASP-Closeness 78
4.6 Average client reading times 79

5.1 qMp node availability 85
5.2 Number of nodes and links 85
5.3 Bandwidth distribution 87
5.4 Bandwidth in three busiest links 87
5.5 Bandwidth asymmetry 88
5.6 Average bandwidth to the cluster heads 96
5.7 Neighborhood connectivity graph of the qMp network 98
5.8 Average video chunk loss in qMp 100
5.9 UpdateActivity when web server placed Randomly 101
5.10 UpdateActivity when web server placed with BASP 101

List of Tables

3.1 List of network-focused Guifi.net services in Catalonia (2016) 38
3.2 List of user-focused Guifi.net services in Catalonia (2016) 39
3.3 Microsoft Azure cluster characteristics 46
3.4 Nodes in the cluster and their location 54
3.5 Summary of the scenario parameters 54
3.6 Nodes, their location and RTT from the client node 59
3.7 Cloudy services used for the experiments 61

4.1 Summary of the Guifi.net network graphs (2016) 70
4.2 Service-specific quality parameters 73

5.1 Input and decision variables 90
5.2 Centrality measures for cluster heads 97
5.3 Cloudsuite benchmark results 102

List of Acronyms

CN Community Network

CNML Community Network Markup Language

qMp Quick Mesh Project

OR Outdoor Router

PASP Policy-aware Service Placement

BASP Bandwidth and Availability-aware Service Placement

ECDF Empirical Cumulative Distribution Function

QoS Quality of Service

QoE Quality of Experience

ISP Internet Service Provider

DC Data Centre

SNMP Simple Network Management Protocol

OSPF Open Shortest Path First

BGP Border Gateway Protocol

1 Introduction

Since early 2000s, community networks (CNs) or "*Do-It-Yourself*" networks have gained momentum in response to the growing demand for network connectivity in rural and urban communities. The main singularity of CNs is that they are built "bottom-up", mixing wireless and wired links, with communities of citizens building, operating and managing the network. The result of this open, agglomerative process is a very heterogeneous network, with self-managing links and devices. For instance, devices are typically "low-tech", built entirely by off-the-shelf hardware and open source software, which communicate over wireless links. This poses several challenges, such as the lack of service guarantees, inefficient use of the available resources, and absence of security, to name a few.

These challenges have not precluded CNs from flourishing around. For instance, `Guifi.net`*, located in the Catalonia region of Spain, is a successful example of this paradigm. `Guifi.net` is defined as an open, free and neutral CN built by its members. That is, citizens and organizations pool their resources and coordinate efforts to build and operate a local network infrastructure. `Guifi.net` was born in 2004 in

*http://guifi.net/

a rural area of Catalonia, and until today, it has grown into a network of more than $32,000$ operational nodes. This makes it the largest CN worldwide [Bai+15a].

The consolidation of today's cloud technologies offer CNs the possibility to collectively build CN *micro-clouds*, building upon user-provided networks, and extending towards an ecosystem of cloud services. CN *micro-clouds* are used to deploy distributed services, such as streaming and storage services, which transfer significant amounts of data between the nodes on which they run. Unfortunately, the provisioning of these type of services is not so simple. Due to the large and irregular topology, high software and hardware diversity of CNs, a "careful" placement of *micro-cloud* services over the network is required. Obviously, without taking into account the underlying network resources, a service may suffer from poor performance, e.g., by sending large amounts of data across slow wireless links while faster and more reliable links remain underutilized.

1.1 Motivation

`Guifi.net` is a "crowdsourced network", i.e. a network infrastructure built by citizens and organisations who pool their resources and coordinate their efforts to make these networks happen. In this network, the infrastructure is established by the participants and is managed as a common resource [Bai+16]. `Guifi.net` is the largest and fast growing CN worldwide. Some measurable indicators are the number of nodes ($> 32,000$), the geographic scope ($> 50,000$ km of links), Internet traffic etc. Regarding the Internet traffic, Figure 1.1 depicts the evolution of the total inbound (i.e., pink colour) and outbound (i.e., yellow colour) traffic from and to the Internet for the last two years. A mere inspection of this figure tells us that `Guifi.net` traffic has tripled (i.e., 3 Gbps peak). Traffic peaks correspond to the arrival of new users and deployment of bandwidth-hungry services in the network.

`Guifi.net` ultimate aim is to create a full digital ecosystem that covers a highly localized area. However, this is not a trivial aim to accomplish. A quick glance at the type of services that users demand reveals that the percentage of Internet services (i.e., proxies and tunnel-based) is higher than 50% [Sel+15a]. This confirms

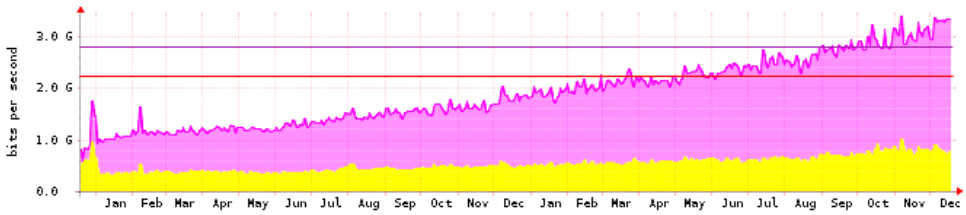


Figure 1.1: Guifi.net inbound and outbound traffic from/to the Internet (Dec. 2014 - Dec. 2016)

that `Guifi.net` users are typically interested in mainstream Internet services, which imposes a heavy burden on the "thin" backbone links, with users experiencing high service variability.

Among other issues, the above-mentioned problem spurred the invention of "alternative" service deployment models to cater for users in `Guifi.net`. One of these models was based on CN *micro-clouds*[†]. A CN *micro-cloud* is nothing but a platform to deliver services to a local community of citizens within the vast CN. Services can be of any type, ranging from personal storage to video streaming and P2P-TV [Sel+15b]. Observe that this model is different from Fog computing [Bon+12], which extends cloud computing by introducing an intermediate layer between devices and data centers. CN *micro-clouds* take the opposite track, by putting services closer to users, so that no further or minimal action takes place in the Internet. The idea is to tap into the shorter, faster connectivity between users to deliver a better service and alleviate overload in the backbone links.

CN *micro-clouds* differ in which hardware resources they are incorporated, ranging from resource-constrained devices such as home gateways, routers, embedded devices etc., to desktop-style hardware used in the local area networks.

Given the characteristics of a communication over a wireless channel, unreliable network and user devices at non-optimal locations, the physical topology of the CN where the *micro-clouds* are deployed is in a constant state of flux. In order to guarantee that the network is operational at all time, it is necessary to continuously adapt the logical configuration of the network, e.g., routing paths and neighborhood lists to

[†]http://cloudy.community/

the conditions in the physical world. This has been the research focus of the past decade i.e., optimizing the routing of packets between the nodes of the wireless mesh network, thus resulting in a great variety of reactive, proactive and hybrid routing protocols [NLN15] [03].

Taking a more service-centric view on CN *micro-clouds*, the distinction between clients and servers still exists as part of the logical network structure whenever certain nodes request services provided by other nodes. Therefore, the question arises whether the performance (i.e., service performance) of a CN such as `Guifi.net` can be improved by carefully choosing the exact nodes that are going to host a particular service. Key factors to take into account when answering this question are the connectivity between individual nodes, availability of the nodes, service demand and the suitability of services to be migrated between nodes.

The question of identifying the appropriate nodes in the CN *micro-clouds* to act as servers is referred to as the *service placement problem*, whose goal is to establish an optimal or near-optimal *service configuration*, i.e., selection of nodes to host the instances of the service which is optimal in regard to some service-specific metric [Wit10].

1.2 Problem Statement

The current network deployment model in `Guifi.net` CN is based on geographic singularities rather than on the QoS (Quality of Service). The resources in the network are not uniformly distributed [CNE13]. Furthermore, wireless links are with asymmetric quality for the services and there is a highly skewed traffic pattern and highly skewed bandwidth distribution. Figure 1.2 depicts the Empirical Cumulative Distribution Function (ECDF) of the link bandwidth in `Guifi.net` network for the first five-months of 2016. Figure reveals that 60% of the links have a bandwidth smaller than 10 Mbps and the rest 40% have a bandwidth between $10 - 100$ Mbps. The highly skewed bandwidth distribution at `Guifi.net` is not the case in the data center (DC) networks such as Rackspace or Amazon EC2 [LaC13]. Figure 1.3 reveals that in the Rackspace data center there is very little spatial variation. In fact, every path has

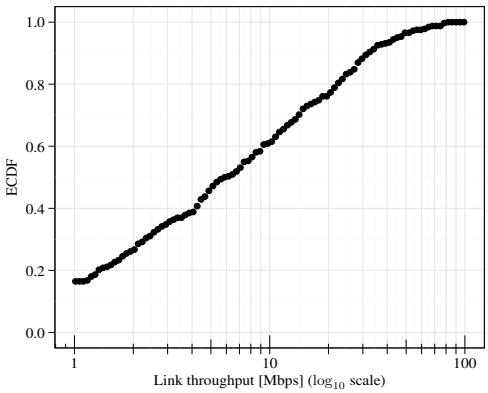


Figure 1.2: Guifi.net bandwidth distribution

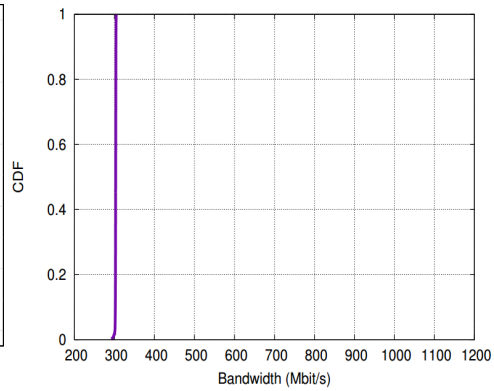


Figure 1.3: Rackspace bandwidth distribution

a bandwidth of almost exactly 300 Mbps. This implies that if a tenant were placing a single service on the Rackspace network, there would be virtually no variation for the service placement algorithms to exploit.

The network topology in a wireless CN such as `Guifi.net` is organic and different with respect to conventional ISP (Internet Service Provider) networks. `Guifi.net` is composed of numerous distributed CNs and they represent different types of network topologies. The overall topology is constantly changing and there is no fixed topology as in the DC environment. The `Guifi.net` network shows some typical patterns from the urban networks (i.e., meshed networks) combined with an unusual deployment, that do not completely fit neither with organically grown networks nor with planned networks [Veg+12]. This implies that a service placement solution (i.e., algorithm) that works in a certain topology might not work in another one.

The infrastructure in the `Guifi.net` CN is highly unreliable and heterogeneous [Veg+15]. Devices and the network are very heterogeneous compared to the DCs where they are very homogeneous. The strong heterogeneity is due to the diverse capacity of nodes and links, as well as the asymmetric quality of wireless links. Employed technologies in the CN *micro-clouds* vary significantly, ranging from very low-cost, off-the-shelf wireless (WiFi) routers, home gateways, laptops to expensive optical fibre equipment [Apo+15]. In terms of demand distribution, the demand comes directly

from the edge so there are no central load balancers as in DC environments.

Non-uniform resource distribution, *not-fixed network topology* and *heterogeneous infrastructure* makes the problem of service placement even more challenging in these environments.

The main challenge when deploying CN *micro-cloud* services is that the optimal placement of *micro-cloud* services to overcome suboptimal performance i.e., the existing in-place strategy performance. Obviously, a placement algorithm that is agnostic to the state of the underlying network may lead to important inefficiencies. Although conceptually straightforward, it is challenging to calculate an optimal decision due to the dynamic nature of CNs and usage patterns.

The problem of service placement in *micro-clouds* deployed over CNs can be stated as follows: *"Given a service and a network graph, where to place the service in the network as to maximize user's QoS (Quality of Service) and QoE (Quality of Experience), while satisfying a required level of availability for each node (N) and considering a maximum of k service copies ?*

The cost function to maximize user QoS or QoE may include metrics such as network bandwidth, latency, node availability or other service-dependent quality metrics (e.g., service overlay diameter, client response time etc.). We consider single-objective optimization to find the best solution. The choice of the cost function is mandated by the *service placement policy*. The type and the consequences of the placement policy vary depending on the service type in the CN. A very fundamental placement policy and in the fact the one we will consider mostly in this work is maximizing the bandwidth required for the service provisioning. Other goals, may be pursued in more specialized service scenarios. For example, in urban wireless CNs the placement policy may aim at a reduction of service access time (i.e., client response time). Alternatively, in a CN with nodes in non-optimal locations, a regionally diverse service placement of service instances may be preferable.

In the CN environment, it is advantageous if the services can be provided by multiple *service instances*, each of which is hosted on a different CN *micro-cloud* node. A service instance is an exact copy of the software component that provides the service,

including the executable binary and the application-level data. Each of these service instances is capable of providing the complete service on its own. These service instances do not differ between them except for which node of the network they are hosted on.

The concept of service instances gives rise to a distinction between *centralized* and *distributed* services. Distribution and decentralisation are concepts closely related to the CN philosophy; nonetheless, since centralised solutions are generally much easier to develop and deploy, in most of the cases they end up being implemented according to the classical *client-server* approach. It is technically infeasible to create multiple instances for the centralized services hence there is only a single service instance. On the other side, for the distributed services, it is possible to create multiple instances (e.g., storage nodes in the distributed storage service or source node in the video streaming service). However, they incur an additional overhead in the CN which is not present in the case of centralized services.

1.3 Research Questions

Quantifying the service performance in CN *micro-clouds* is very important in order to guarantee that the services will run successfully and not disrupt the proper function of the network. The approach for performance assessment of services in CN *micro-clouds* is to set the experimental conditions as seen from the end user: experiment in production CNs, focus on metrics that are of interest for end users and deploy services on real nodes integrated in CNs. To this end, the first question that needs to be addressed is the following one:

Q1: Is it feasible to run bandwidth-intensive and latency-sensitive services in CN micro-clouds and what are the service workload upper bounds for the successful operation of these types of services? *Which metrics should be applied for analysis?*

Not all services can be deployed in the CN *micro-clouds*. However, those services that can be deployed have different QoS requirements. For instance, bandwidth

intensive services (e.g., distributed storage) and latency sensitive services (e.g., live-video streaming) can operate successfully upon different workloads. Metrics that quantify the success of a different service type are important to be included in the service placement algorithms.

The placement of the distributed service components depends largely on the interactions and the semantics between the service components. Based on that, services that require intensive inter-component communication (e.g., streaming service), can perform better if the replicas (i.e., service components) are placed close to each other in high capacity links [Sel+15b]. On the other side, bandwidth-intensive services (e.g., distributed storage, video on-demand) can perform much better if their replicas are as close as possible to their final users (i.e., overall reduction of bandwidth for service provisioning) [Sel+16a]. The service components of bandwidth-intensive and latency-sensitive services deployed, create a service overlay graph. Therefore, we should tackle the following question:

Q2: What is the impact of the service overlay diameter on the CN micro-cloud service performance? *What is the optimal diameter (i.e., radius) for service allocation in Guifi.net?*

As services become more network-intensive, they can become bottle-necked by the network, even in well-provisioned clouds. In the case of CN *micro-clouds*, network awareness is even more critical due to the limited capacity of nodes and links, and an unpredictable network performance. Without a network-aware system for placing services, locations with poor network paths may be chosen while locations with faster, more reliable paths may remain unused, resulting ultimately in a poor user experience. To fulfil this need, we should tackle the following question:

Q3: Given a CN micro-cloud infrastructure, what is an effective and low-complexity service placement solution that maximises the end-to-end performance (e.g., bandwidth)? *Can the redundant placement of services further improve performance?*

1.4 Contributions

In this section, we outline the major contributions of the thesis by mapping each contribution to the associated research question. The major contributions of the thesis are listed as follows:

C1: *A performance assessment of a distributed storage service, live-video streaming service and service discovery in a CN micro-cloud platform*. First, this contribution identifies the requirements of deploying these type of services in a CN *micro-cloud* environment. Second, we characterize and define workload upper bounds for the successful operation of such services. Third, we conduct cross-layer analysis and optimizations on the service level to improve the service performance in a CN *micro-cloud* environment. This contribution specifically addresses the research question **Q1** and we discuss it in Chapter 3. The main results related to this contribution were originally reported in the publications [**P1**], [**P2**], [**P6**] and [**P7**].

C2: *A service placement algorithm* PASP *(Policy-aware Service Placement) that explores different placements, searching for the local minimal service overlay diameter, while at the same time fulfilling different service type quality parameters*. The algorithm finds the minimum possible distance in terms of number of hops between two furthest selected resources (i.e., service components), without the need to verify the whole solution space. In addition to minimizing the service overlay diameter, the *PASP* exploratory algorithm considers the latency and availability metric for the latency-sensitive services and closeness metric for bandwidth-intensive services. This contribution specifically addresses the research question **Q2**, and we discuss it in Chapter 4. The main results related to this contribution were originally reported in the publication [**P4**].

C3: *A placement heuristic called* BASP *(Bandwidth and Availability-aware Service Placement), which uses the state of the underlying CN to improve the service deployment*. In particular, it considers two sources of information: i) network bandwidth and ii) node availability to make optimized decisions. Compared to brute-force search,

which runs on the order of hours to complete, *BASP* runs much faster; it just takes a few seconds, while achieving reasonably good results. This contribution addresses the research question **Q3** and we discuss it in Chapter 5. The main results related to this contribution were originally reported in the publications **[P3]** and **[P5]**.

1.5 Scope

The goal of this research work is to assess the current service placement in effect in a representative CN, analyse its inefficiencies, and propose and evaluate a feasible and an effective solution to improve it. The algorithms are designed to interact closely with the domains of routing and service discovery. There are several other, closely related areas of research, which we will not consider in depth in this work:

- **Incentives for cooperation:** it is out of scope to establish under which motivation nodes of the CN *micro-clouds* should decide to host service or forward packets for other nodes. The assumption that nodes are willing to cooperate to achieve a common goal is widely used in the wireless network research. However, we we can reference the work of Khan [Kha16] that was particularly done for CN scenario.

- **Security:** It is beyond the scope of our current work to make the system or algorithms robust against attacks from malicious nodes.

1.6 Outline of the Thesis

Figure 1.4 depicts the chapters where we discuss the contributions. Furthermore, the figure shows the publications accepted and how they match with the thesis chapters. Based on that, the work in thesis is structured as follows:

We begin with a review of the fundamentals of wireless CNs and service placement in Chapter 2. In this chapter, we give brief definitions for the terms and concepts that we have introduced informally in the whole thesis. We also present an in-depth review

of the state of the art of service placement problem and classify current proposals by their applicability environment.

Chapter 3 presents the current state of service deployment in the `Guifi.net` CN and the performance assessment of three type of popular services in these environments. This chapter presents the first part of our contribution, which is the feasibility and in-depth performance assessment of CN *micro-cloud* services such as distributed storage, video streaming and service discovery.

In Chapter 4 we present our *PASP* algorithm, i.e., exploratory algorithm that finds all the optimal and sub-optimal service overlay placements. First we study the effectiveness of our approach in simulations using real-world node and usage traces from `Guifi.net` nodes. Subsequently, we deploy our algorithm, driven by these findings, in a real production CN and quantify the performance and effects of our algorithm with a distributed storage service.

In Chapter 5 we present our low-complexity service placement heuristic called *BASP* that maximises the network bandwidth and node availability when deploying CN *micro-cloud* services. We present algorithmic details, analyse its complexity, and carefully evaluate its performance with realistic settings.

Chapter 6 concludes the thesis and indicates future directions.

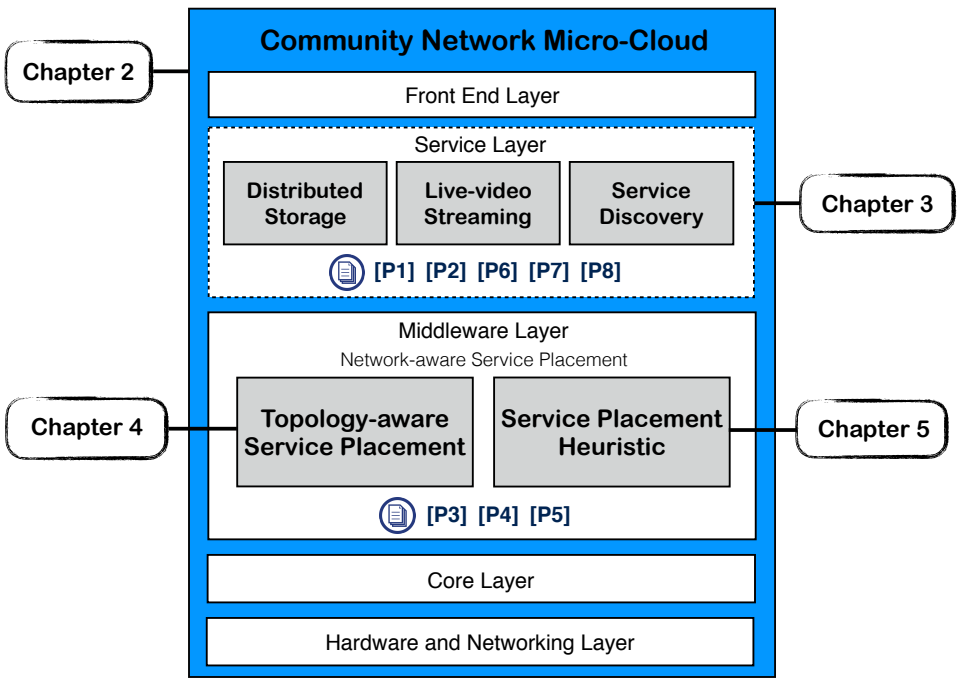


Figure 1.4: Outline of the thesis

2

Background and Related Work

Community networks (CNs) would greatly benefit from the additional value of applications and services deployed inside the network through CN *micro-clouds*. CN *micro-clouds* are a social collective model, and need contribution from its participants for its sustainability and growth.

In this chapter, first we characterize the CNs, their topological structure, tools used in these networks and then we give brief definitions for the terms and concepts introduced in the thesis. Further, we present an in-depth review of the state of the art of service placement problem and classify current proposals by their applicability environment.

2.1 Community Networks

CNs are decentralized and self-organized communication networks built and operated by citizens for citizens. In these networks, the infrastructure is established by the participants and is managed as a common resource. The infrastructure consists of mesh routers, mesh clients and optionally gateways (i.e., proxies) as shown in

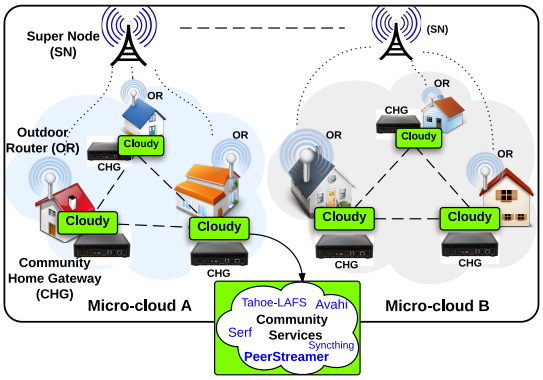


Figure 2.1: Community network micro-clouds

Figure 2.2: Micro-cloud devices

Figure 2.1. The routers (i.e., outdoor routers) communicate with each other via radio transmissions and employ special-purpose routing protocols as BMX6 or OLSR [NLN15]. Mesh clients access the network via one of the routers, while gateways provide connectivity to the Internet. Client nodes consists of home gateways, laptops or desktop PCs. The main goal of the CNs is to satisfy community's demand for Internet access and information and technology services. There are several large CNs in Europe having from 500 to 32000 nodes, such as `Guifi.net` in Spain, FunkFeuer[*] in Austria, AWMN (Athens Wireless Metropolitan Network)[†] in Greece, Freifunk[‡] in Germany and many more worldwide. Most of them are based on Wi-Fi technology (i.e., ad-hoc networks, IEEE 802.11a/b/g/n/ac access points in the first hop, long-distance point-to-point WiFi links for the trunk network), but also optical fiber links are used in some areas.

2.1.1 Guifi.net

`Guifi.net` is defined as an open, free and neutral CN. It started in the Catalonia region of Spain and its built by CN members: citizens and organizations pooling their

[*] http://www.funkfeuer.at
[†] http://www.awmn.gr
[‡] http://freifunk.net/

resources and coordinating efforts to build and operate a local network infrastructure. The underlying principle behind `Guifi.net` is the common pool resource (CPR) model as the optimal method to manage a network [Bai+15a].

`Guifi.net` network started in 2004 and today it has more than 32.000 operational nodes, which makes it one of the largest CNs worldwide. The network grows very quickly; in a week about 30 nodes can become operational and a hundred new nodes can be planned [Veg+15]. The `Guifi.net` CN consists of a set of *nodes* interconnected through mostly wireless equipment that users, companies, administrations install and maintain in addition to its links, typically on building rooftops. The set of nodes and links are organized under a set of mutually exclusive and abstract structures called administrative *zones*, which represent the geographic areas where nodes are deployed. A *zone* can represent nodes from a neighborhood or a city. Each *zone* can be further divided in child zones that cover smaller geographical areas where nodes are close to each other.

The `Guifi.net` network offers a wide range of services, provided by individuals, social groups, small nonprofit or commercial service providers. The predominant trend in `Guifi.net` is to use the available resources mostly as a means to access external services provided elsewhere in the Internet. The most common way for users to use the Internet services is through proxies [DMN17] (i.e., web proxies). Using proxies, `Guifi.net` users can access their favourite Internet-based cloud services.

At the routing level, `Guifi.net` network is split into Autonomous Systems; most of them are internally running Open Shortest Path First (OSPF) and interconnected via Border Gateway Protocol (BGP). With respect to the network interconnection, `Guifi.net` is connected to many networks, including the Internet, in different ways. In fact, `Guifi.net` serves as an "umbrella" for the other small urban and rural CNs. Peering with other networks is the preferred method because this fits better with the principles of the project, carriers, and domestic Internet connections.

2.1.2 qMp: an Urban Community Network

qMp (Quick Mesh Project) began as an effort to bring quick and easy Wifi networks into large, crowded events such as concerts, demonstrations, public events etc., by leveraging ad-hoc and dynamic routing technologies. *qMp* network, began deployment in 2009 in a quarter of the city of Barcelona, Spain, called Sants, as part of the Quick Mesh Project [16a]. *qMp* is an urban mesh network and it is a subset of the `Guifi.net`, sometimes called qMpSU. In the rest of the thesis we will use the name *qMp* to refer to the `Guifi.net` CN in an urban area.

At the time of writing the thesis, *qMp* has around 71 nodes. There are two gateways (i.e., proxies) distributed in the network that connect *qMp* to the rest of `Guifi.net` and Internet as shown in the Figure 2.4. A detailed description of the *qMp* network can be found in [CNE13].

Typically, *qMp* users have an outdoor router (OR) with a WiFi interface on the roof, connected through Ethernet to an indoor AP (access point) as a premises network. The most common OR in *qMp* is the NanoStation M5, which is used to build links on the network and integrates a sectorial antenna with a router furnished with a wireless 802.11an interface. Some strategic locations have several NanoStations, that provide larger coverage. In addition, some links of several kilometers are set up with parabolic antennas (i.e., NanoBridges). ORs in *qMp* are flashed with the Linux distribution which was developed inside the *qMp* project which is a branch of OpenWRT [17a] and uses BMX6 as the mesh routing protocol [NLN12].

The user devices connected to the ORs consists of Minix Neo Z64 and Jetway mini PCs, which are equipped with an Intel Atom CPU. They run the *Cloudy* [Bai+15b] operating system, which allows running services in LXC [17b] and Docker [17c] containers.

2.1.3 Methodology and Data Collection

Network measurements in CNs have been obtained by connecting via SSH (Secure Shell) to each *qMp* OR or `Guifi.net` node and running basic system commands available in the *qMp* distribution. This method has the advantage that no changes

or additional software need to be installed in the nodes. Live measurements used in Chapter 4 and Chapter 5 have been taken hourly over a five-month period, starting from July 2016 to November 2016, and live monitoring page and data is publicly available in the Internet[§]. We use this data to analyse main aspects of *qMp* network.

Regarding the network topology, we have collected network description data through CNML files [16b]. CNML (Community Networks Markup Language) is an XML-based language used to describe CNs. `Guifi.net` publishes a snapshot of its network structure every 30 min with a description of registered nodes, links and their configurations. In the CNML description, the information is arranged according to different geographical zones in which the network is organised. Furthermore, we used a *Node database*: a dump of the CN database that, in addition to the data described in CNML, includes other details about dates and people involved in the creation and update of the configuration of nodes and links.

2.1.4 Topology

`Guifi.net` is composed of numerous distributed CNs (e.g., *qMp* network) and they represent different type of network topologies. The overall topology is constantly changing and there is no fixed topology as in the data center (DC) environment. The network has a mesh topology in the *backbone*, and each node of the backbone (i.e., super-node) provides access to the client nodes [Veg+12]. The backbone interconnects different super-nodes by point to point links and by local networks provides access to the client nodes. The role of the *super-nodes* is to distribute (i.e., route) the traffic to the other super-nodes. In conventional star topology networks (i.e., rural CNs), the super-nodes have much more capacity than simple nodes. Figure 2.3 shows the topology structure followed in `Guifi.net`. Client nodes are connected to the super-nodes. These super-nodes interconnect through wireless links different administrative zones. Figure 2.4 depicts the topology used in the *qMp* network. As it can be seen, in urban CN, the used topology is a mesh-based.

The CNML information obtained from the `Guifi.net` CN has been used to build

[§]http://dsg.ac.upc.edu/qMpsu/index.php

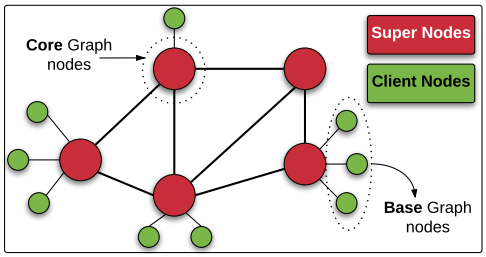


Figure 2.3: Guifi.net topology structure

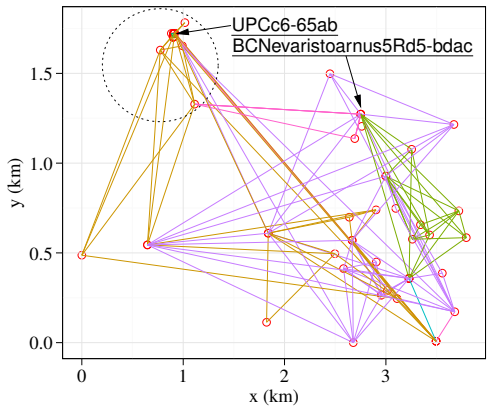


Figure 2.4: qMp network topology

two topology graphs: *base-graph* and *core-graph*. The *base-graph* of `Guifi.net` is constructed by considering only operational nodes, marked in *Working* status in the CNML file, and having one or more links pointing to another node in the zone. Additionally, we have discarded some disconnected clusters. All links are bidirectional, thus, we use an undirected graph. We have formed what we call the *core-graph* by removing the terminal nodes of the *base-graph* (i.e., client nodes).

2.1.5 Micro-Clouds in Community Networks

CN *micro-clouds* are built on top of the CNs. In this model, a cloud is deployed closer to CN users and other existing infrastructure. The CN *micro-cloud* model is different from Fog computing, which extends cloud computing by introducing an intermediate layer between devices and data centers. CN *micro-clouds* take the opposite track, by putting services closer to consumers, so that no further or minimal action takes place in Internet. They are deployed over a single or set of user nodes, and comparing to the public clouds they have a smaller scale, so one still gets high performance due to locality and control over service placement.

The devices forming the CN *micro-clouds* are co-located in either users homes (e.g., as home gateways, routers, laptops etc., as shown in Figure 2.2) or within other infra-

structures distributed in the CNs. The concept of *micro-clouds* can also be introduced in order to split deployed CN nodes into different groups. For instance, a *micro-cloud* can refer to these nodes which are within the same service announcement and discovery domain. Different criteria can be applied to determine to which *micro-cloud* a node belongs to. Applying technical criteria (e.g., Round-trip time (RTT), bandwidth, number of hops, resource characteristics) for *micro-cloud* assignment is a possibility to optimize the performance of several services. But also social criteria may be used, e.g., bringing in a *micro-cloud* cloud resources together from users which are socially close may improve acceptance, the willingness to share resources and to maintain the infrastructure.

2.1.6 Cloudy: Micro-Cloud-in-a-Box

The failure of services gaining traction in CNs was largely due to the difficulty of implementing the services and for the end-users to consume these services. To overcome these issues, CN enthusiasts designed a CN *micro-cloud* distribution, codenamed *Cloudy* [Bai+15b] [17d]. *Cloudy* is a tool that fosters the adoption and uptake of CN *micro-cloud* services among the users. It is a volunteer-based approach where users can deploy their preferred services and share with the others in CNs. *Cloudy* is the core software of our *micro-clouds*, because it unifies the different tools and services of the cloud system in a Debian-based Linux distribution. It can run directly on a bare metal machine or on a virtual machine and it is open-source and can be downloaded from public repositories[¶].

The current prototype of *Cloudy* implements the modules/layers shown in the Figure 2.5. *Cloudy's* main components can be considered a layered stack with services residing both inside the kernel and higher up at the user-level. The following three groups classify *Cloudy* services:

[¶]http://repo.clommunity-project.eu/images/

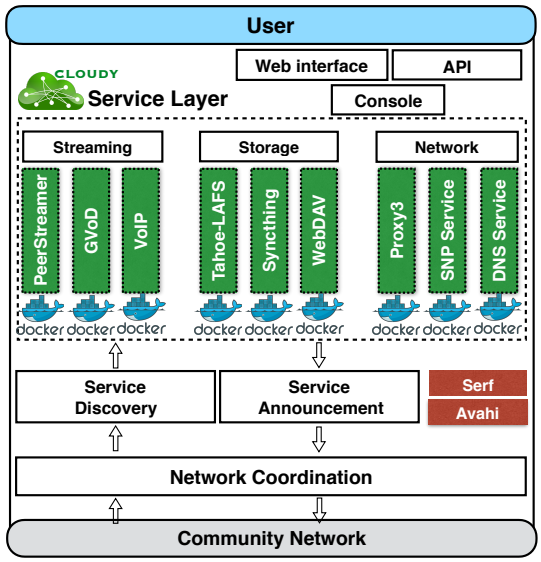


Figure 2.5: Cloudy architecture

Infrastructure Services

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

OpenVZ [17e] is an operating system-level virtualisation technology for Linux based on containers. OpenVZ allows creating multiple secure, isolated operating system instances called containers (i.e., commonly known as VPSs) on a single physical machine enabling better server utilisation and ensuring that applications do not conflict with each other. OpenVZ is the preferred solution for providing virtual machines in *Cloudy* with low to mid-end hardware as only a negligible portion (i.e., 1-2%) of the CPU resources is spent on virtualisation. The *Cloudy* distribution includes a script that downloads and installs all the required OpenVZ packages in one click and *Cloudy* instances can be run on the virtual machines created using the OpenVZ Web Panel.

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

Service Discovery and Network Coordination Services

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

For service discovery, *Cloudy* includes a customised version of Avahi [17f] to provide decentralised service discovery at Layer 2, which is needed to discover other services that will be used to provide higher-level services. For the network coordination component, *Cloudy* adopts TincVPN [17g], a virtual private network (VPN) daemon that uses tunnelling and encryption to create a secure private Layer 2 network between hosts of different domains. This Layer 2 connectivity is needed between nodes, since they may reside on different administrative domains and even be located behind firewalls. The TincVPN is automatically installed and configured on every *Cloudy* node, ready to be activated.

Cloudy also includes Serf [17h], a lightweight tool to announce and discover services in CNs. Serf is a decentralised solution for cluster membership, failure detection, and orchestration. It relies on an efficient and lightweight gossip protocol to communicate with other nodes that periodically exchange messages between each other. This protocol is, in practice, a fast and efficient method to share small pieces of information. An additional by-product of having this service distributed all over the community *micro-clouds* is that it allows the evaluation of the quality of the point-to-point connections between different *Cloudy* instances. This way, *Cloudy* users can decide which service provider to choose based on network metrics, such as RTT, number of hops,

or packet loss. The combination of Avahi, TincVPN, and Serf in *Cloudy* facilitates the coordination of the resources and the services in the CN *micro-cloud*.

User Services

Platform as a Service (PaaS). Providing attractive platform services to community members, such as a distributed file system, highly available key-value store, file synchronisation, video streaming, video-on-demand, VoIP, network address translation (NAT) traversal support, and many more, is of high importance. One of the promising services for storage is Tahoe-LAFS [WW08]. Tahoe-LAFS is a free, open, and secure cloud storage system. The configuration of Tahoe-LAFS and the process of deploying a whole storage grid are assisted by the Avahi and Serf service discovery tools using the web interface of *Cloudy*. A detailed description of Tahoe-LAFS service is given in Section 3.3.1. Etcd [17i], a highly available key value store for shared configuration and service discovery, and Syncthing [17j], an open-source file synchronisation client/server application, are already included in the Cloudy distribution.

Software as a Service (SaaS). *Cloudy* allows the user services to be present inside the CN and to be easily deployed and managed via the *Cloudy* interface. One of these multimedia services included in *Cloudy* is PeerStreamer [16c], an open source live streaming platform. Streaming is assisted by *Cloudy* by supporting the user in publishing a video stream or connecting to a peer (i.e., assisted by Serf or Avahi). A detailed description of PeerStreamer service is given in Section 3.4.1.

2.2 Definitions

Service A service is a software component executed on one or several nodes of the CN *micro-clouds*. It consists of both service-specific logic and state. A service is accessed by local or remote clients by the means of issuing service request that, in case of remote clients are transmitted across the network. In our case the services are running in a Docker and LXC containers. Several attributes of a service are of special interest in the context of service placement:

- **Centralized vs. distributed services** The service placement is applicable to both centralized and distributed services. For centralized services i.e., running on one node, service placement it controls which node should host the service. For distributed services, the service placement algorithm it also manages the granularity with which the service is to be split up, i.e., the number of service components that are to be distributed in the network.

- **Monolithic vs. composite services** If a service can be decomposed into multiple independently deployable sub-services, each of which contributes a different aspect of the overall service, then it is a *composite service*. In contrast, a *monolithic* service cannot be split into sub-services either due to semantics or implementation concerns. Mainly, we focus on monolithic services in this thesis work.

Service overlay When service components are deployed on the CN *micro-clouds* they create a service overlay graph. Our service placement algorithm searches in a large space of solutions, looking for those that minimize the overlay diameter i.e., number of maximum hops between two selected resources in the sub-graph.

Network and Service Graph The physical network topology (i.e., *network graph*) refers to the connectivity between nodes that form the wireless CN. Each node in the network graph is denoting network elements such as servers, routers, home-gateways etc., and edges denoting communication links between the nodes. The network graph is subject to continues change due to node churn, mobility and changing properties of wireless links. The logical network topology (i.e., *service graph*) consists of links created from the service component interactions. Each node of the service graph represent processing/computation modules in the service and edges represent communication demand between the nodes [Wan15]. Figure 2.6 illustrates the network and service graph.

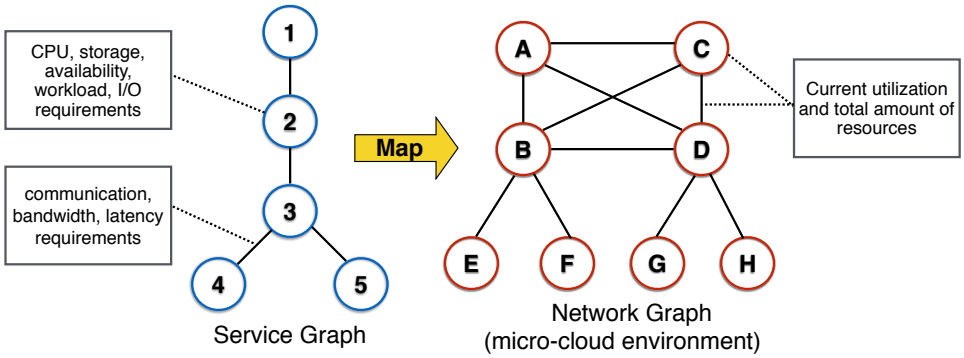


Figure 2.6: Service placement problem

Offline and Online Service Placement We can abstract the service placement problem as a graph (i.e., node) placement problem as illustrated in the Figure 2.6. Service placement can be seen as an application of facility location problem (FLP) [Ver11] to ad hoc networking (i.e., wireless CNs). FLPs are NP-Hard in their general formulation, but approximations exist. Service placement problem is the problem of mapping the service graph into an actual network graph, subject to some network or node constraints (e.g., bandwidth, latency, availability etc.). Throughout this thesis, we say that a service placement is *offline* when our goal is to place a single or a set of service graphs "in one shot". In contrast, an *online* service placement is the case where we have an incoming stream of service graphs, which have to be sequentially placed onto the network graph as each service graph arrives.

2.3 Assumptions

Deploying service placement algorithms in CN *micro-clouds* relies upon several assumptions about the context in which the wireless CNs are deployed and about the capabilities of the nodes. The assumptions are as follows:

- **Bounded heterogeneity of devices:** This work takes the network characteristics into account, while most of the service placement approaches generally consider only device characteristics (e.g., CPU, memory etc). However, the het-

erogeneity of the devices with regard to their capabilities needs to be bounded. We assume, most, if not all, nodes in the network possess sufficient resources (i.e., CPU and memory) to host a service instance.

- **Node cooperation**: Since we are dealing with a contributory computing environment like wireless CNs, service placement relies upon the assumption that the nodes are willing to cooperate with each other in order to achieve a common goal. This assumption is used also in the core of routing protocols that are used and service placement applies to the area of service provisioning.

2.4 Community Clouds

Carving our path towards community clouds in CNs, we must consider the cloud essential characteristics, as described in [MG11]. *Broad network access* is already offered by the CNs and *resource pooling* should be an outcome of the resource sharing described above. *Measured services* are very important in order to guarantee that the services will not disrupt the proper function of the network. *On-demand self-service* is a higher-level concept concerning the responsiveness and the transparency of the system; thus, this is an important feature but of secondary priority. Similarly, *rapid elasticity* of the resources offered is a welcome property; yet, it should not be considered an essential one due to its complexity because of the highly distributed environment.

The idea of collaboratively built community clouds follows earlier distributed voluntary computing platforms, such as BOINC [And04], Folding@home [Beb+09], PlanetLab [Chu+03], and Seattle [Cap+09], which largely rely on altruistic contributions of resources from users, functioning as research platforms. There are only a few research proposals for community cloud computing [MB09], and most of them do not go beyond the architecture level, whereas very few present a practical implementation.

The Cloud@Home [DP12] project aims to harvest resources from the community for meeting the peaks in demand, working with public, private, and hybrid clouds

to form cloud federations. The Clouds@Home [Yi+11] project focuses on providing guaranteed performance and ensuring quality of service, even when using volatile Internet volunteered resources. The P2PCS [BMT12] project has built a prototype implementation of a decentralised peer-to-peer cloud system. It uses Java JRMI technology and builds an IaaS system that provides very basic support for creating and managing virtual machines. These implementations, to our knowledge, are not actually deployed inside real CNs, considering the infrastructure diversity, and are not aiming to satisfy end-user needs.

Social cloud computing [Cha+12] is a relevant research field that takes advantage of the trust relationships between members of social networks to motivate contribution towards a cloud storage service. Users trade their excess capacity to earn virtual currency and credits that they can utilise later, and consumers submit feedback about the providers after each transaction, which is used to maintain the reputation of each user. Social clouds have been deployed in the CometCloud framework by federating resources from multiple cloud providers [Pun+13]. The social compute cloud [Cat+14], implemented as an extension of the Seattle platform [Cap+09], enables the sharing of infrastructure resources between friends connected through social networks and explores bidirectional preference-based resource allocation.

Among federated cloud infrastructures, Gall et al. [GSF13] have explored how an InterCloud architecture [BRC10] can be adapted to community clouds. Further, Esposito et al. [Esp+13] presented a flexible federated cloud architecture based on a scalable 'publish and subscribe' middleware for dynamic and transparent interconnection between different providers. Moreover, Zhao et al. [ZLL14] explored efficient and fair resource sharing among the participants in community-based cloud systems. In addition, Jang et al. [Jan+14] implemented personal clouds that federate local, nearby, and remote cloud resources to enhance the services available on mobile devices.

2.5 Service Performance Evaluation

Development and deployment of services in the CN *micro-clouds* it can be very challenging. The feasibility of running services in CN *micro-clouds* can be demonstrated

by carefully performing measurements taking into account different data workloads on these cloud infrastructures. The services used in the thesis are selected according to their potential relevant for CN users. Some of the popular open source services we consider are the distributed storage service, live-video streaming service and service discovery [Sel+15a].

2.5.1 Distributed Storage Service

After basic connectivity, storage is the most general service being fundamental for cloud take-up in the CN scenarios. In terms of providing cloud storage services in WAN settings, Chen's paper [Che+13] is the most relevant to our work. The authors deployed open source distributed storage services such as Tahoe-LAFS [WW08], QFS [16d], and Swift [16e] in a multi-site environment and measured the impact of WAN characteristics on these storage systems. The authors deployed their experiments on a multi-site data center with very different characteristics to our scenario.

The authors in [Gra+13] present a measurement study of a few Personal Cloud solutions such as DropBox, Box, and SugarSync. The authors examine central aspects of these Personal Cloud storage services to characterize their performance, with emphasis on the data transfers. They report that they found interesting insights such as the high variability in transfer performance depending on the geographic location; the type of traffic, namely inbound or outbound; the file size; and the hour of the day. Their findings regarding the impact of location on the performance is relevant for our work to better understand network dependence of distributed storage services.

Another work [Tse+12] implements a distributed file system for Apache Hadoop. The original Hadoop distributed file system is replaced with the Tahoe-LAFS cloud storage. The authors investigated the total transmission rate and download time with two different file sizes. Their experiment showed that the file system accomplishes a fault-tolerant cloud storage system even when parts of storage nodes had failed. However in the experiments only three storage nodes and one introducer node of Tahoe-LAFS were used, and their experiments were run in a local context, which is an unrealistic setting for our scenario. Another paper [SD12] evaluates XtreemFS

[16f], Ceph [16g], GlusterFS [16h] and SheepDog [16i], using them as virtual disk image stores in a large-scale virtual machine hosting environment. The StarBED testbed with powerful machines is used for their experiments. Differently, we target a distributed and heterogeneous set of storage nodes.

The paper of Roman [10] evaluates the performance of XtreemFS under the IO load produced by enterprise applications. They suggest that XtreemFS has a good potential to support transactional IO load in distributed environments, demonstrating good performance of read operations and scalability in general. XtreemFS is an alternative candidate for implementing a storage service upon. Tahoe-LAFS, however, more strongly addresses fault-tolerance, privacy and security requirements.

From the review of the related work it can be seen that not all of the experimental studies regarding the distributed storage, were conducted in the context of CNs. In our work, we emphasized the usage of distributed storage services, such as Tahoe-LAFS, in a real deployment within CN *micro-clouds*, to understand its performance and operational feasibility under real network conditions. Furthermore, we use heterogeneous and less powerful machines as storage nodes.

2.5.2 Live-video Streaming Service

The work of Baldesi et al. [BML14] [BMC15], evaluates PeerStreamer [16c], a P2P video streaming platform, on the Community-Lab, the wireless CN testbed of the EU FIRE project CONFINE [16j]. Their experiments highlight the feasibility of P2P video streaming, but they also show that the streaming platform must be tailored ad-hoc for the wireless CNs itself to be able to fully adapt and exploit its features and overcome its limitations. However they evaluated with a limited number of nodes (i.e., 16 `Guifi.net` nodes), which were located in the city of Barcelona and they do not use live-video streaming. A recent PhD dissertation [Ala13] includes some discussion on P2P streaming on wireless CNs, but does not elaborate on live streaming, but consider streaming of Video on Demand (VoD) retrieval.

Another work [Tra+12] studies different strategies to choose neighbours in a P2P-TV system (i.e., PeerStreamer). The authors evaluate PeerStreamer on a cluster and

on Planetlab. In wireless networks PULLCAST [RC13], 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 [Oli+13]. The paper in [Cou+11] investigates the impact of peer bandwidth heterogeneity on the performance of a mesh based P2P system for live streaming.

From the review of the related work, it can be seen that most of the experimental studies related to video streaming service were not conducted in the context of CNs. In our work we emphasis on studying the video streaming service in a real deployment scenario within CNs, in order to understand their performance and operation feasibility under real network conditions.

2.5.3 Service Discovery

In terms of examining the dependability aspects of decentralized service discovery concepts in unreliable networks, Dittrich's and Salfner's paper [DS10] is the most relevant to our work. The authors evaluate the responsiveness of domain name system (DNS) based service discovery under influence of packet loss and with up to 50 service instances. Their empirical results show that the responsiveness of the used service discovery mechanisms decreases dramatically with moderate packet loss of around 20 percent. However their experimental evaluation is based on simulations and has not been applied to wireless scenarios.

The research described in [DMQ07] presents an alternative approach to extend the limit of Zeroconf beyond the local link. Here, the robustness of existing discovery mechanism is evaluated under increasing failure intensity. However, responsiveness is not covered in particular. The z2z toolkit [Lee+07] combines the Zeroconf with the scalability of DHT-based peer-to-peer networks allowing services to reach beyond local links. z2z connects multiple Zeroconf subnets using OpenDHT. Robustness of service discovery with respect to discovery delay times is addressed in [Oh+04]. The

work of [CG06] describes and compares through simulation the performance of service discovery of two IETF proposals of distributed DNS: Multicast DNS and LLMNR (Link-Local Multicast Name Resolution). The authors propose simple improvements that reduce the traffic generated, and so the power consumption.

In wireless settings, the work of Wirtz [Wir+12] proposes DLSD (DHT-based Localized Service Discovery), a hierarchy of localized DHT address spaces that enable localized provision and discovery of services and data. Another work[Dit+14] proposes a stochastic model family to evaluate the user-perceived responsiveness of service discovery, the probability to find providers within a deadline, even in the presence of faults.

From the review of the related work it can be seen that the reviewed experimental studies related to service discovery were not conducted in the context of the CNs, which we address as scenario. Further, we propose a novel usage of Avahi [17f] in combination with TincVPN [17g], which was not investigated before. Finally, our work conducted the service discovery evaluation in a real deployment scenario to understand its performance and operation feasibility under real network conditions.

2.6 Service Placement

Service placement is a key function of cloud management systems. Typically, by monitoring all the physical and virtual resources on a system, service placement aims to balance load through the allocation, migration and replication of tasks. When reviewing current service placement approaches, we observed that this area of research is generally tackled either as a byproduct of middleware research (i.e., focus is on centralized services and they employ heuristics) or as an application of facility location theory (i.e., facility location problem [Ver11] which is NP-Hard but approximations exist). Regarding the applicability environment, we look at the service placement problem in three different environments: data centre (DC), distributed data centres and wireless networks.

2.6.1 Service Placement in Data Centre Environment

Choreo [LaC13] is a measurement-based method for placing applications in the cloud infrastructures to minimize an objective function such as application completion time. Choreo makes fast measurements of cloud networks using packet trains as well as other methods, profiles application network demands using a machine-learning algorithm, and places applications using a greedy heuristic, which in practice is much more efficient than finding an optimal solution. In [Her10] the authors proposed an optimal allocation solution for ambient intelligence environments using tasks replication to avoid network performance degradation. Volley [Aga+10] is a system that performs automatic data placement across Microsoft data centers. Volley analyzes the logs or requests using an iterative optimization algorithm based on data access patterns and client locations, and outputs migration recommendations back to the cloud service. A large body of work of service placement in DCs has been devoted to finding heuristic solutions [Gha+14].

2.6.2 Service Placement in Distributed Data Centres

When the service placement algorithms decide how the communication between computation entities is routed in the substrate network, then we speak of network-aware service placement, i.e., closely tied to Virtual Network Embedding (VNE).

There are few works that provides service placement in distributed clouds with network-aware capabilities. The work in [SG12] proposes efficient algorithms for the placement of services in distributed cloud environment. The algorithms need input on the status of the network, computational resources and data resources which are matched to application requirements. In [KIH12] authors propose a selection algorithm to allocate resources for service-oriented applications and the work in [AL12] focuses on resource allocation in distributed small data centers. Another example of a network-aware approach is the work from Moens in [Moe+14] which employs an Service Oriented Architecture (SOA), where applications are constructed as a collection of services. Their approach performs node and link mapping simultaneously. The work in [SBL15] extends the work of Moens et al. in wireless settings taking

into account IoT (Internet of Things). Another work is Mycocloud [Dub+15], which provides elasticity through self-organized service placement in decentralized clouds.

The recent work in [Tär+16] simultaneously and holistically take into account rapid user mobility and vast resource cost and capacity heterogeneous infrastructures when placing applications in Mobile Cloud Networks (MCN). Based on their proposed system model, a globally optimal placement of static and mobile applications is designed. Their optimal solution achieves a 25% reduction in cost compared to the naive methods. A recent work of Tantawi [Tan16] uses biased statistical sampling methods for cloud workload placement.

Regarding the service placement through migration, the authors in [Urg+15a] and [Wan+15a] study the dynamic service migration problem in mobile edge-clouds that host cloud-based services at the network edge. They formulate a sequential decision making problem for service migration using the framework of Markov Decision Process (MDP) and illustrate the effectiveness of their approach by simulation using real-world mobility traces of taxis in San Francisco. The work in [Urg+15b] studies when services should be migrated in response to user mobility and demand variation. Another work [Mac+16] proposes a three-layer framework for migrating running applications that are encapsulated either in virtual machines (VMs) or containers. They evaluate the migration performance of various real applications under the proposed framework.

2.6.3 Service Placement in Wireless Networks

To the best of our knowledge, not many works regarding the service placement problem are present in the wireless environment. Some of the works that we find similarities in this thesis, are the following below.

The authors in [Wit10] propose a service placement framework as a novel approach to service placement in wireless ad hoc network. Their SP_i framework takes advantage of the inter-dependencies between service placement, service discovery and the routing of service requests to minimize signaling overhead. They propose two service placement algorithms: one for centralized services with a single instance, and

one algorithm for distributed services with variable number of instances. The SP_i framework employs these algorithms to optimize the number and location of service instances based on usage statistics and a partial network topology derived from routing information. The examine the performance of their algorithms in simulations, emulations and real-world experiments on a IEEE 802.11 wireless testbed.

Another work in [Cab14] tries to optimize the resource selection for service allocation in the contributory computing model. They claim that by using historical availability behavior to select nodes in the system, can lead to higher service availability levels. Thus, user impression is improved and more users and services could be attracted to contributory environments. Furthermore, they design a network-aware methodology to allocate service replicas in a network graph aiming to minimize the distances from each client to its closest service replica. They model the problem as a Facility Location Problem, on which service replicas are facilities (with a set up cost) and the clients in the network are customers. Then, the goal is to minimize the sum of distances and set up costs of the selected replicas.

The work in [Nov+15] analyzes network topology and service dependencies, and combined with set of system constraints determines the placement of services within the wireless network. The authors use a multi-layer model to represent a service-based system embedded in a network topology and then apply an optimization algorithm to this model to find where best to place or reposition the services as the network topology and workload on the services changes. This technique improves the reachability of services when compared to uninformed placements.

The work of Davide et.al [Veg+14] introduces a service allocation algorithm, that from being optimal in computation time, provides near-optimal overlay allocations without the need to verify the whole solution space. This work is related to ours since it is done in the `Guifi.net` CN. Their algorithm uses static data from the `Guifi.net` to identify node traits and propose several algorithms that minimize the coordination and overlay cost along the network.

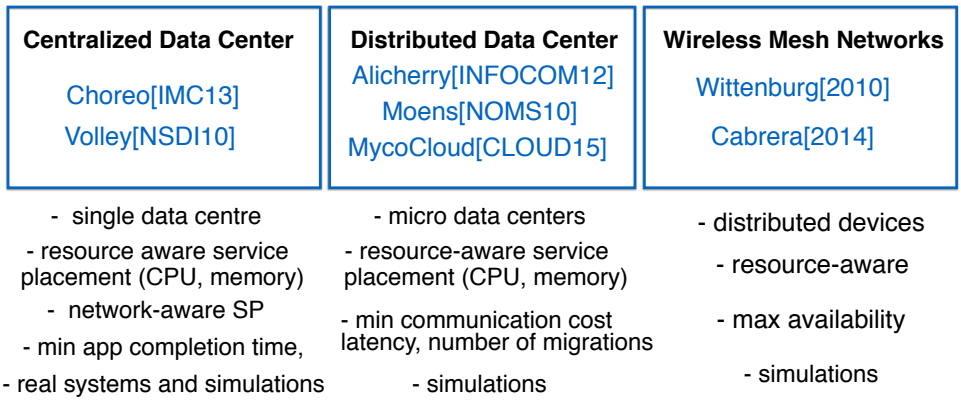


Figure 2.7: Classification by applicability environment

2.7 Discussion

We looked at the service placement in three different environments: data centre (DC), distributed data centre and wireless environment. Figure 2.7 depicts some of the selected service placement approaches classified by their applicability environment.

The service placement in DC network is approached by selecting the racks and servers within DC where the services will be placed. Usually, in DC environment the decision on service placement is made taking into account the computational resources i.e., resource-aware service placement. CPU (Central Processing Unit) and memory are considered as the main resources. Since the demand for bandwidth between servers inside a data center is growing faster than the demand for bandwidth to external hosts [LaC13], some network-aware approaches came recently. Their goal is mostly to minimize the application completion time. Examples include Hadoop jobs, analytic database workloads, storage/backup services, and scientific or numerical computations.

Most of the work in the DC environment is not applicable to our case because we have a strong heterogeneity given by the limited capacity of nodes and links, as well as asymmetric quality of wireless links. The difference/asymmetry in the link capacities across the CNs makes the service placement a very different problem than

in a mostly homogeneous cloud DC. As explained in Chapter 5, our measurement results demonstrate that 25% of the links have a symmetry deviation higher than 40% [Sel+16b] [Sel+17].

In the *distributed data centres*, service placement answers the question of optimally selecting the data center in a distributed cloud and within data center selecting the racks and servers. In this environment, service placement approaches are resource and network-aware. Minimizing the communication cost or number of migrations between data centers is a key issue. Not many real systems are present in the research community.

Most of the work in the distributed data centres consider micro-DCs (i.e., small scale data centres), where in our case the CN *micro-clouds* consists of constrained/low-power devices such us home gateways. Furthermore, in our case we have a partial information regarding the computational devices and their approaches are not fully applicable to our environment.

In *wireless mesh networks*, to the best of our knowledge, not many working systems are present. Service placement is approached by selecting the distributed devices in the network where the service will be running. Mostly the approaches in this environment are resource-aware and they consider historical data (e.g., device availability) for placement decision.

The focus of our work in this thesis is on network-aware service placement methodologies in distributed and heterogeneous wireless CNs. Our goal is to to design network-aware and low-complexity service placement heuristics for CN *micro-cloud* services, in order to improve the user QoS and QoE.

3

Performance Assessment of Micro-Cloud Services

Internet access is often considered the main service of community networks (CNs), but the provision of services of local interest within the network is a unique opportunity for CNs, which is currently predominantly unexplored. The consolidation of today's cloud technologies offers CNs the possibility to collectively build CN *micro-clouds*, building upon user-provided networks, and extending towards an ecosystem of cloud services. Internet cloud services have equivalent alternatives that are owned and operated at the community level; in other cases, however, there are no locally driven alternatives, yet. Possible reasons for the absence of these community-owned services can be found in the difficulty to deploy such services and the shortage or lack of individuals, organisations, or companies interested in the commercial operation of these services. Furthermore, the adoption of the CN *micro-cloud* services requires carefully addressing the service deployment and performance requirements.

In this chapter, first we present the current state of service deployment in the `Guifi.net` CN and identify the popular services within this network. Then, we

Table 3.1: List of network-focused Guifi.net services in Catalonia (2016)

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

conduct real deployments of the CN *micro-cloud* services in the `Guifi.net` and evaluate services such as distributed storage, live-video streaming and service discovery. This deployment experience supports the feasibility of CN *micro-clouds* and our measurements demonstrate the performance of services running in these clouds. The work in this chapter addresses the research question **Q1**.

3.1 Current State of Service Deployment in Guifi.net

To obtain the dimension of the current situation, we analyse the list of services published (i.e., publicly announced) by the `Guifi.net` CN. We do so by means of the list of services available on the `Guifi.net` web page for the Catalonia region of Spain, the origin and most dense location of this network [16k].

Tables 3.1 and 3.2 indicate the network-focused and user-focused services, respectively, of `Guifi.net` and the proportion of each service in the services offered. We consider that the number of instances of a service implies the demand of the service inside the network. Comparing the tables, we notice that the services related to the network operation itself slightly outnumber the services intended for end-users. Considering that network management is of interest only to a fragment of the network members compared to user-focused services, which could be of interest to all users, we would expect user-focused services to be more developed. Moreover, the most

Table 3.2: List of user-focused Guifi.net services in Catalonia (2016)

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

frequent of all the services, whether user-focused or network-focused, are the proxy services [DMN17]. Specifically for the user-focused services, the percentage of Internet access services (i.e., proxies and tunnel-based) is higher than 55%, confirming that the users of `Guifi.net` are typically interested in accessing the Internet. We can also claim that there is a diverse set of services inside `Guifi.net`, even though their adoption is overshadowed by Internet access.

It is important to point out that this situation is not unique to `Guifi.net`. Other CNs exhibit similar situations, where the network is typically used to access the Internet, and the few services available within the CN are similar to those available in `Guifi.net`. For instance, Elianos et al. [Eli+09] presented similar information regarding AWMN [16l]. The authors mainly focused on user-oriented services, which are quite similar to the `Guifi.net` services. The most popular services are web host-

ing, data storage, VoIP, and video streaming.

The services provided by `Guifi.net` can be categorised under cloud computing service models, though not following the traditional cloud elastic on-demand service approach: *IaaS* (Infrastructure as a Service), *PaaS* (Platform as a Service) and *SaaS* (Software as a Service). Concerning *IaaS*, following the global trend, the popularity of virtualisation technologies is rising in `Guifi.net`. Currently, almost all critical services are run on virtualised environments, frequently using Proxmox [16m]. `Guifi.net` also provides specific hardware infrastructure and software, supporting virtual networks and tunnelling. Additionally, some efforts have been made in the past to provide the end users with tools to help them with the deployment and expansion of the CN from the software and services perspective. This was the case of Guinux [16n], a GNU/Linux distribution for end users allowing them to deploy servers with services useful for community networking, namely Proxy, DNS (Domain Name System), and SNMP (Simple Network Management Protocol) graphs servers. Similarly to *PaaS*, a diverse set of services has been deployed, such as automated node configuration, user authentication, service monitoring (i.e., servers and network), and an on-line service directory as well as network information and administration databases. Finally, taking into account the *SaaS* model in the context of CNs, data storage services have been sporadically deployed by enthusiastic users who wanted to share some of their content (e.g., pictures, documents, etc.) with the rest of the community [Sel+14]. In some cases, users have also enabled uploading to folders, allowing other users to upload their files for sharing with the community. Despite this, it should not be considered a data storage service for end users. Moreover, `Guifi.net` users have developed GuifiTV [160], a project initially conceived to harmonise the captured video formats and the content from seminars and workshops, which later included video streaming services.

The services described above are representative examples of those usually deployed in CNs. Nevertheless, both network-oriented and user-oriented services are centralised and offered by individuals. Distribution and decentralisation are concepts that are closely related to the CN philosophy; nonetheless, since centralised solutions are

generally much easier to develop and deploy, in most of the cases they end up being implemented according to the classical client-server approach. As a result, basic cloud service requirements, as described in [MG11], are not fulfilled. Most importantly, there is no common pool of resources but instead a set of separate resources, since the same services are deployed independently and not coordinated in a common way. As a result, service coordination and resource sharing mechanisms are the first milestones towards creating cloud services for CNs.

Our effort targets fostering the deployment of the CN *micro-clouds* and cloud-based services on top of the CNs. Based on the experiences of current CNs, providing end users with the appropriate services has proven to be an effective way of encouraging them to use, provide, and promote these services.

3.2 Experimental Environment

In this section, we explain our work on deploying and evaluating the CN *micro-cloud* services. In order to have a realistic CN setting, which includes geographically distributed nodes, we have used the Community-Lab [17k] testbed nodes for setting up our CN *micro-cloud* infrastructure. Community-Lab is a distributed infrastructure developed by the Community Networks Testbed for the Future Internet (CONFINE) project [Bra+13], where researchers can deploy experimental services on several nodes deployed within merged CNs. The Community-Lab testbed is currently deployed on the nodes from `Guifi.net` and the AWMN CNs. This allows us to run our experiments on nodes from both the CNs, which has the added advantage that we can test how *Cloudy* services perform over large geographical distances.

In the Community-Lab, `Guifi.net` and the AWMN are connected on the IP layer though the Federated E-infrastructure Dedicated to European Researchers (FEDERICA) [17l], enabling the federation of both networks. Most Community-Lab nodes are built with a Jetway or Minix Neo Z64 devices that are equipped with an Intel Atom N2600 CPU, 4 GB of RAM and 120 GB SSD (Solid-state drive). Nodes in the Community-Lab testbed run a custom firmware (i.e., based on OpenWRT) provided by CONFINE, which allows running several container instances (i.e., slivers) on one

node simultaneously implemented as LXC or Docker containers. We deploy the *Cloudy* distribution in these containers on the nodes of the Community-Lab.

We report results from our experiments related to *distributed storage* service, *live-video streaming* service and *service discovery*.

We choose to experiment with the *distributed storage service*, based on Tahoe-LAFS. Tahoe-LAFS has features that are very important for the CN environment, such as data encryption at the client side, coded transmission and data dispersion among a set of storage nodes. This approach of Tahoe-LAFS results in high availability (i.e., even if some of the storage nodes are down or taken over by an attacker, the entire file system continues to function correctly, while preserving privacy and security). Furthermore, in order to compare and understand the impact of homogeneous network and hardware resources on the performance of Tahoe-LAFS, we evaluate it also in the *Microsoft Azure* [C+11] commercial cloud platform.

Regarding the *live-video streaming service*, we are experimenting with the PeerStreamer. PeerStreamer is an open source live P2P video streaming service, and mainly used in the *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.

Finally, we choose to experiment with the *service discovery*, based on Avahi-TincVPN and Serf, since it is a crucial building block of *Cloudy* that enables distributed services to be orchestrated in order to provide platform and application services. Our goal is not to compare Avahi-TincVPN and Serf, as both of them are available in the *Cloudy* and can be used by users for different scenarios. One of them is lightweight and fast (i.e., Serf), and the other is not scalable and is suitable and preferable for small scale networks (i.e., Avahi-TincVPN).

3.3 Distributed Storage

After basic connectivity, storage is the most general service being fundamental for cloud take-up in CN scenarios. Allowing users in a CN to share and use the storage of other users in a reliable, secure, and privacy-preserving way, is of great importance.

For this reason, we use Tahoe-LAFS as a main storage service in *Cloudy*. Understanding the performance (i.e., successful operation) of Tahoe-LAFS from an experimental scenario that represents real use case situations is highly important because it informs the end users regarding the application performance they will receive. Such performance results are needed to pave the way for bringing applications such as Tahoe-LAFS as well as other applications into CNs.

3.3.1 Tahoe-LAFS

Tahoe-LAFS [WW08] is a decentralised storage system with provider-independent security. This feature means that the user is the only one who can view or modify disclosed data. The data and metadata in the cluster is distributed among servers using erasure coding and cryptography. The erasure coding parameters determine how many servers are used to store each file, which is denoted as N, and how many of them are necessary for the files to be available, denoted as K. The default parameters used in Tahoe-LAFS are $K = 3$ and $N = 10$ (i.e., 3-of-10). The Tahoe-LAFS cluster consists of a set of *storage nodes*, *client nodes*, and a single coordinator node called the *introducer*. The storage nodes connect to the introducer and announce their presence, and the client nodes connect to the introducer to obtain the list of all connected storage nodes. The introducer does not transfer data between clients and storage nodes, but the transfer is done directly between them. The introducer is a first-point-of-contact for new clients or new storage peers, because they need it for joining the storage grid. When the client uploads a file to the storage cluster, a unique public/private key pair is generated for that file, file is encrypted on the client side prior to upload, erasure coded, and distributed across storage nodes (i.e., with enough storage space) [WW08]. The location of erasure coded shares is decided by a server selection algorithm that hashes the private key of the file to generate a distinct server permutation. Then, servers without enough storage space are removed from the permutation, and the rest of the servers are contacted in sequence and asked to hold one share of the file. To download a file, the client asks all known storage nodes to list the number of shares of that file they hold, and in the subsequent rounds (i.e., second round-trip),

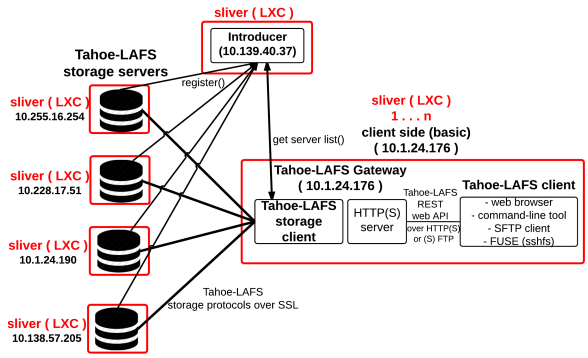


Figure 3.1: Tahoe-LAFS deployed in the Community-Lab testbed

the client chooses which share to request based on various heuristics such as latency, node load, etc.

To deploy Tahoe-LAFS in the Community-Lab nodes, we use the *Cloudy* distribution, which contains Tahoe-LAFS, and place the *introducer*, *storage*, and *client* nodes of Tahoe-LAFS inside the LXC and Docker containers of the testbed. Figure 3.1 shows the resulting Tahoe-LAFS architecture used by our experiments in the Community-Lab testbed.

3.3.2 Experiment Setup

All tests were conducted using the IOzone [16p] cloud storage benchmark. IOzone is a filesystem benchmark tool, which generates the cloud storage workload and measures various file operations. The benchmark tests file input/output (I/O) performance of many important storage-benchmarking operations, such as read, write, re-read, re-write, random read/write, etc. We run all 13 IOzone tests and vary the file size from 64 KB to 128 MB and record length of 128 KB. We use a FUSE (Filesystem in Userspace) kernel module in combination with SSHFS (SSH Filesystem), an SFTP client that allows filesystem access via FUSE, to mount a Tahoe-LAFS directory to the local disk of the client. Results presented in this work with regard to performance are measured in MB/s and are referred to as operation speed (i.e., read and write performance).

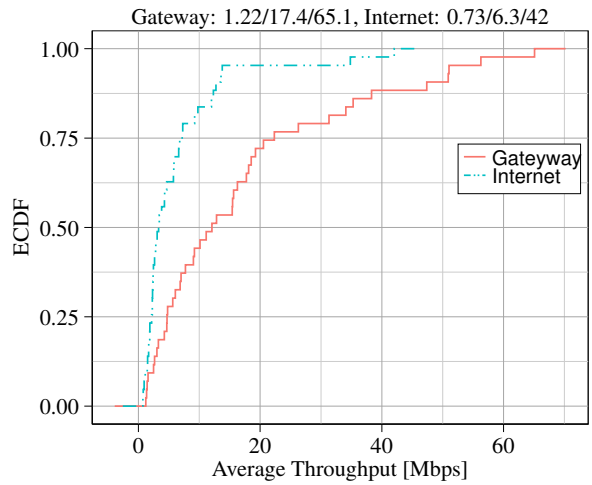


Figure 3.2: ECDF of the average throughput to the gateway/Internet.

Tests with concurrent reading and writing were not conducted.

We assess the performance of Tahoe-LAFS when it is deployed in CN *micro-clouds* and in the commercial *Microsoft Azure* cloud platform [17m] .

In CN *micro-clouds*, two sets of tests were conducted. One is when the writes/reads are initiated from a client that has the best connectivity in the network, such as best RTT and throughput to the other nodes, and this is the *baseline* case; the other is when they are initiated from a client node, which is the farthest to the other nodes in the network (i.e., in terms of number of hops, RTT, etc.), and this is referred as a *set1* case in the graphs.

To better understand the impact that the network imposes on a CN environment, we established a Tahoe-LAFS cluster of 30 available nodes, geographically distributed in the `Guifi.net` CN and connected to the outdoor routers (ORs) of the network. We characterize the throughput of the nodes by connecting to them hourly (i.e., 10 captures obtained per day), during the whole month of February 2015 (i.e., 300 captures obtained in total). A *capture* is an hourly network snapshot that we take from the network regarding its links. Figure 3.2 shows the Empirical Cumulative Distribution Function (ECDF) of the average throughput to the gateway (i.e., `Guifi.net`

Table 3.3: Microsoft Azure cluster characteristics

Number of VMs	20
Size	A1(1 core, 1.75 GB memory)
Region	West Europe (WE)
Throughput between VMs	220-230 Mbps
RTT between VMs	1-2 ms

proxy) and the Internet. Internet values are measured using a server located outside of `Guifi.net` network. On the top of the figure, the minimum/mean/maximum values are given. As shown in the Figure 3.2, the overall mean throughput to the gateway is 17.4 Mbps and 6.3 Mbps to the Internet. This reveals that there is a bottleneck inside the portion of `Guifi.net` connecting the gateways with the Internet.

In the *Microsoft Azure* cloud, two sets of tests were conducted. One is when the reads/writes are initiated from a client node located in the same *Microsoft Azure* cluster which is referred to as local read/write (i.e., *baseline*); the other is when they are initiated from a client node located in a different location (i.e., CN), which is referred to as remote read/write (i.e., *set1*). *Microsoft Azure* cluster consists of 20 VMs in total. The average throughput from the CN nodes to the *Microsoft Azure* cluster nodes is 6.3 Mbps. Table 3.3 depicts node and network characteristics of the used *Microsoft Azure* cluster.

3.3.3 Experimental Results in Community Networks

Figure 3.3 and Figure 3.4 show the best and worst client write/read performance. Figure 3.5 depicts the summary of all tests performed with the IOzone benchmark. Median, first and third quartile values of read and write operations are plotted in the Figure 3.5. A few observations are noted below.

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

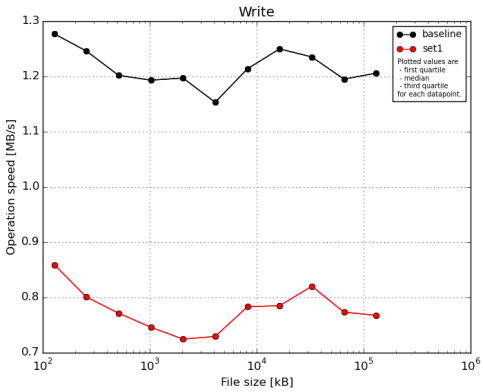


Figure 3.3: Write performance in CNs

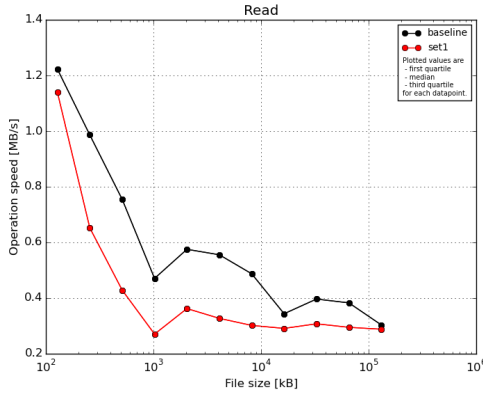


Figure 3.4: Read performance in CNs

terms of RTT to the other nodes than the *set1* client.

- In terms of write performance, the *baseline* client performs better. Write performance for the baseline is higher and more stable than the read performance. As the file size increases, the write performance of the baseline client slightly decreases (i.e., minimum throughput achieved is 1.15 MB/s when writing a 4 MB file). The higher throughput (i.e., 1.28 MB/s) is achieved when writing a small file (i.e., 128 KB file), as shown in Figure 3.3.

  It is interesting to note that when writing smaller files, Tahoe-LAFS performs better, and this can be attributed to the fact that the default stripe size of Tahoe-LAFS is well optimised for writing small objects (i.e., the stripe size determines the granularity at which data is being encrypted and erasure coded). The same thing happens with the *set1* client, where the maximum write throughput achieved is 0.86 MB/s when writing a 128 KB file, and the minimum write throughput is 0.72 MB/s when writing a 2 MB file. Furthermore, write performance is affected by another factor; when writing new objects, Tahoe-LAFS generates a new public/private key, which is a computationally expensive operation.

- Read operations are accomplished by submitting the request to all storage

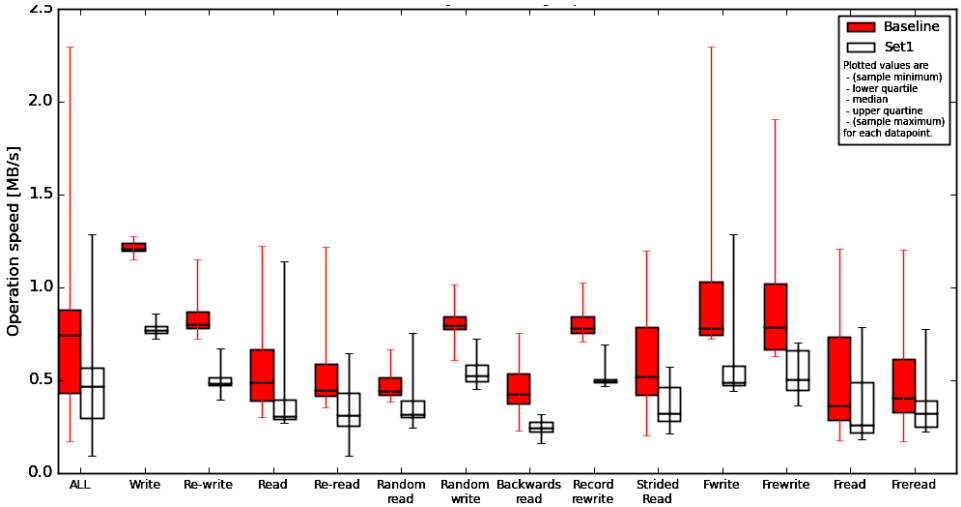


Figure 3.5: Summary of all storage benchmark operations in CNs

nodes simultaneously; hence, the relevant peers are found with one round-trip to every node. The second round-trip occurs after choosing the peers from which to read the file. The intention of the second round-trip is to select which peers to read from, after the initial negotiation phase, based on certain heuristics. When reading from the storage nodes, the performance of both clients drops significantly as the file size increases as shown in Figure 3.4. This is because when reading a file of 128 MB, a client must contact more Tahoe-LAFS storage peers in order to complete the shares of the file. In addition, reading the file system meta-object (i.e., the mutable directory objects) every time an object is accessed results in overhead, thus influencing the results.

- Figure 3.5 shows the summary of all tests performed with the IOzone benchmark in CNs. The benchmark tested file I/O performance for the 13 operations as shown in Figure 3.5. As illustrated, the *baseline* client performs better than the *set1* client, reaching an average operation speed of 0.74 MB/s for all 13 tests performed.

3.3.4 Experimental Results in Microsoft Azure Cloud

Figures 3.6 and Figure 3.7 show the local and remote client write/read performance in the *Microsoft Azure* cloud platform. Median, first and third quartile values of read and write operations are plotted in Figure 3.8. A few observations are noted below.

- Write performance for the *baseline* client is better and more stable than the read performance. As the file size increases, the read performance of the *baseline* client decreases up to 0.7 MB/s when reading a 128 MB file. This is because, when reading bigger files (i.e., 128 MB), a client has to contact more Tahoe-LAFS storage peers in order to complete the shares of the file.

- When accessing (i.e., reading and writing) the *Microsoft Azure* cluster from a remote location (i.e., *set1* client), the performance drops significantly. The client accessing the cluster is one of the CN nodes. The performance drop is related to the problem of proxy (i.e., gateway) selection inside the network [Dim+17]. Currently, the criteria when the remote client is choosing a proxy does not include the proxy load, network congestion and proxy performance. This results in a performance drop (i.e., less than 1 MB/s read and write operation).

- Figure 3.8 shows the summary of all tests performed with the IOzone benchmark in *Microsoft Azure* cloud. As shown, the *baseline* client performs better than the *set1* client, reaching an average operation speed of 1.4 MB/s for all 13 tests performed. The average operation speed of the *set1* client for all tests is 0.7 MB/s.

3.4 Live-video Streaming

Among the services that are very appealing in CNs, P2P live streaming is an important candidate, as can be seen by the growing success and usage of commercial systems such as PPLive, SopCast [16q], etc. P2P live streaming systems allows users to watch live streams such as events or television channels over a network, granting anyone

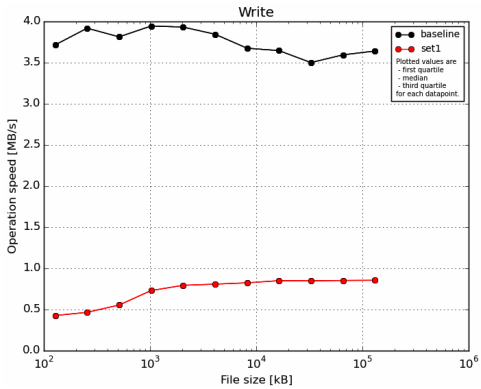


Figure 3.6: Write performance in Azure cloud

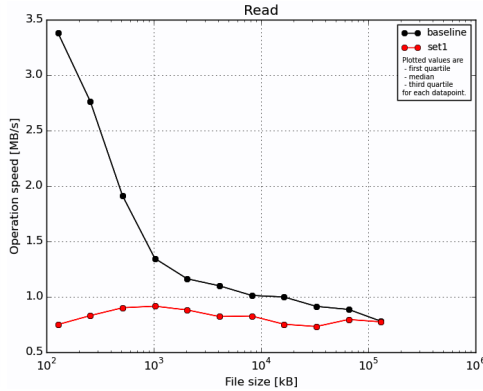


Figure 3.7: Read performance in Azure cloud

to become a content provider. To enable these types of services within CN nodes is very challenging, since CNs are diverse and dynamic networks with limited capacity of wireless links and often low-resource devices. Streaming services, however, have high demands of bandwidth, they require low and stable latency and only withstand low packet loss [Tra+12].

We evaluate the performance of PeerStreamer as a P2P live streaming service deployed over geographically distributed real CN *micro-cloud* nodes. We then study the effects of different parameters of PeerStreamer on its performance in the CN environment.

3.4.1 PeerStreamer

PeerStreamer [16c] 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 (i.e., by default, this is one frame of the video). At the beginning of the streaming process, these chunks are all from the same peer (i.e., since only one peer is the source),

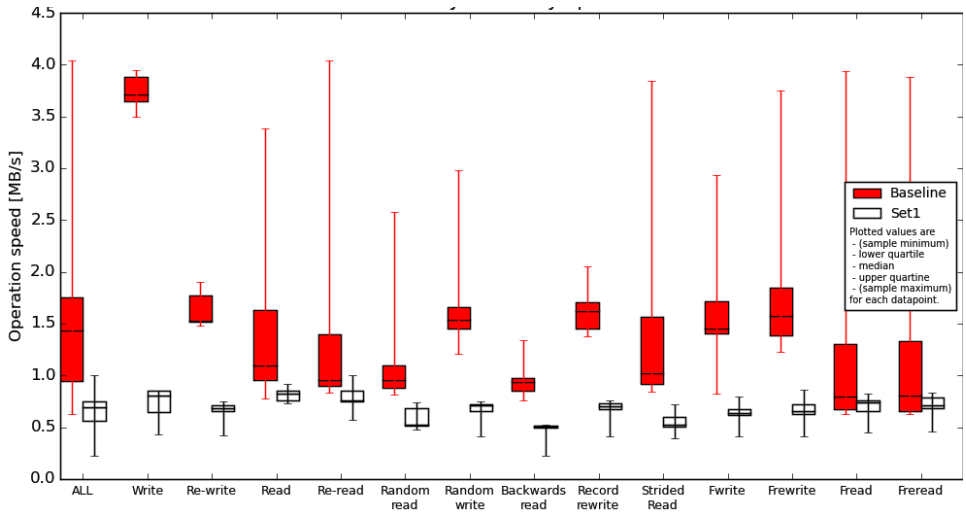


Figure 3.8: Summary of all storage benchmark operations in the Azure cloud

then the source sends m copies of the chunks to the random peers (i.e., $m = 3$ by default), creating an overlay topology with all peers. Then, the peers exchange chunks between them. The whole architecture and vision of PeerStreamer is described in detail in [B+11].

3.4.2 PeerStreamer Assumptions and Notation

We call the CN 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 (i.e., next hop is known). We can model the underlay graph as:

$$G^{ug} = (S, L^{ug}) \tag{3.1}$$

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

In the *micro-cloud* level (um) we have a set of outdoor routers (OR) that are con-

nected to each other in the same *micro-cloud*,

$$G^{um} = (OR, L^{um}) \tag{3.2}$$

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 (i.e., 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 (i.e., randomly) of the other peers as a set of neighbours that are called $N_i(t)$. The peer P_i exchanges video frames (i.e., 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 service, also represented as graph and called *overlay*. The overlay built by PeerStreamer is time-varying directed graph:

$$G^{og}(t) = (P_{set}, L^{og}(t)) \tag{3.3}$$

where P_{set} is the set of peers and

$$L^{og}(t) = (P_i, P_j) : P_j \in N_i(t) \tag{3.4}$$

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 the PeerStreamer.

3.4.3 Experiment Setup

For the experimental study, 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 (i.e., as home gateways, set-top-boxes etc.) or within other infrastructures around each city.

Two CNs (i.e., `Guifi.net` and AWMN) are connected on the IP layer enabling network federation. The nodes of Ninux CN in Italy are not connected to them, there-

fore we experiment with them separately. In our experiments the nodes from UPC (Technical University of Catalonia) are a subset of `Guifi.net` nodes which are distributed in the UPC campus in Barcelona. We use these nodes as a *baseline* in order to be able to better understand the effects of the network in the reliable operation of live-video streaming service.

Our experimental evaluation is comprised of 55 physical nodes distributed across Europe, among the working nodes available from three CNs. Table 3.4 shows the number of nodes used in three CNs, their location and type of nodes deployed.

In our experiments we connect a live streaming camera (i.e., 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 network 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. Further, the source node initially sends three copies (i.e., $m = 3$) 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. On the quality of service side, we measure the *number of chunks that are received* by peers and the *chunk loss percentage*. We choose these metrics 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 peer to understand the quality of the images that the edge nodes show to the users.

3.4.4 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

Table 3.4: Nodes in the cluster and their location

Nr. of nodes	CN	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

Table 3.5: Summary of the scenario parameters

Scenario	1 and 2
Total number of nodes	55
Groups of nodes	UPC, Guifi.net, AWMN, Ninux
Experiment 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

our findings with a scenario reflecting different parameters of PeerStreamer usage, in order to understand possible improvements of the overlay network created by the PeerStreamer instances. The parameters used in the scenarios are summarized in Table 3.5.

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 the other networks, we deliberately executed the experiment apart from the other CNs, in order to understand different CN 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 networks.

Each experiment is composed of 20 runs, where each run has 10 repetitions, and averaged over all the successful runs (i.e., 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.

Further, we 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. 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 with the churn rate of nodes. This happens in the CNs because most of the nodes are connected wirelessly and their connectivity depends on many factors (i.e., 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 CNs.

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 behaviors of the PeerStreamer algorithms such that the overlay network that it constructs can be optimized. 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 (i.e., $TopoKeepBest$); and the addition of the peers that can be selected to the neighborhood by extending the default RTT (i.e., 10 ms) of the peer selection metric to 20 ms [B+11].

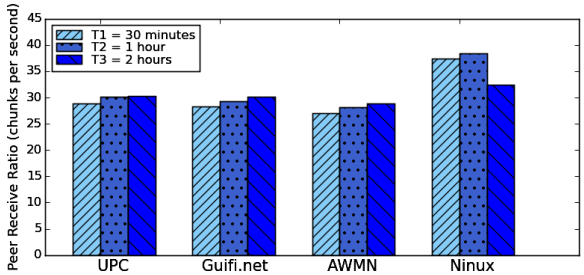


Figure 3.9: Average peer receive ratio

3.4.5 Experimental Results

Figure 3.9 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 (e.g., `Guifi.net` and AWMN) in relation to the source, receive less chunks than the closer groups (i.e., UPC). 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.

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 (i.e., live TV channel stream), in which Source 2 sends around 55 chps, 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 of this group of nodes, where the loss of data is more visible when dealing with longer times.

Figure 3.10 shows the average chunk loss for each group of peers. Figure reveals that the loss is greater for shorter time-frames (e.g., loss in UPC is 7%, `Guifi.net` 9% and AWMN 13%) and are amortized for longer time-frames (e.g., loss in UPC 2%, `Guifi.net` 3% and AWMN 7%). We also notice that distant groups from the source

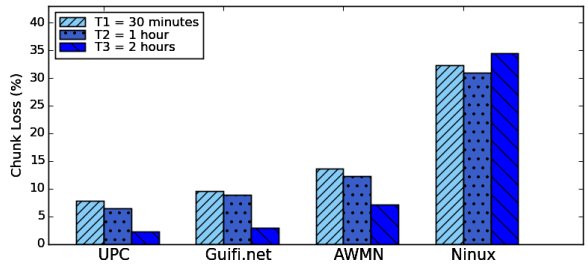


Figure 3.10: Average chunk loss

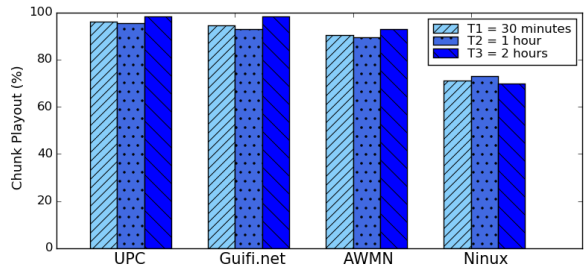


Figure 3.11: Average chunk playout

are more affected by the diminished rate of chunks received, which demonstrates the influence the network has to the amount of data lost. As for the Ninux group, the network is more volatile since there is a higher packet loss (i.e., 34% loss).

Figure 3.11 illustrates the chunk playout on the peers side. The closer groups to the source, display more chunks than farther groups. We also notice that the longer time-frames have on average a better chunk playout because more chunks arrive on time to be displayed (e.g., UPC 98%, `Guifi.net` 98%, AWMN 92%, Ninux 71%).

Figure 3.12 demonstrates the chunk loss gathered during 30 minutes of experiment, with different parameters given to the PeerStreamer. The parameters shown (i.e., TopoKeepBest, RTT = 20 ms and m = 5) have been selected in order to predict the behavior and improvements that PeerStreamer can have when executed in CN environment.

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

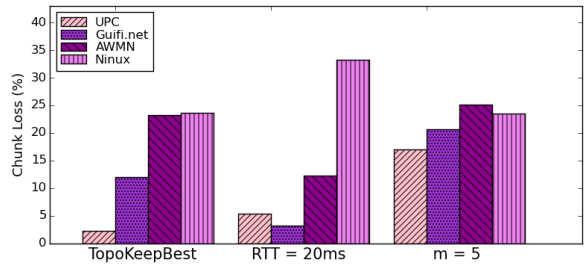


Figure 3.12: Average chunk loss with different parameters

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 chunk to the peers (i.e., not shown in the figure). We also notice that, keeping the best neighbors on topology overlay updates, decreases chunk loss in the groups 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, when keeping the best nodes on topology updates there is a greater improvement (i.e., 23% in loss, comparing with default parameters where we got 32% loss). We also show that there is improvement when changing the number of chunk copies Source 2 sends to the peers (i.e., $m = 5$).

3.5 Service Discovery

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

Table 3.6: Nodes, their location and RTT from the client node

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

3.5.1 Experiment setup

For our service discovery experiment, we use 25 nodes spread between two CNs as shown in Table 3.6. We use 20 nodes from the `Guifi.net` CN, where 13 of the nodes are located in the city of Barcelona (UPC) and 7 of them are located in the Catalonia region of Spain. From AWMN we use 5 nodes, which are located in Athens, Greece.

Figure 3.13 shows the throughput of three categories of nodes in our cluster. Network measurements have been obtained connecting to each node and measuring the average aggregated throughput from the client nodes. For some scenarios we use more than one client node. The clients are located in the `Guifi.net` CN. Connections to the nodes have been done every half an hour (i.e., 12 hours per day), during the entire month of January 2015, where 720 captures are obtained for each category of nodes. All obtained captures are plotted in the graph. The average throughput obtained for the UPC nodes is 10.5 Mbps, for the `Guifi.net` nodes is 4.8 Mbps, and AWMN nodes is 1.9 Mbps.

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

We run the discovery requests from three client nodes that are searching and locating service instances. All other nodes acted as *service providers* responding to discovery requests. All service providers are spread between two CNs. Discovery times

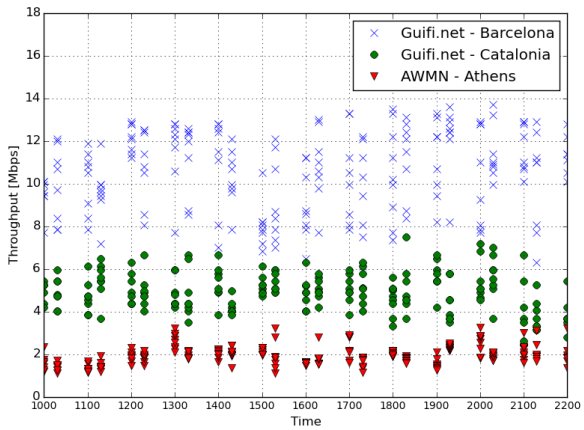


Figure 3.13: Throughput of the nodes

are measured on the clients directly before the request was sent and directly after responses were received to measure user-perceived responsiveness. No nodes joined or left the network; therefore, no configuration on the network layers occurred during measurements which would interfere with the discovery operation. We consider the discovery successful when all instances have been discovered. Discoveries were aborted and considered failed if no responses arrived until an experiment ran with a deadline of 25 seconds in the Community-Lab testbed. This value was chosen because in Zeroconf [15], the time between retries doubles after each retry to reduce the network load. Depending on the scenarios that we consider, each service discovery experiment is comprised of several runs (i.e., normally between 15 to 20) and is averaged over all the successful runs. Each run consists of 20 repetitions.

In Avahi, when publishing and discovering, no entries are cached per interface; thus, no caching is used. After service discovery, a client should have enough information to contact a service instance. Hence, discovery in our case means resolving the IP address and port for every service instance. During the experiments we use various *Cloudy* services to publish and discover as summarised in the Table 3.7.

Table 3.7: Cloudy services used for the experiments

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

3.5.2 The Studied Scenarios

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

Scenario 1: Single service discovery. Our first goal is to measure the responsiveness of single service discovery. In this scenario, the service network consists of one client and one provider. The client is allowed to wait up to 10 seconds for a positive response. This is a common scenario for service discovery and can be considered the *baseline*. Only one answer needs to be received and there is enough time to wait for it. In this case, the client discovers a Tahoe-LAFS distributed storage service and contacts the service. For this scenario, a service provider from the `Guifi.net` CN is considered. Both Avahi-TincVPN and Serf are used for this scenario.

Scenario 2: Timely service discovery of the same service type. Service networks are populated with multiple instances of the same service type. The clients need to discover as many instances as possible and will then choose one that optimally fits their requirements. The faster discovery is better. In this scenario, we have one service client and 25 service providers (i.e., from `Guifi.net` and the AWMN). The discovery is successful if all provided service instances of the same type have been discovered. We measure how responsiveness increases with time. The faster we reach a high value, the better. In this scenario, the providers publish one or more PeerStreamer live-video

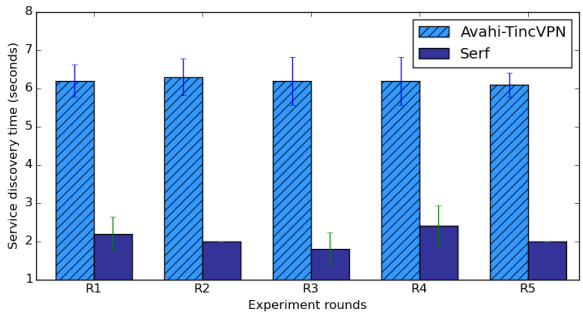


Figure 3.14: Single service discovery time (Scenario 1)

streaming services. The client waits 15 seconds to receive responses. In this scenario only the service discovery based on Serf is used. The total number of services published by providers is 40.

Scenario 3: Client perceived responsiveness. In this scenario, we have three service clients and 20 service providers that publish five popular *Cloudy* services such as PeerStreamer, Tahoe-LAFS, Syncthing, DNSService, and OpenVZ. The total number of *Cloudy* services published is 23 (i.e., 7 PeerStreamer services, 3 Tahoe-LAFS services, 6 Syncthing services, 3 DNSServices and 4 OpenVZ services). The clients are located in different parts of the `Guifi.net`, and they need to discover 23 instances of different service types. Considering the dynamic environment of the `Guifi.net` CN, the discovery is successful if all clients discover the same number of services (i.e., 23) published by service providers. For this scenario, the service discovery based on Serf is used. The clients are allowed to wait 20 seconds.

3.5.3 Experimental Results

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

Scenario 1: Single service discovery. The discovery of a single service instance within 10 seconds proved to be reasonably responsive. This experiment is comprised of 15 runs, where each run has 20 repetitions. In Figure 3.14, the standard deviation error bars per round are plotted on the mean values obtained. The values are obtained using

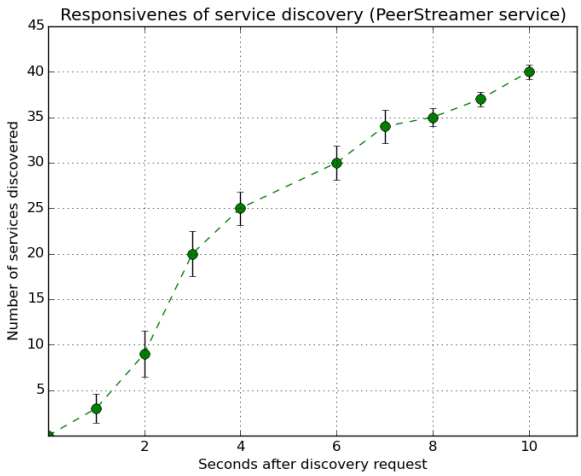


Figure 3.15: Responsiveness of service discovery (Scenario 2)

Avahi-TincVPN and Serf. Due to the efficient and lightweight gossip protocol that Serf uses, the discovery time is decreased for $3x$, reaching an average of 2 seconds for a single service discovery compared to the Avahi-TincVPN combination that reaches 6 seconds.

Scenario 2: Timely service discovery of the same service type. Figure 3.15 illustrates that the discovery of services increases rapidly with the time. The standard deviation error bars are plotted on the mean values. In the first 6 seconds the client discovers 75% of the published services, which is equal to 30 PeerStreamer video-streaming services. The last 25% of the services are discovered from seconds 6 to 10. These 10 services are from the AWMN CN. The eventually consistent gossip model of Serf with no centralised servers allows the client to discover in a very fast and extremely efficient way all the nodes for a PeerStreamer service based on the tags their agent is running. However, the structure and diameter of the CN graph (i.e., topology), fluctuations in the network due to load, and faults can increase the discovery time [Veg+12].

Scenario 3: Client perceived responsiveness. Figure 3.16 demonstrates the number of services discovered by three clients. The standard deviation error bars are plotted on the mean values. As shown, three clients perceive a different number of ser-

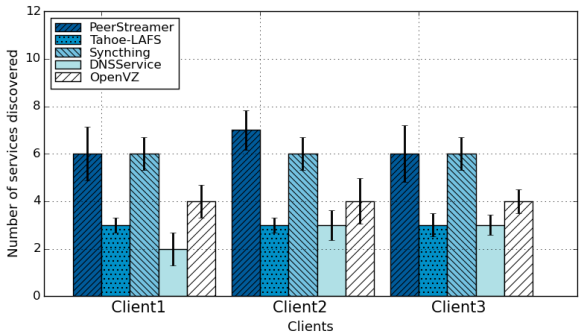


Figure 3.16: Number of Cloudy services discovered by clients (Scenario 3)

vices. Only Client2 discovers all services (i.e., 23 services). Client1 is missing 1 PeerStreamer service and 1 DNSService. Client3 is missing 1 PeerStreamer service. Missing services can be subject to the high diversity in the the quality of wireless links, the availability of nodes and the location of client nodes. Heterogeneous low-resource hardware and packet loss between nodes also can impact the service discovery performance.

3.6 Discussion

Distributed Storage: We evaluated how the Tahoe-LAFS storage system performs when it is deployed over distributed CN *micro-cloud* nodes in a real CN. In addition, Tahoe-LAFS was deployed in the *Microsoft Azure* commercial cloud, to compare and understand the impact of homogeneous network and hardware resources on the performance of the Tahoe-LAFS storage service.

In CN *micro-cloud* environment, we observed that the write operation of Tahoe-LAFS resulted in better performance than the read operation. Read operation is a more expensive operation which is done in two round-trips. In the first round-trip the relevant peers holding the shares are found, and the second round trip occurs after choosing the peers from which to read. Specifically for the CN environment such as `Guifi.net`, successfully running the distributed storage as Tahoe-LAFS is a trade-off between the replication settings and read performance. Keeping the default

Tahoe-LAFS replication setting 3-of-10 (i.e., $N = 10$, $K = 3$) and workload size up to 128 MB works satisfactorily when having up to 30 nodes in the network. In order to use the storage service with bigger workloads, taking network characteristics into account when deciding about the placement of storage nodes is a must. Chapter 4 and Chapter 5 addresses this issue.

In the *Microsoft Azure* cloud platform, Tahoe-LAFS read operation achieved better performance, where the reading from multiple nodes benefited from the homogeneity of the network and nodes in DC environment. Further, we observed that accessing DC services from the CN environment, requires a "smart" client proxy selection for better performance.

A general important result from the experiments is that, Tahoe-LAFS performed correctly in uploading and retrieving all the different file sizes under the challenging conditions of the CNs, which make Tahoe-LAFS a promising application to consider for preserving privacy, and secure and fault-tolerant storage in the dynamic environment of CNs.

Live-video Streaming: 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.

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 neighbors with bad connectivity. We also found that, 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 in the CNs are with high diversity in bandwidth, this issue need to be studied more thoroughly.

Regarding the amount of data exchanged between peers, we consider that in current wireless CNs such as `Guifi.net` using the high quality video streams (i.e., 1080p) affects the service performance since more data needs to pass through the

network to the peers and this may cause congestion in the network. While using standard quality video streams, as shown in our evaluation the amount of loss is lower and more efficiently exchanged between the peers in the network.

Service Discovery: Service discovery operates in parallel at both the global CN cloud level and at the *micro-cloud* level. At the *micro-cloud* level, a number of *Cloudy* instances are federated and share a common, private Layer 2 over Layer 3 network. At that level, Avahi was 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 [Sel+15c]. The multicast-based design does not allow the Avahi service to reach beyond the local link, which is the case in CNs, where services are spread over different nodes that belong to different broadcast domains. However, in the context of an orchestrated *micro-cloud*, Avahi can be used not only for publishing cloud services but also other resources like network folder shares, etc.

Because of the scaling issues we had with Avahi and TincVPN, Serf, a decentralized solution for cluster membership, failure detection, and orchestration was used, 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.

3.7 Summary

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

In this chapter, a deployed CN *micro-cloud* operating in the `Guifi.net` was demonstrated, supporting the feasibility of such a system. In addition, performance measurements were conducted, which demonstrate the usability of cloud-based services for end users.

We identified three services to focus on, considering their importance for the sys-

tem operation and the end users. On the user-level we evaluated distributed storage service Tahoe-LAFS and live-video streaming service PeerStreamer. Tahoe-LAFS performed correctly in uploading and retrieving all the files under the challenging conditions of the CNs and appeared to be a promising application to consider for preserving privacy, and for secure and fault-tolerant storage in CN *micro-clouds*. PeerStreamer, a P2P live streaming service was deployed over geographically distributed real CN nodes. We then studied the effects of different parameters of PeerStreamer on its performance in this environment. We experimented with service discovery based on Avahi and Serf, and the results exhibited their proper functioning. These system-level services allowed users to discover services offered by other CN *micro-cloud* nodes as well as announce their own services.

Performance measurement of services and applications provided by the CN *micro-clouds* were conducted in order to assess their usability by end users. Our results demonstrated the operation of the CN *micro-clouds* in the CNs and the usability of services.

Notes

The work discussed in Chapter 3 was included in the following publications:

[Sel+15a] Mennan Selimi, Amin M Khan, Emmanouil Dimogerontakis, Felix Freitag, and Roger Pueyo Centelles. "Cloud services in the Guifi.net community network". In: *Computer Networks* 93.P2 (Dec. 2015), pp. 373–388.

[Sel+15b] Mennan Selimi, Felix Freitag, Llorenç Cerdà-Alabern, and Luís Veiga. "Performance evaluation of a distributed storage service in community network clouds". In: *Concurrency and Computation: Practice and Experience* 28.11 (2015), pp. 3131–3148.

[Sel+15c] M. Selimi, N. 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 2015)*. Nov. 2015, pp. 202–209.

[Sel+15d] M. Selimi, F. Freitag, R. P. Centelles, A. Moll, and L. Veiga. "TROBADOR: Service Discovery for Distributed Community Network Micro-Clouds". In: *2015 IEEE 29th International Conference on Advanced Information Networking and Applications (AINA 2015)*. Mar. 2015, pp. 642–649.

4

Topology-aware Service Placement

Services running in community network (CN) *micro-clouds* face specific challenges intrinsic to these infrastructures, such as the limited capacity of nodes and links, their dynamics and geographic distribution, wireless links with asymmetric quality for services etc. CN *micro-clouds* are used to deploy distributed services, such as storage and streaming services, which transfer significant amounts of data between the nodes on which they run. If the underlying network resources are not taken into account, these service may suffer from poor performance, e.g, by sending large amounts of data across slow wireless links while faster and more reliable links remain underutilized. Therefore, a key challenge in CN *micro-clouds* is to determine the location of the service deployments, i.e., server locations at a certain geographic points in the network. Due to the dynamic nature of CNs (i.e., topology) and usage patterns, it is challenging to calculate an optimal placement.

In this chapter, first we introduce a service placement algorithm that provides optimal service overlay allocations without the need to verify the whole solution space. The *PASP* (Policy-aware Service Placement) algorithm finds the minimum possible distance in terms of the number of hops between two furthest selected resources and

Table 4.1: Summary of the Guifi.net network graphs (2016)

	nodes/edges	node degree max/mean/min	diameter	zones
BaseGraph	**13636/13940**	537/2.04/1	35	129
CoreGraph	**687/991**	20/2.88/1	32	85

at the same time fulfil different service type quality criteria. Then, we extensively study the effectiveness of our approach in simulations using real-world node and usage traces from $32,000$ `Guifi.net` nodes. From the results obtained in the simulation study, we are able to determine the key features of the network and node selection for different service types. Subsequently, we deploy our algorithm, driven by these findings, in a real production CN and quantify the performance and effects of our algorithm with a distributed storage service. The work in this chapter addresses the research question **Q2**.

4.1 Network Structure

The CNML (Community Network Markup Language) information obtained from the `Guifi.net` has been used to build two topology graphs: *base-graph* and *core-graph*. The *base-graph* of `Guifi.net` is constructed by considering only operational nodes, marked in *Working* status in the CNML file, and having one or more links pointing to another node in the zone. Additionally, we have discarded some disconnected clusters. All links are bidirectional, thus, we use an undirected graph.

We have formed what we call the *core-graph* by removing the client nodes of the *base graph* (i.e., terminal nodes). Table 4.1 summarizes the main properties of *base-grap* and *core-graph* that we use in our study e.g., number of nodes, node degree, diameter (i.e., number of max hops in the sub-graph) and number of zones traversed in these two graphs.

4.2 Allocation Model and Architecture

In order to generalize the placement model for community services, we made the following assumptions that give to our model the flexibility to adapt to many different types of real services. In our case, a *service* is a set of S generic processes or replicas (i.e., with different roles or not) that interact or exchange information through the CN. The service can also be a distributed service built from many service parts. These parts or components of a service would create an overlay and interact with each other to offer more complex services. Each of the service replicas or components will be deployed over a node in the network, where each node will host only one process no matter which service it belongs to.

It is important to remark that the services aimed in this work are at infrastructure level (IaaS), as cloud services in current dedicated data centers. Therefore the services are deployed directly over the core resources of the network (i.e., nodes in the *core-graph*) and accessed by *base-graph* clients. Services can be deployed by `Guifi.net` users or administrators. The architecture that we consider is based on a hybrid peer-to-peer model with three hierarchical levels of responsibility. On each level, members are able to share information among themselves.

The coordination is managed by some peer (i.e., as a super-peer) designated from the immediate upper layer. Three types of peers can be identified:

1. **Community Nodes**: are the computing equipment placed along the wireless CN by users. Besides contributing to the network quality and stability, they share all or part of their physical resources with other community members in an infrastructure as a service (IaaS) fashion. In terms of type and amount of resources, our model assumes the nodes are different. This means that from service point of view there is allocation preference.

2. **Zone Managers**: are single nodes - only one within each zone, selected among all the *Community nodes* with the extra responsibility to manage local zone services and coordinate inter zones aggregated information. In our model we

do not explicitly identify these managers and we assume the existence of at least one of them in each area.

3. **The Controller**: is a unique centralized entity in our system. The role of the controller is to manage all the service allocation requests from the users and update service structures by pulling the configuration information for the zone managers. The allocation algorithms are implemented in the controller.

4.3 Service Quality Parameters

Resource dispersion in a CN scenario can be a drawback or an advantage, as the Nebula [Ryd+14] authors claim in their research. The overlay created by the services abstracts from the actual underlying network connections. Based on that, services that require intensive inter-component communication (e.g., streaming service), can perform better if the replicas (i.e., service components) are placed close to each other in high capacity links [Sel+15b]. On other side, bandwidth-intensive services (e.g., distributed storage, video on-demand) can perform much better if their replicas are as close as possible to their final users (i.e., overall reduction of bandwidth for service provisioning).

Having information about the service SLAs (Service level agreement) in CNs and node behaviour from the underlying network, placement decisions can be made accordingly to promote that certain types of services to be executed in certain type of nodes with better QoS.

Our algorithm considers the following network and graph metrics as shown in Table 4.2, when allocating different type of services.

- **Availability:** The availability of a node is defined as the percentage of ping requests that the node replies when requested by the graph-server system. Graph-servers are distributed in the `Guifi.net` and are responsible for performing network measurements between nodes. This is an important metric for service life-cycle and is considered for two service types. It is measured in percentage (%).

Table 4.2: Service-specific quality parameters

Type of service	Examples of services	Network metrics	Graph metrics
Bandwidth-intensive	distributed storage, video-on-demand	availability	**closeness**
	network graph server, mail server	**closeness**	**centrality**
Latency-sensitive	VoIP, video-streaming	availability	**betweenness**
	game server, radio station server	**latency**	**centrality**

- **Latency:** The latency of a node is defined as the time it takes for a small IP packet to travel from the `Guifi.net` graph-servers through the network to the nodes and back. It is an important metric for latency-sensitive service in CN *micro-clouds*. It is measured in milliseconds (ms).

- **Closeness:** The closeness is defined as the average distance (i.e., number of hops) from the solution obtained by the algorithm to the clients. It is an important metric for bandwidth-sensitive services. It is measured in number of *hops*.

In terms of graph centrality metrics, we consider closeness and betweenness centrality. Closeness centrality is a good measure of how efficient a particular node is in propagating information through the network. Betweenness centrality quantifies the number of times a node acts as a bridge along the shortest path between two other nodes.

4.4 PASP: Policy-Aware Service Placement

We designed an algorithm that explores different placements searching for the local minimal service overlay diameter while, at the same time, fulfilling different service type quality parameters. The diameter is defined as the number of maximum hops between two selected resources in the sub-graph.

Algorithm 4.1 Policy-Aware Service Placement (PASP)

Input: $N(V_n, E_n)$ ▷ Network graph
Input: $Z(V_z, E_z)$ ▷ Zones graph
Input: $Zone$ ▷ Search solution zone
Input: S ▷ Number of nodes in the service
Input: $ServicePolicy$ ▷ Service specific policies

```
1: procedure PASP(N, Z, Zone, S, ServicePolicy)
2:     Community ← V_n ∈ V_{z_i}
3:     BestSolution ← null
4:     for all node ∈ Community do
5:         solution ←BreadthFirstSearch(N, Community, node, S)
6:         if isBetter(solution, BestSolution, ServicePolicy) then
7:             BestSolution ← solution
8:         end if
9:     end for
10:    return BestSolution
11: end procedure
12: procedure isBetter(currentSolution, bestSolution, ServicePolicy)
13:    for all p ∈ ServicePolicy do
14:        result ←CheckPolicy(currentSolution, bestSolution, p)
15:    end for
16:    return result
17: end procedure
```

The Algorithm 4.1 relies on the method *PASP()* to evaluate the different service placements in different zones and generate the solutions. The algorithm tries to find a solution in each zone by applying Breadth-First Search (BFS) and utilizing the *IsBetter* method to choose the best solutions by applying service policies shown at Table 4.2. In the case of equal diameter allocations, the mean out-degree (i.e., the mean boundary of the nodes in the service overlay with the nodes outside of it) is taken. The service allocation with smallest diameter and largest mean out-degree fulfilling different service quality parameters is kept as the optimal.

The algorithm iterates using Breadth-First-Search algorithm (BFS) in the network

graph, taking as *root* the given node and selecting the first *S-1* closest resources to it. Initially, the node with high degree centrality is chosen as a root. Degree centrality is the fraction of nodes that a particular node is connected to. In the case of several nodes at the same distance, nodes are selected randomly, distributing thus uniformly. Thanks to this feature, our algorithm performs faster than a pure exhaustive search procedure, since size equivalent placements are not evaluated. It is worth noting that the same set of nodes might be obtained from different root nodes, since placements in nearby network areas would involve the exact same nodes. We avoid re-evaluating these placements with a cache mechanism, that improves algorithm efficiency. After the placement solutions for different number of services are returned from BFS, the solutions are compared regarding the service quality parameters.

For each solution set obtained, we check our defined service-specific policies and then accordingly we calculate different scores using the *CheckPolicy* method (e.g., latency, availability or closeness score). Once we have the these scores for each solution set, we utilize the *IsBetter* method to compare the solutions and to choose the new best placement solution according to different service types. Currently, the algorithm has not been optimized regarding the computation time, but it provides near-optimal overlay allocations, as our results show, without need of verifying the whole solution space.

4.5 Experimental Results

Our service placement algorithm proposed in Section 4.4 is used to simulate the placement of different services in `Guifi.net`. Our goal is to determine the key features of the network and its nodes, in particular to understand the network metrics that could help us to design new heuristic frameworks for smart service placement in CN *micro-clouds*. As a first step, it is vital to understand the behaviour of two important metrics (e.g., node availability and latency). We achieve this, by characterizing the nodes and network performance of the `Guifi.net` CN.

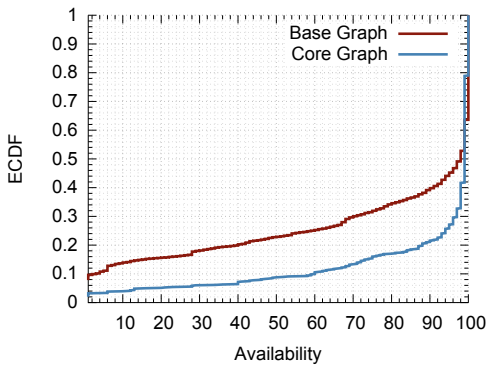


Figure 4.1: Guifi.net node availability

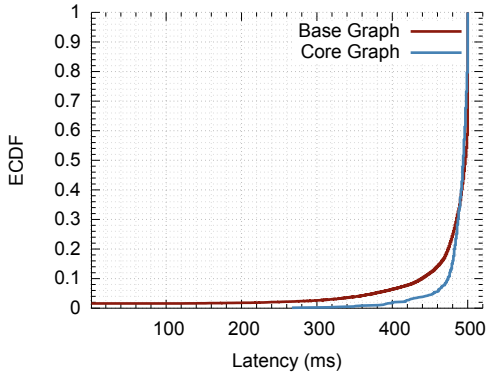


Figure 4.2: Guifi.net node latency

4.5.1 Network Behaviour and Algorithmic Performance

From the data obtained, our first interest is to analyse the availability and latency of `Guifi.net` nodes. This can be used as an indirect metric of quality of connectivity that new members may expect from the network.

Figure 4.1 shows that 40% of the *base-graph* nodes are reachable from the network 90% or less of the time. The situation seems to be better with the *core-graph* nodes, which are the most stable part of the network (i.e., 20% of the *core-graph* nodes have an availability of 90% or less). *Core-graph* nodes have higher availability because they comprise the backbone of the network and connect different administrative zones. The successful operation of the *base-graph* nodes depends on the *core-graph* nodes. In this work, the services are placed on the *core-graph* nodes. Therefore, selecting the *core-graph* nodes with higher availability is of high importance (i.e., avoiding *core-graph* nodes with 90% or less availability).

Figure 4.2 depicts the Empirical Cumulative Distribution Function (ECDF) plot of the node latency. Similar to the availability case, the latency of the *core-graph* nodes is slightly better. For both cases, 30% of the nodes have latency of 480 ms or less, which makes the other 70% of the nodes to have higher latency. The availability and latency graph demonstrate the importance of, and indeed the need for, a more effective, network-aware placement in CN *micro-clouds*. By not taking the performance of

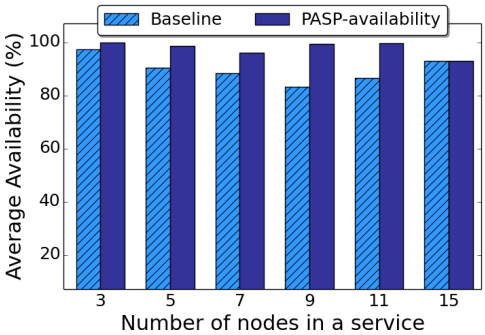


Figure 4.3: PASP-Availability

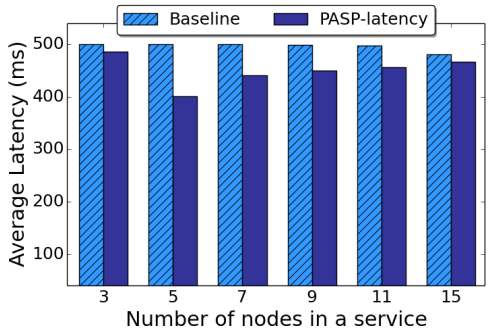


Figure 4.4: PASP-Latency

the underlying network into account, applications can end up sending large amounts of data across slow wireless links while faster and more reliable links and nodes remain under-utilized.

In order to see the effects of the network-aware placement in the solutions obtained, we compare two versions of our algorithm. The first version i.e., *Baseline*, allocates services just with the goal of minimizing the service overlay diameter without considering node properties such as availability, latency or closeness. The second version of the algorithm called *PASP*, tries to minimize the service overlay diameter, *while* taking into account these node properties.

The availability and latency of the *Baseline* solutions are calculated by taking the average of nodes in the optimal solutions (i.e., after the optimal solution is computed), where the optimal solution is the best solution that minimizes the service overlay diameter, that can only be calculated exhaustively offline.

We allocate services of size 3, 5, 7, 9, 11 and 15. The service size can be chosen arbitrarily. Figure 4.3 and Figure 4.4 reveal that nodes obtained in the solutions with *PASP* have higher average availability and lower latency than with *Baseline*, with minimum service overlay diameter. On average, the gain of *PASP* over *Baseline* is 8% for the availability, and 45 ms for the latency (i.e., $5 - 20\%$ reduction).

We find that our *PASP* algorithm is good in finding placement solutions with higher availability and lower latency, however the service solutions obtained might

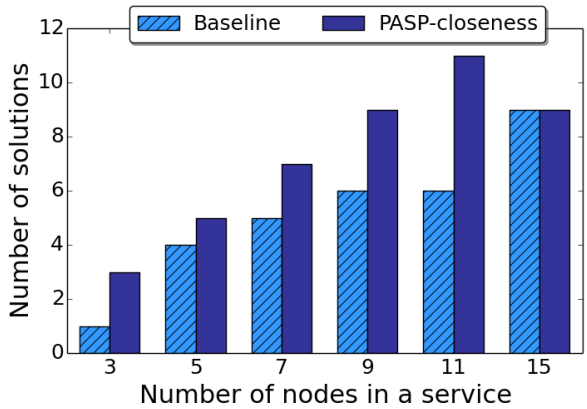


Figure 4.5: PASP-Closeness

not be very close in terms of number of hops to the *base-graph* clients. Because of this we also developed another flavour of *PASP* algorithm called *PASP-closeness*. Figure 4.5 shows the number of solutions obtained that are 1-hop close to the base-graph clients. When *PASP-closeness* algorithm allocates three services, on average there are three solutions whose internal nodes (e.g, any of the nodes) are at 1-hop distance to any of the *base-graph* client nodes, contrary to the *Baseline* where on average there is one solution whose nodes are at 1-hop distance to *base-graph* clients.

Overall, in the two algorithms, there is a trade-off between latency and closeness. For bandwidth-intensive applications closeness seems to be more important when allocating services (e.g., *PASP-closeness* can be used), while for latency-sensitive applications it is the latency the one that naturally seems to be more important (e.g., *PASP-latency* can be used).

4.5.2 Deployment in a Real Production Community Network

In order to understand the gains of our network-aware service placement algorithm in a real production CN, we deploy our algorithm in real hardware connected to the nodes of the *qMp* network (i.e., a subset of the `Guifi.net` CN). The *qMp* network is explained in Section 2.1.2.

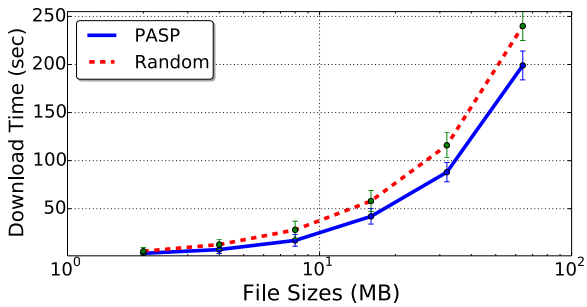


Figure 4.6: Average client reading times

We use 16 nodes connected to the wireless nodes of *qMp* network. The nodes and the attached servers are geographically distributed in the city of Barcelona. The hardware of the servers consists of Jetway devices, which are equipped with an Intel Atom N2600 CPU, 4 GB of RAM and 120 GB SSD. They run an operating system based on OpenWRT, which allows running several slivers (VMs) on one node simultaneously implemented as Linux containers (LXC).

The slivers host the *Cloudy* operating system. Cloudy contains some pre-integrated distributed applications, which the CN user can activate to enable services inside the network. For our experiments, we use the storage service based on Tahoe-LAFS.

As the controller node we leverage the experimental infrastructure of *Community-Lab*. Community-Lab provides a central coordination entity that has knowledge about the network topology in real time. Out of the 16 devices used, 3 of them are storage nodes and 13 of them are clients (i.e., chosen randomly) that read files. The clients are located in different geographic locations of the network. The controller is the one that allocates the distributed storage service in these 3 nodes and clients access this service. On the client side we measure the file reading times. We monitored the network for the entire month of January 2016. The average throughput distribution of all the links for one month period was 9.4 Mbps.

Figure 4.6 shows the average download time for various file sizes ($2 - 64$ MB) perceived at the 13 clients, after allocating services using *Random* algorithm (i.e., currently used at `Guifi.net`) and using our *PASP* algorithm. Allocation of services

using *Random* algorithm by *Controller* is done without taking into account the performance of the underlying network. It can be seen for instance that when using our *PASP* algorithm for allocation, it takes around 17 seconds for the clients on average to read a 8 MB file. In the *Random* case, the time is almost doubled, reaching 28 seconds for reading a file from the clients side. We observed therefore that when allocating services, taking into account the closeness and availability parameters in the allocation decision, on average (i.e., for all clients) our algorithm reduces the client reading times for 16%. Maximum improvement (i.e., around 31%) has been achieved when reading larger files (e.g., 64 MB). When reading larger files client needs to contact many nodes in order to complete the reading of the file.

4.6 Discussion

From working with the `Guifi.net` data, we found that augmenting the service overlay diameter with service specific metrics has a direct impact on the service performance. Our first interest was to ascertain the minimal diameter that could be obtained when allocating services of different sizes. We found that the optimal flooding radius [Wan+15b] in `Guifi.net` is small, i.e., 2 hops. This means that it is always possible to find an optimal service placement solution with an overlay diameter of 2 hops within `Guifi.net` network. Additionally, the average solutions diameter increases as the number of nodes that composes the services does.

Furthermore, we observed some patterns in the node features that conforms optimal placements. We saw that the solution overlay diameter depends on the nodes degree centrality. Minimum degree centrality can be used to select the first node that composes the service (i.e., the solution). We saw that most of the solutions obtained are concentrated on a small set of of average centrality values. Selecting the next nodes in a particular range of closeness centrality (i.e., for bandwidth-intensive services) and betweenness centrality (i.e., for latency-sensitive services) is specially useful to obtain more optimal overlays.

4.7 Summary

In this chapter, we addressed the need for the network-aware service placement in CN *micro-cloud* infrastructures. We looked at a specific case of improving the deployment of service instances in CN *micro-clouds* for enabling an improved distributed storage service in a CN. As services become more network-intensive, the bandwidth and latency between the used nodes becomes the bottleneck for improving service performance. Network awareness in placing services allows to chose more reliable and faster paths over poorer ones.

We introduced a service placement algorithm *PASP* that provides improved service overlay allocations for distributed services considering service quality parameters. For our simulations we employed a topological snapshot from `Guifi.net` to identify node traits in the optimal service placements. We deployed our service placement algorithm in a real network segment of `Guifi.net` and quantified the performance and effects of our algorithm. We conducted our study on the case of a distributed storage service. We found that our *PASP* algorithm reduces the client reading times by an average of 16% (i.e., with a max. improvement of 31%) compared to the currently used organic placement scheme. Our results show how the choice of an appropriate set of nodes, taken from a larger resource pool, can influence service performance significantly.

Notes

The work discussed in Chapter 4 was included in the following publication:

[Sel+16] Mennan Selimi, Davide Vega, Felix Freitag, and Luís Veiga. "Towards Network-Aware Service Placement in Community Network Micro-Clouds". In: *22nd International Conference on Parallel and Distributed Computing (Euro-Par 2016)*. Grenoble, France, 2016, pp. 376–388.

5

Service Placement Heuristic

By putting services closer to users, community network (CN) *micro-clouds* pursue not only a better service performance, but also a low entry barrier for the deployment of mainstream Internet services within the CN. Unfortunately, the provisioning of the services is not so simple. Due to the large and irregular topology, high software and hardware diversity of CNs, it requires of a ”careful” placement of CN *micro-cloud* services over the network. Obviously, a placement algorithm that is agnostic to the state of the underlying network may lead to important inefficiencies. Although conceptually straightforward, it is challenging to calculate an optimal decision due to the dynamic nature of CNs and usage patterns. This chapter tries to answer the following two questions: first, given that sufficient state information is in place, is network-aware placement enough to deliver satisfactory performance to CN users? Second, can the redundant placement of services further improve performance?

To achieve this, in this chapter we propose to leverage state information about the network to inform service placement decisions, and to do so through a fast heuristic algorithm, which is vital to quickly react to changing conditions. We present a new placement heuristic called *BASP* (Bandwidth and Availability-aware Service Place-

ment), which uses the state of the underlying CN to optimize service deployment. In particular, it considers two sources of information: i) network bandwidth and ii) node availability to make optimized decisions. Compared with brute-force search, which it takes of the order of hours to complete, *BASP* runs much faster; it just takes a few seconds, while achieving reasonably good results. Driven by these findings, we then ran *BASP* in a real CN and quantified the boost in performance achieved after deploying a live video-streaming and Web 2.0 service. The work in this chapter addresses the research question **Q3**.

5.1 Network Characterization

Our service placement strategy considers two aspects: node availability and network bandwidth. As the first step it is vital to understand the behaviour of these two dimensions in a real CN. We achieve this by characterizing a production wireless CN such as a *qMp* network over a five-month period. A detailed description of the *qMp* network is given in Section 2.1.2.

Our goal is to determine the key features of the network (i.e., bandwidth distribution) and its nodes (i.e., availability patterns) that could help us to design new heuristics for intelligent service placement in CNs.

5.1.1 Node Availability

The quality and state of the heterogeneous hardware used in *qMp*, influences the stability of the links and network performance. Availability of the *qMp* nodes is used as an indirect metric for the quality of connectivity that new members expect from the network.

Figure 5.1 shows the Empirical Cumulative Distribution Function (ECDF) of the *qMp* node availability collected for a period of five months. We define the availability of a node as the percentage of times that the node appears in a *capture*, counted since the node shows up for the first time. A capture is an hourly network snapshot that we take from the *qMp* network (i.e., we took 2718 captures in total). Figure 5.1 reveals

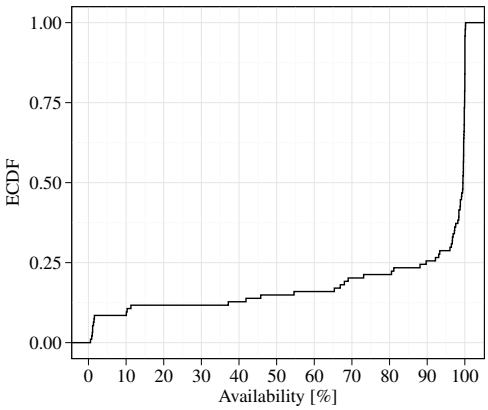


Figure 5.1: qMp node availability

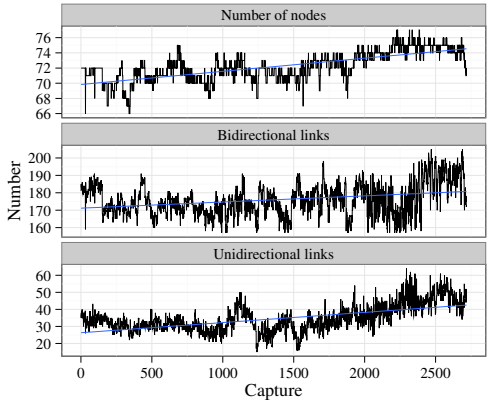


Figure 5.2: Number of nodes and links

that 25% of the nodes have an availability lower than 90% and others nodes left have an availability between $90 - 100\%$. In a CN such as *qMp*, users do not tend to deliberately reboot the device unless they have to perform an upgrade, which is not very common. Hence, the percentage of times that node appears in a capture is a relatively good measure of the node availability due to random failures.

When we compare the availability distribution reported in a similar study and environment on PlanetLab [VP11], a *qMp* node has a higher probability of being disconnected or not reachable from the network. The fact that PlanetLab showed a higher average availability (i.e., sysUpTime) on its nodes may be because it is an experimental testbed running on much more stable computers and environment. Furthermore, the *qMp* members are not only responsible for the maintenance of their nodes, but also for ensuring a minimum standard of connectivity with other parts of the network.

Figure 5.2 depicts the number of nodes and links during captures. Figure shows that *qMp* is growing. Overall, 77 different nodes were detected. From those, 71 were alive during the entire measurement period. Around 6 nodes were missed in the majority of the captures. These are temporarily working nodes from other mesh networks and laboratory devices used for various experiments. Figure 5.2 also reveals that on average 175 of the links used between nodes are bidirectional and 34 are unidirectional. For bidirectional links, we count both links in opposite direction as a

single link.

In summary, node availability is important to identify those nodes that will minimize service interruptions over time. Based on the measurements, we assign availability scores to each of the nodes. The highly available nodes are the possible candidates for deploying on them the CN *micro-cloud* services.

5.1.2 Bandwidth Characterization

A significant amount of services that run on *qMp* and `Guifi.net` network are network-intensive (i.e., bandwidth and delay sensitive), transferring large amounts of data between the network nodes [Sel+15a]. The performance of such kind of services depends not just on computational and disk resources but also on the network bandwidth between the nodes on which they are deployed. Therefore, considering the network bandwidth when placing services in the network is of high importance.

First, we characterize the wireless links of the *qMp* network by studying their bandwidth. Figure 5.3 shows the average bandwidth distribution of all the links. The figure shows that the link bandwidth can be fitted with a mean of 21.8 Mbps. At the same time Figure 5.3 reveals that around 60% of the nodes have 10 Mbps or less bandwidth. The average bandwidth of 21.8 Mbps obtained in the network allows many popular bandwidth-hungry service to run without big interruptions. This high performance can be attributed to the 802.11an devices used in the network.

In order to see the variability of the bandwidth, Figure 5.4 shows the bandwidth averages in both directions of three busiest links. Upload operation is depicted with a solid line and download operation with a dashed line. The nodes of three busiest links are highlighted on the top of the figure. We noted that the asymmetry of the bandwidths measured in both directions it not always due to the asymmetry of the user traffic (i.e., not shown in the graphs). For instance, node GSgranVia255, around 6 am, when the user traffic is the lowest and equal in both directions, the asymmetry of the links bandwidth observed in Figure 5.4 remains the same. We thus conclude that even though bandwidth time to time is slightly affected by the traffic, the asymmetry of the links that we see might be due to the link characteristics, as level of interferences

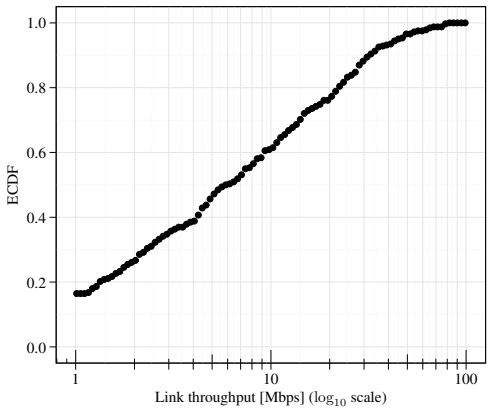


Figure 5.3: Bandwidth distribution

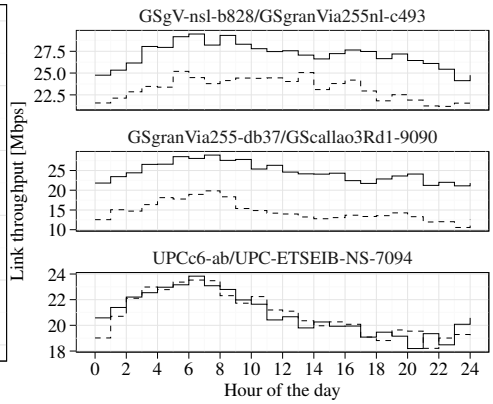


Figure 5.4: Bandwidth in three busiest links

present at each end, or different transmission powers.

In order to measure the link asymmetry, Figure 5.5 depicts the bandwidth measured in each direction. A boxplot of the absolute value of the deviation over the mean is also depicted on the right. The figure shows that around 25% of the links have a deviation higher than 40%. At the same time, the other 25% of the links have a deviation less than 10%. After performing some measurements regarding the signaling power of the devices, we discovered that some of the community members have re-tuned the radios of their devices (e.g., transmission power, channel and other parameters), trying to achieve better performance, thus, changing the characteristics of the links. Thus, we can conclude that the symmetry of the links, an assumption often used in the literature of in wireless mesh networks, is not very realistic for our case and service placement algorithms unquestionably need to take into account.

5.1.3 Observations

Here are some observations (i.e., features) that we have derived from the measurements in *qMp* network:

Dynamic Topology: *qMp* network is highly dynamic and diverse due to many reasons, e.g., its community nature in an urban area; its decentralised organic growth

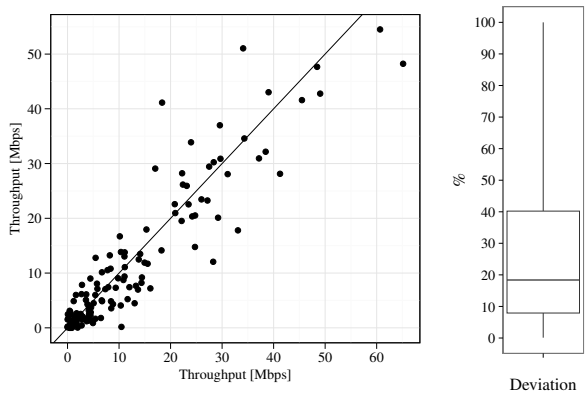


Figure 5.5: Bandwidth asymmetry

with extensive diversity in the technological choices for hardware, wireless media, link protocols, channels, routing protocols etc.; its mesh nature in the network etc. The current network deployment model is based on geographic singularities rather than QoS. The network is not scale-free. The topology is organic and different w.r.t. conventional ISP network.

Non-uniformly distributed resources: The resources are not uniformly distributed in the network. Wireless links are with asymmetric quality for services (i.e., 25% of the links have a deviation higher than 40%). We observed a highly skewed traffic pattern and highly skewed bandwidth distribution (i.e., shown in the Figure 5.3).

Currently used organic (i.e., random) placement scheme in *qMp* and `Guifi.net` in general, is not sufficient to capture the dynamics of the network and therefore it fails to deliver the satisfying QoS. The strong assumption under random service placement, i.e., uniform distribution of resources, does not hold in such environments.

Furthermore, the services deployed have different QoS requirements. Services that require intensive inter-component communication (e.g., streaming service), can perform better if the replicas (i.e., service components) are placed close to each other in high capacity links [Sel+15b]. On other side, bandwidth-intensive services (e.g., distributed storage, video-on-demand service) can perform much better if their replicas are as close as possible to their final users (i.e., overall reduction of bandwidth for

service provisioning) [Sel+16a].

Our goal is to build on this insight and design a network-aware service placement algorithm that will improve the service quality and network performance by optimizing the usage of scarce resources in CNs such as bandwidth.

5.2 Context and Problem

First we describe our model for network and service graph. Subsequently we build on this to describe the service placement problem. The symbols used are listed in Table 5.1.

5.2.1 Network Graph in qMp

We call the CN the *underlay* to distinguish it from the *overlay* network which is built by the services. The underlay network is supposed to be connected and we assume each node knows whether other nodes can be reached (i.e., next hop is known). We can model the underlay graph as: $G \leftarrow (N, E)$ where N is the set of nodes connected to the outdoor routers (ORs) present in the CNs and E is the set of wireless links that connects them. Physical links between nodes are characterized by a given bandwidth (B_i). Furthermore, each link has a bandwidth capacity (B_e). Each node in the network has an availability score (R_n) derived from the real measurements in the *qMp* network.

5.2.2 Service Graph in qMp

The services aimed in this work are at infrastructure level (IaaS), as cloud services in current dedicated data centers. Therefore, the services are deployed directly over the core resources of the network and accessed by clients. Services can be deployed by *qMp* users or administrators.

The services we consider in this work are distributed services (i.e., independently deployable services as in the Microservices Architecture[*]). The distributed services

[*] http://microservices.io/patterns/microservices.html

Table 5.1: Input and decision variables

Symbol	Description
N	set of physical nodes in the network
E	set of edges (physical links) in the network
S	set of services
D	set of service copies
k	max number of service copies
B_e	bandwidth capacity of link e
$\beta_{s1,s2}$	bandwidth requirement between services s1 and s2
R_n	Availability of node n
λ	Availability threshold
$X_{s1,s2}$	use of physical link e by at least one service for the placement of virtual link between s1 and s2, 1 iff placed

can be monolithic and composite services (i.e., non-monolithic) built from simpler parts, e.g., video streaming (i.e.„ built from the source and peers component), web service (i.e., built from database, memcached and client component) etc. In the real deployment, one service component corresponds to one Docker container. These parts or components of the services create an overlay and interact with each other to offer more complex services. Bandwidth requirement between two services s_1 and s_2 is given by $\beta_{s1,s2}$. At most k copies can be placed for each service s. The decision variable $X_{s1,s2}$ tells how the virtual links (i.e., service overlay links) are routed over the physical links.

A service may or may not be tied to a specific node of the network. Each node can host one or more type of services. In this work we assume an *offline* service placement approach where a single or a set of applications are placed "in one shot" onto the underlying physical network. We might rearrange (i.e., migrate) the placement of the same service over the time because of the service performance fluctuation (e.g., weather conditions, node availability, changes in use pattern, and etc.). We do not consider real-time service migration.

5.2.3 Service Placement Problem

The concept of service and network graph allows us to formulate the problem statement more precisely as: *"Given a service and network graph, where to place the service in the network as to maximize user's QoS and QoE, while satisfying a required level of availability for each node (N) and considering a maximum of k service copies ?*

Let B_{ij} be the bandwidth of the path to go from node i to node j. We want a partition of k clusters (i.e., services) : $C \leftarrow C_1, C_2, C_3, ..., C_k$ of the set of nodes in the mesh network. The cluster head i of cluster C_i is the location of the node where the service will be deployed. The partition maximizing the bandwidth from the cluster head to the other nodes in the cluster is given by the objective function:

$$\arg\max_C \sum_{i=1}^{k} \sum_{j \in Ci} B_{ij} \tag{5.1}$$

with respect to the following constraints:

1. The total bandwidth used per link cannot exceed the total link capacity:

$$\forall e \in \mathbb{E} : \sum_{s1,s_2 \in \mathbb{S}} X_{s1,s2}(e) \times \beta_{s1,s2} \leq B_e \tag{5.2}$$

2. Availability-awareness: the node availability should be higher than the predefined threshold λ:

$$\forall n \in \mathbb{N} : \sum_{n \in \mathbb{N}} R_n \geq \lambda \tag{5.3}$$

3. Admission control: At most, k copies can be placed for each service:

$$|D| \leq k \tag{5.4}$$

5.2.4 BASP: Bandwidth and Availability-aware Service Placement

Solving the problem stated in Equation 5.1 in brute force for any number of N and k is NP-hard and very costly. The naive brute force method can be estimated by calculating the *Stirling number of the second kind* [16r] which counts the number of ways to partition a set of n elements into k nonempty subsets, i.e., $\frac{1}{k!}\sum_{j=0}^{k}(-1)^{j-k}\binom{n}{k}j^n \Rightarrow \mathcal{O}(n^k k^n)$. Thus, due to the obvious combinatorial explosion, we propose a low-cost and fast heuristic called *BASP*. The *BASP* (Bandwidth and Availability-aware Service Placement) allocates services taking into account the bandwidth of the network and the node availability.

Our BASP algorithm (i.e., Algorithm 5.1) runs in three phases:

1. **Phase 1: K-Means:** Initially, we use the naive K-Means partitioning algorithm in order to group nodes based on their geo-location. The idea is to get back clusters of nodes that are close to each other. The K-Means algorithm forms clusters of nodes based on the Euclidean distances between them, where the distance metrics in our case are the geographical coordinates of the nodes. In traditional K-Means algorithm, first, k out of n nodes are randomly selected as the cluster heads (i.e., centroids). Each of the remaining nodes decides its cluster head nearest to it according to the Euclidean distance. After each of the nodes in the network is assigned to one of k clusters, the centroid of each cluster is re-calculated. Each cluster contains a full replica of a service, i.e., the algorithm in this phase partitions the network topology into k (i.e., maximum allowed number of service replicas) clusters. Grouping nodes based on geo-location is in line with how the *qMp* is organized. The nodes in *qMp* are organized into a tree hierarchy of *zones*. A zone can represent nodes from a neighborhood or a city. Each zone can be further divided in child zones that cover smaller geographical areas where nodes are close to each other. From the service perspective we consider placements inside a particular zone. We use K-Means with geo-coordinates as an initial heuristic for our algorithm. As an alternative, clustering based on network locality can be used. Several graph

community detection techniques are available for our environment [LF09].

2. **Phase 2: Aggregate Bandwidth Maximization:** The second phase of the algorithm is based on the concept of finding the cluster heads maximizing the bandwidth between them and their member nodes in the clusters C_k formed in the first phase. The bandwidth between two nodes is estimated as the bandwidth of the link having the minimum bandwidth in the shortest path. The cluster heads computed are the candidate nodes for the service placement. This is plotted as Naive K-Means in the Figure 5.6.

3. **Phase 3: Cluster Re-Computation:** The third and last phase of the algorithm includes reassigning the nodes to the selected cluster heads having the maximum bandwidth, since the geo-location of nodes in the clusters formed during phase one is not always correlated with their bandwidth. This way the clusters are formed based on nodes bandwidth. This is plotted as *BASP* in the Figure 5.6.

Complexity:

The complexity of the *BASP* is as follows: for *BASP*, finding the optimal solution to the K-Means clustering problem (i.e., phase one) if k and d (i.e., the dimension) are fixed (e.g., in our case $n = 71$, and $d = 2$), the problem can be exactly solved in time $\mathcal{O}(n^{dk+1} \log n)$, where n is the number of entities to be clustered. The complexity for computing the cluster heads in phase two is $\mathcal{O}(n^2)$, and $\mathcal{O}(n)$ for the reassigning the clusters in phase three. Therefore, the overall complexity of *BASP* is polylogarithmic $\mathcal{O}(n^{2k+1} \log n)$, which is significantly smaller than the brute force method and thus practical for commodity processors.

5.3 Experiment Setup

We take a network snapshot (i.e., capture) from 71 physical nodes of the *qMp* network regarding the bandwidth of the links[†] and node availability. The node and bandwidth

[†]http://tomir.ac.upc.edu/qmpsu/index.php?cap=56d07684

Algorithm 5.1 B A S P

Input: $G(N, E)$ ▷ Network graph
$C \leftarrow C_1, C_2, C_3, ..., C_k$ ▷ k partition of clusters
B_i ▷ bandwidth of node i
R_n, λ ▷ availability of node n, λ availability threshold

1: **procedure** PERFORMKMEANS(G, k)
2: **if** $R_n \geq \lambda$ **then**
3: **return** C
4: **end if**
5: **end procedure**
6: **procedure** FINDCLUSTERHEADS(C)
7: $clusterHeads \leftarrow list()$
8: **for all** $k \in C$ **do**
9: **for all** $i \in C_k$ **do**
10: $B_i \leftarrow 0$
11: **for all** $j \in setdiff(C, i)$ **do**
12: $B_i \leftarrow B_i + estimate.route.bandw(G, i, j)$
13: **end for**
14: $clusterHeads \leftarrow \max B_i$
15: **end for**
16: **end for**
17: **return** $clusterHeads$
18: **end procedure**
19: **procedure** RECOMPUTECLUSTERS($clusterHeads, G$)
20: $C' \leftarrow list()$
21: **for all** $i \in clusterHeads$ **do**
22: $cluster_i \leftarrow list()$
23: **for all** $j \in setdiff(G, i)$ **do**
24: $B_j \leftarrow estimate.route.bandw(G, j, i)$
25: **if** B_j is best from other nodes i **then**
26: $cluster_i \leftarrow j$
27: **end if**
28: $C' \leftarrow cluster_i$
29: **end for**
30: **end for**
31: **return** C'
32: **end procedure**

data obtained has been used to build the topology graph of the *qMp*. The *qMp* topology graph is constructed by considering only operational nodes, marked in "working" status, and having one or more links pointing to another node. The nodes of the *qMp* consists of Intel Atom N2600 CPU, 4 GB of RAM and 120 GB of disk space.

Our experiment is comprised of 5 runs and the presented results are averaged over all the runs. Each run consists of 15 repetitions.

5.4 Comparison

To emphasise the importance of the different phases of Algorithm 5.1, we compare in this section the two phases of our heuristic with *Random* placement, i.e., the default placement at *qMp*.

Random Placement: Currently, the service deployment much as network deployment at *qMp*, is not centrally planned but initiated individually by the CN members. Public, user and community-oriented services are placed randomly on super-nodes and users' premises, respectively. The only parameter taken into account when placing services is that the devices must be in "production" state. The network is not taken into consideration at all. All nodes in the production state appear equally to the users.

Naive K-Means Placement: This corresponds to the second phase of the Algorithm 5.1. The service is placed on the node having the maximum bandwidth on the initial clusters formed by the K-Means. We limit the choice of the cluster heads to be inside the sets of clusters obtained using K-Means.

BASP Placement: It includes the three phases of the Algorithm 5.1. The service is placed on the node having the maximum bandwidth after the clusters are recomputed.

5.5 Results

Figure 5.6 depicts the average bandwidth to the cluster heads obtained with the *Random*, *Naive K-Means* and the *BASP* algorithm. This figure reveals that for any number

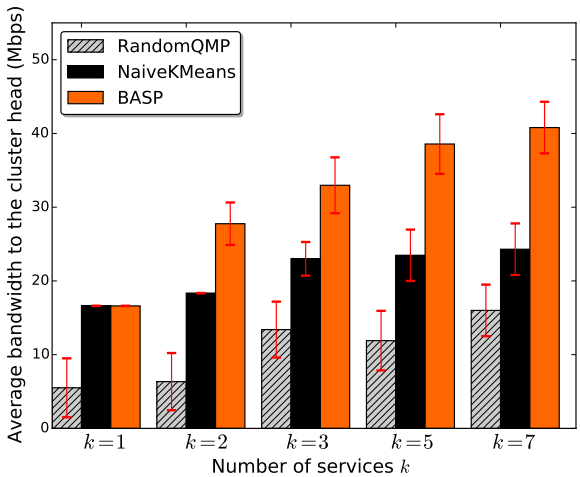


Figure 5.6: Average bandwidth to the cluster heads

of services k, *BASP* outperforms both *Naive K-Means* and *Random* placement.

For $k = 2$, the average bandwidth to the cluster heads is increased from 18.3 Mbps (i.e., *Naive K-Means*) to 27.7 Mbps (i.e., *BASP*), which represents a 50% improvement. The largest increase of 67% is achieved when $k = 7$. On average, when having up to 7 services in the network, the gain of *BASP* over *Naive K-Means* is of 45%. Based on the observations from Figure 5.6, the gap between the two algorithms grows as k increases. We observe that k will increase as the network grows. And hence, *BASP* will presumably render better results for larger networks than the rest of strategies.

Regarding the comparison between *BASP* and *Random* placement, we find that *Random* placement leads to an inefficient use of network's resources, and consequently to suboptimal performance. As depicted in the Figure 5.6, the average gain of *BASP* over naive *Random* placement is 211%.

Comparison to the optimal solution. Note that our heuristic enables us to select cluster heads that provide much higher bandwidth than any other random or naive approach. But, if we were about to look for the optimum bandwidth within the clusters (i.e., optimum average bandwidth for the cluster), then this problem would be NP-hard. The reason is that finding the optimal solution entails running our algorithm

Table 5.2: Centrality measures for cluster heads

	k = 1	k = 2		k = 3			k = 5				
Clusters [node id]	C1 [27]	C1 [20]	C2 [39]	C1 [20]	C2 [39]	C3 [49]	C1 [20]	C2 [4]	C3 [49]	C4 [51]	C5 [39]
Head degree	20	6	6	6	6	10	6	10	10	12	6
Neighborhood Connectivity	7.7	9.6	9.6	9.6	9.6	10.8	9.6	8.7	10.8	8.1	9.6
Diameter	6	5	3	4	3	5	4	2	3	1	3
RandomqMp - Bandwidth [Mbps]	5.3	6.34		13.4			11.9				
Naive K-Means Bandwidth [Mbps]	16.6	18.3		23			23.4				
BASP - Bandwidth [Mbps]	16.9	27.7		32.9			38.5				
BASP - Running Time [sec]	7	15		23			30				

for all the combinations of size k from a set of size n. This is a combinatorial problem that becomes intractable even for small sizes of k or n (e.g., $k = 5$, $n = 71$). For instance, if we wanted to find the optimum bandwidth for a cluster of size $k = 3$, then the algorithm would need to run for every possible (i.e., non-repeating) combination of size 3 from a set of 71 elements, i.e., $choose(71, 3) = 57K$ combinations. We managed to do so and found that the optimum average was 62.7 Mbps. For $k = 2$, the optimum was 49.1 Mbps. For $k = 1$, it was 16.9 Mbps.

The downside was that, the computation of the optimal solution took very long time in a commodity machine. Concretely, it took 5 hours for $k = 3$ and 30 minutes for $k = 2$. Instead, *BASP* spent only 23 seconds for $k = 3$ and 15 seconds for $k = 2$. Table 5.2 shows the improvement of *BASP* over *Random* and *Naive K-Means*. To summarize, BASP is able to achieve good bandwidth performance with very low computation complexity.

Correlation with centrality metrics. Table 5.2 shows some centrality measures and some graph properties obtained for each cluster head. Further, Figure 5.7 shows the neighborhood connectivity graph of the *qMp* network. The neighborhood connectivity of a node v is defined as the average connectivity of all neighbors of v. In the figure, nodes with low neighborhood connectivity values are depicted with bright colors and high values with dark colors. It is interesting to note that some the nodes with the highest neighborhood connectivity are those chosen by *BASP* as cluster heads. The cluster heads (i.e., $k = 2$ and $k = 3$) are illustrated with a rectangle in the graph. A deeper investigation into the relationship between service placement and network topological properties is out of the scope of this thesis work and will be reserved as

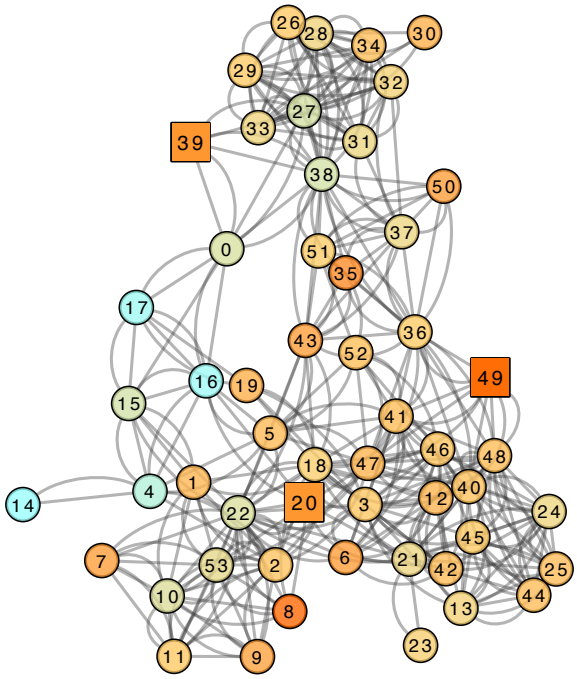


Figure 5.7: Neighborhood connectivity graph of the qMp network

our future work.

5.6 Evaluation in a Real Production Community Network

In order to foster the adoption and transition of the CN*micro-cloud* environment, we use the *Cloudy* distribution. *Cloudy* includes a tool for users to announce and discover services in the *micro-clouds* based on Serf, which is a decentralized solution for cluster membership and orchestration. On the network coordination layer, having sufficient knowledge about the underlying network topology, *BASP* decides about the placement of the service which then is announced via Serf. Thus, the service can be discovered by the other users.

In order to understand the gains of our network-aware service placement algorithm in a real production CN, we deploy our algorithm in real hardware connec-

ted to the nodes of the *qMp* network, located in the city of Barcelona. We concentrate on benchmarking two of the most popular network-intensive applications: *Video streaming service*, and *Web 2.0 Service* performed by the most popular websites.

5.6.1 Live-video Streaming Service

PeerStreamer[‡], an open source live P2P video streaming service, has been paradigmatically established as the live streaming service in *Cloudy*. This service is based on *chunk diffusion*, where peers offer a selection of the chunks they own to some peers in their neighborhood. A chunk consists of a part of the video to be streamed (i.e., one frame of the video). PeerStreamer differentiates between a source node and a peer node. A source node is responsible for converting the video stream into chunks and sending to the peers in the network. In our case, both the source nodes and peers run in a Docker containers atop the *qMp* nodes.

Setup: We use 20 real nodes connected to the wireless nodes of *qMp*. These nodes are co-located in either users homes (i.e., as home gateways, set-top-boxes, etc.) or within other infrastructures distributed around the city of Barcelona. As the controller node, we leverage the experimental infrastructure of Community-Lab[§]. Community-Lab provides a central coordination entity that has knowledge about the network topology in real time. The nodes of *qMp* that are running the live video streaming service are part of Community-Lab. In our experiments, we connect a live streaming camera (i.e., maximum bitrate of 512 kbps, 30 frame-per-second) to a local PeerStreamer instance that acts as a source node.

The location of the source in such a dynamic network is therefore crucial. Placing the source in a *qMp* node with weak connectivity will negatively impact the QoS and QoE of viewers. In order to determine the accuracy of *BASP* upon choosing the appropriate *qMp* node where to host the source, we measure the average chunk loss percentage at the peer side, which is defined as the percentage of chunks that were lost and not arrived in time. This simple metric will help us understand the role of

[‡]http://peerstreamer.org/

[§]`https://community-lab.net/`

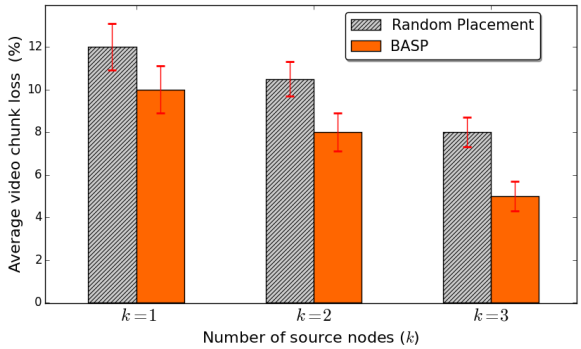


Figure 5.8: Average video chunk loss in qMp

the network on the reliable operation of live-video streaming over a CN.

Our experiment is composed of 20 runs, where each run has 10 repetitions. Results are averaged over all the successful runs. 90% of them were successful. In the 10% of failed runs, the source was unable to stream the captured images from the camera, so peers did not receive the data. This experiment was run for 2 weeks, with roughly 100 hours of live video data and several MBytes of logged content. The presented results are from one hour of continuous live streaming from the PeerStreamer source.

Results: Figure 5.8 shows the average chunk loss for an increasing number of sources k. The data reveals that for any number of source nodes k, *BASP* outperforms the currently adopted random placement in *qMp* network. For $k = 1$, *BASP* decreases the average chunk loss from 12% to 10%. This case corresponds to the scenario where there is one single source node streaming to the 20 peers in the *qMp* network. Based on the observations from the Figure 5.8, the gap between the two algorithms is growing as k increases. For instance, when $k = 3$, we get a 3% points of improvement w.r.t. chunk loss, and a significant 37% reduction in the loss packet rate.

5.6.2 Web 2.0 Service

The second type of service that we experiment is the Web 2.0 Service. The workloads of Web2.0 websites differ from the workloads of older generation websites. Older generation websites typically served static content, while Web2.0 websites serve dynamic

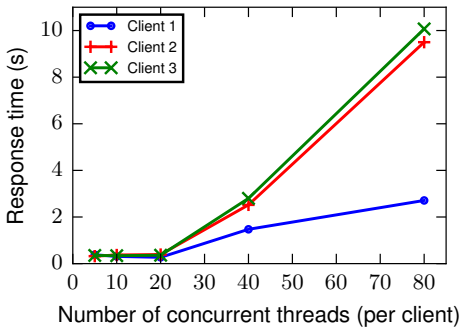


Figure 5.9: UpdateActivity when web server placed Randomly

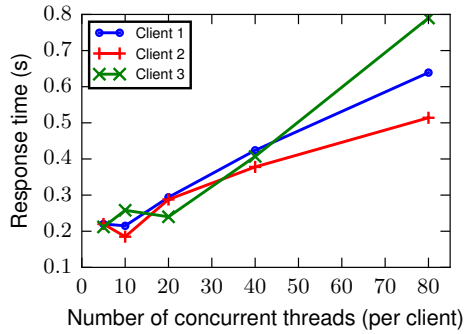


Figure 5.10: UpdateActivity when web server placed with BASP

content. The content is dynamically generated from the actions of other users and from external sources, such as news feeds from other websites. We are experimenting with a social networking service, which is an example of a Microservices architecture, since it is formed by a group of independently deployable service components (i.e., web server, database server, memcached server and clients). In this type of service, the placement of the web server (i.e., together with the database server) is decisive for the user QoS.

Setup: For the evaluation, we use the dockerized version of the CloudSuite Web Serving benchmark [PSF16]. Cloudsuite benchmark has four tiers: the web server, the database server, the memcached server, and the clients. Each tier has its own Docker image. The web server runs Elgg[¶] and it connects to the memcached server and the database server. The Elgg social networking engine is a Web2.0 application developed in PHP, similar in functionality to Facebook. The clients (i.e., implemented using the Faban workload generator) send requests to login to the social network and perform different operations. We use 10 available *qMp* nodes in total, where 3 of them act as clients. The other 7 nodes are candidates for deploying the web server. The web server, database server and memcached server are always collocated in the same host. On the client side, we measure the response time when performing some

[¶]https://elgg.org/

Table 5.3: Cloudsuite benchmark results

Operations	Update live feed				Do login			
Threads	10	20	40	80	10	20	40	80
qMp-Random	T	F	F	F	T	T	F	F
qMp-BASP	T	T	T	F	T	T	T	F
Stdev (sec)	0.02	0.03	0.01	0.01	0.02	0.02	0.01	0.03
Improvement (sec)	0.1	0.2	1.8	6.7	0.1	0.1	1.2	4.2

operations such as login, live feed update, message sending, etc. In Cloudsuite, to each operation is assigned an individual QoS latency limit. If less than 95% of the operations meet the QoS latency limit, the benchmark is considered to be failed (i.e., marked as F in the Table 5.3). The location of the web server, database server and memcached server has a direct impact on the client response time.

Results: Figure 5.9 and Figure 5.10 depicts the response time observed by three clients for the update live feed operation, when placing the web server with *Random* and *BASP*, respectively.

When placing the web server with the *Random* approach, Figure 5.9 reveals that, as far as we increase the number of threads (i.e., concurrent operations) per client, the response time increases drastically in three clients. For up to 120 operations per client (i.e., 20 threads), all clients perceive a similar response times (i.e., 300-350 ms). Response time increases more than one order of magnitude in Client2 and Client3, and an order of magnitude in Client1 when performing 160 operations (i.e., 80 threads).

Figure 5.10 depicts that, the client response times for higher workloads *decreases an order of magnitude* when using our *BASP* heuristic compared to the *Random* approach shown in the Figure 5.9. For up to 120 operations per client, the response times that three clients perceive is slightly better (i.e., 200-280 ms) than the response time when the web server is deployed with the *Random* approach. Furthermore, Table 5.3 demonstrates the successful and failed tests for the update and login operations in the Cloudsuite benchmark. Table reveals that, using the *BASP* heuristic the number of successful tests i.e., those that met the QoS latency limit, is higher than the number

of successful tests with the *Random* approach. Further, it also shows the standard deviation values and average client response time improvements (i.e., in seconds) when using *BASP* heuristic over *Random* approach. We can notice that the gain brought by the *BASP* heuristic is higher for more intensive workloads.

5.7 Discussion

We addressed the problem of workload placements in CN *micro-clouds* (i.e., IaaS-like infrastructures where resources are provided by citizens/organizations and interconnected through a CN). We saw that the major issue in these infrastructures are the uncertainties regarding the resources, as we derived from the network measurements (i.e., resources not uniformly distributed, heterogeneous devices in the network and constantly changing environment). Hence, this makes the efficient placement of workloads challenging. To solve this, we proposed a low-cost and fast heuristic *BASP* to allocate services taking into account the bandwidth of the network and the node availability.

Comparing to other works done in this field, this work takes into account the network characteristics i.e., bandwidth, while most of the service placement approaches generally consider only CPU and memory requirements. Second, it is applied to wireless networks, with all their specific challenges, including range and stability, whereas network-aware service placement are often applied to more traditional wired networks.

Bandwidth is a scarce resource in the CN environment. However, augmenting the bandwidth with the other network metrics (e.g., network latency and traffic) can help more accurately to places services in the network. Furthermore, in the presence of dynamic changes in the network and user mobility, migrating service instances (i.e., containers) from one *micro-cloud* node to another one can have beneficial results.

5.8 Summary

In this chapter, we motivated the need for bandwidth and availability-aware service placement in CN *micro-cloud* infrastructures. CNs provide a perfect scenario to deploy and use community services in contributory manner. Previous work done in CNs has focused on better ways to design the network to avoid hot spots and bottlenecks, but did not relate to schemes for network-aware placement of service instances.

However, as services become more network-intensive, they can become bottlenecked by the network, even in well-provisioned clouds. In the case of CN *micro-clouds*, network awareness is even more critical due to the limited capacity of nodes and links, and an unpredictable network performance. Without a network-aware system for placing services, locations with poor network paths may be chosen while locations with faster, more reliable paths remain unused, resulting ultimately in a poor user experience.

We proposed a low-complexity service placement heuristic called *BASP* to maximise the bandwidth allocation when deploying CN *micro-cloud* services. We presented algorithmic details, analysed its complexity, and carefully evaluated its performance with realistic settings. Our experimental results show that *BASP* consistently outperforms the currently adopted random placement in `Guifi.net` by 211% in bandwidth gain. Moreover, as the number of services increases, the gain tends to increase accordingly. Furthermore, we deployed our service placement algorithm in a real network segment of *qMp* network, a production CN, and quantified the performance and effects of our algorithm. We conducted our study on the case of a live video streaming service and Web 2.0 Service integrated through *Cloudy* distribution. Our real experimental results show that when using *BASP* algorithm, the video chunk loss in the peer side is decreased up to 3% points, i.e., worth a 37% reduction in the loss packet rate. When using the *BASP* with the Web 2.0 service, the client response times decreased up to an order of magnitude, which is a significant improvement.

Notes

The research discussed in Chapter 5 was included in the following publications:

[Sel+17] Mennan Selimi, Llorenç Cerdà-Alabern, Marc Sanchez-Artigas, Felix Freitag, and Luís Veiga. "Practical Service Placement Approach for Microservices Architecture". In: *17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid 2017)*. Madrid, Spain, June 2017.

[Sel+16] M. Selimi, L. Cerdà-Alabern, L. Wang, A. Sathiaseelan, L. Veiga, and F. Freitag. "Bandwidth-Aware Service Placement in Community Network Micro-Clouds". In: *IEEE 41st Conference on Local Computer Networks (LCN)*. Nov. 2016, pp. 220–223.

6 Conclusion

The work in this thesis focused on the service placement problem in community network (CN) *micro-cloud* environment. The hierarchical structure of CN *micro-clouds* allows to exploit and develop new algorithms for service placement problem which are more efficient than existing in-place approaches. Since CN *micro-clouds* bring much more infrastructure and user dynamics compared to a traditional cloud environment, we proposed low-complexity heuristics for service placement algorithms to cope with these dynamics.

As our contribution in this field, first we presented the current state of service deployment in `Guifi.net` CN and performed in-depth performance assessment of three type of popular services in these environments, such as distributed storage, live video-streaming and service discovery. Based on the performance and feasibility studies that we performed on these services, we proposed service placement algorithms for the CN environment.

We conducted extensive evaluations, employing simulations and real-world experiments and were able to demonstrate that our placement algorithms, in particular *PASP* (Policy-aware Service Placement) and *BASP* (Bandwidth and Availability-aware

Service Placement) outperform the existing in-place approaches in the `Guifi.net` CN with respect to end-to-end bandwidth, latency and availability when used with CN *micro-cloud* services.

6.1 Future Work

The service placement problem presented in this work addresses the major concerns of placing a service in the wireless CN *micro-cloud* environment. Given that our algorithms has proven itself in a variety of experiments, it can be expected to serve as a strong basis to investigate the following topics listed in this section.

6.1.1 Composite Service Placement

The main focus of this work was the service placement of monolithic services. Our current work does not cover the placement of composite services. The placement of composite services is more complex and depends largely on the interactions and the semantics between the sub-services. This requires more sophisticated placement algorithms which include the flow of information between instances of sub-services into their model of the network. In order to achieve this, a deeper understanding is needed on how the sub-services interact with each other through observations at run time.

6.1.2 Distributed Decision Making

In our algorithms presented in the thesis, decisions on service placement are made by a centralized controller (i.e., client node or Community-Lab controller). This is very practical because we can always see the controller as a service running at one or multiple CN *micro-clouds*. However, it is desirable to have a distributed control mechanism for the sake of robustness. Issues regarding distributed service placement can be studied in the future, where randomization techniques such as in [Nee16] can be used.

6.1.3 Service Migration

As the network topology, network performance or volume of service requests from different clients change over time, the current service configuration needs to be adapted to the new situation. This means migrating the services from one CN *micro-cloud* node to another one. However, migrating services or creating new service instances incurs additional network traffic. For this reason, the exact circumstances under which an adaptation actually makes sense are not trivial if the goal is the overall reduction of network traffic. Issues regarding the service migration in edge-clouds can be studied as in [Wan15] [Urg+15b].

6.1.4 Security

Distributed service provisioning involves problems from a security perspective. Instead of only having to trust a single central instance, each client essentially has to extend its trust to all nodes of the CN *micro-cloud* since each node is a potential service host. For a more advanced technical readiness of service placement in systems, this significant aspect needs to be well understood.

Bibliography

[Bai+15a] Roger Baig, Ramon Roca, Felix Freitag, and Leandro Navarro. "guifi.net, a crowdsourced network infrastructure held in common". In: *Computer Networks* 90 (2015). Crowdsourcing, pp. 150–165 (cit. on pp. 2, 15).

[Bai+16] Roger Baig, Lluis Dalmau, Ramon Roca, Leandro Navarro, Felix Freitag, and Arjuna Sathiaseelan. "Making Community Networks Economically Sustainable, the Guifi.Net Experience". In: *Proceedings of the 2016 Workshop on Global Access to the Internet for All*. GAIA '16. Florianopolis, Brazil: ACM, 2016, pp. 31–36 (cit. on p. 2).

[Sel+15a] Mennan Selimi, Amin M Khan, Emmanouil Dimogerontakis, Felix Freitag, and Roger Pueyo Centelles. "Cloud services in the Guifi.net community network". In: *Computer Networks* 93.P2 (Dec. 2015), pp. 373–388 (cit. on pp. 2, 27, 86).

[Sel+15b] M. Selimi, N. 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 2015)*. Nov. 2015, pp. 202–209 (cit. on pp. 3, 8, 72, 88).

[Bon+12] Flavio Bonomi, Rodolfo Milito, Jiang Zhu, and Sateesh Addepalli. "Fog Computing and Its Role in the Internet of Things". In: *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing*. MCC '12. Helsinki, Finland: ACM, 2012, pp. 13–16 (cit. on p. 3).

[NLN15] Axel Neumann, Ester López, and Leandro Navarro. "Evaluation of mesh routing protocols for wireless community networks". In: *Computer Networks* 93, Part 2 (2015). Community Networks, pp. 308–323 (cit. on pp. 4, 14).

[03] *Optimized Link State Routing Protocol (OLSR), RFC 3626*. 2003. URL: `https://tools.ietf.org/html/rfc3626` (cit. on p. 4).

[Wit10] Georg Wittenburg. "Service Placement in Ad Hoc Networks". PhD thesis. Berlin, Germany: Department of Mathematics and Computer Science, Freie Universität Berlin, 2010 (cit. on pp. 4, 32).

[CNE13] 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. Barcelona, Spain: ACM, 2013, pp. 23–30 (cit. on pp. 4, 16).

[LaC13] Katrina et al. LaCurts. "Choreo: Network-aware Task Placement for Cloud Applications". In: *Proceedings of the 2013 Conference on Internet Measurement Conference*. IMC '13. Barcelona, Spain: ACM, 2013, pp. 191–204 (cit. on pp. 4, 31, 34).

[Veg+12] Davide Vega, Llorenc Cerda-Alabern, Leandro Navarro, and Roc Meseguer. "Topology patterns of a community network: Guifi.net". In: *1st International Workshop on Community Networks and Bottom-up-Broadband (CNBuB 2012), within IEEE WiMob*. Barcelona, Spain: IEEE, Oct. 2012, pp. 612–619 (cit. on pp. 5, 17, 63).

[Veg+15] Davide Vega, Roger Baig, Llorenç Cerdà-Alabern, Esunly Medina, Roc Meseguer, and Leandro Navarro. "A technological overview of the guifi.net community network". In: *Computer Networks* 93, Part 2 (2015). Community Networks, pp. 260–278 (cit. on pp. 5, 15).

[Apo+15] N. Apolónia, R. Sedar, F. Freitag, and L. Navarro. "Leveraging Low-Power Devices for Cloud Services in Community Networks". In: *2015 3rd International Conference on Future Internet of Things and Cloud*. Aug. 2015, pp. 363–370 (cit. on p. 5).

[Sel+16a] Mennan Selimi, Davide Vega, Felix Freitag, and Luís Veiga. "Towards Network-Aware Service Placement in Community Network Micro-Clouds". In: *22nd International Conference on Parallel and Distributed Computing (Euro-Par 2016)*. Grenoble, France, 2016, pp. 376–388 (cit. on pp. 8, 89).

[Kha16] Amin M. Khan. "Managing Incentives in Community Network Clouds". PhD thesis. Barcelona, Spain: Department of Computer Architecture, Universitat Politècnica de Catalunya, BarcelonaTech, 2016 (cit. on p. 10).

[DMN17] Emmanouil Dimogerontakis, Roc Meseguer, and Leandro Navarro. "Internet Access for All: Assessing a Crowdsourced Web Proxy Service in a Community Network". In: *Passive and Active Measurement Conference (PAM)*. 2017 (cit. on pp. 15, 39).

[16a] *QMP Network*. 2016. URL: http://qmp.cat/ (cit. on p. 16).

[17a] *OpenWRT*. 2017. URL: https://openwrt.org/ (cit. on p. 16).

[NLN12] A. Neumann, E. Lopez, and L. Navarro. "An evaluation of BMX6 for community wireless networks". In: *8th IEEE International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), 2012 I*. Oct. 2012, pp. 651–658 (cit. on p. 16).

[Bai+15b] Roger Baig, Rodrigo Carbajales, Pau Escrich Garcia, Jorge L. Florit, Felix Freitag, Agusti Moll, Leandro Navarro, Ermanno Pietrosemoli, Roger Pueyo Centelles, Mennan Selimi, Vladimir Vlassov, and Marco Zennaro. "The cloudy distribution in community network clouds in Guifi.net". In: *IFIP/IEEE International Symposium on Integrated Network Management, (IM 2015), Ottawa, ON, Canada, 11-15 May, 2015*. 2015, pp. 1161–1162 (cit. on pp. 16, 19).

[17b] *LXC: Linux Containers*. 2017. URL: https://linuxcontainers.org/ (cit. on p. 16).

[17c] *Docker: Containerization Platform*. 2017. URL: https://www.docker.com/ (cit. on p. 16).

[16b] *Guifi.net CNML file*. 2016. URL: https://guifi.net/en/guifi/cnml/2413 (cit. on p. 17).

[17d] *Cloudy: A community networking cloud in a box*. 2017. URL: http://cloudy.community/ (cit. on p. 19).

[17e] *OpenVZ Linux Containers*. 2017. URL: http://openvz.org/ (cit. on p. 20).

[17f] *Avahi Service Discovery Tool*. 2017. URL: http://avahi.org/ (cit. on pp. 21, 30).

[17g] *Tinc Virtual Private Network*. 2017. URL: http://tinc-vpn.org/ (cit. on pp. 21, 30).

[17h] *Serf: a tool for cluster membership, failure detection, and orchestration*. 2017. URL: https://www.serfdom.io/ (cit. on p. 21).

[WW08] Zooko Wilcox-O'Hearn and Brian Warner. "Tahoe: The Least-authority Filesystem". In: *Proceedings of the 4th ACM International Workshop on Storage Security and Survivability*. StorageSS '08. Alexandria, Virginia, USA: ACM, 2008, pp. 21–26 (cit. on pp. 22, 27, 43).

[17i] *Etcd key-value store*. 2017. URL: https://github.com/coreos/etcd (cit. on p. 22).

[17j] *Syncthing*. 2017. URL: http://syncthing.net/ (cit. on p. 22).

[16c] *PeerStreamer: fast and efficient P2P streaming*. 2016. URL: http://peerstreamer.org/ (cit. on pp. 22, 28, 50).

[Wan15] Shiqiang Wang. "Dynamic Service Placement in Mobile Micro-Clouds". PhD thesis. London, UK: Department of Electrical and Electronic Engineering, Imperial College London, 2015 (cit. on pp. 23, 109).

[Ver11] Vedat Verter. "Uncapacitated and Capacitated Facility Location Problems". In: *Foundations of Location Analysis*. Ed. by H. A. Eiselt and Vladimir Marianov. Boston, MA: Springer US, 2011, pp. 25–37 (cit. on pp. 24, 30).

[MG11] Peter Mell and Timothy Grance. "The NIST Definition of Cloud Computing". In: *NIST Special Publication* 800.145 (2011) (cit. on pp. 25, 41).

[And04] David P Anderson. "BOINC : A System for Public-Resource Computing and Storage". In: *Fifth IEEE/ACM International Workshop on Grid Computing*. Pittsburgh, USA, Nov. 2004, pp. 4–10 (cit. on p. 25).

[Beb+09] Adam L. Beberg, Daniel L. Ensign, Guha Jayachandran, Siraj Khaliq, and Vijay S. Pande. "Folding@home: Lessons from eight years of volunteer distributed computing". In: *2009 IEEE International Symposium on Parallel & Distributed Processing*. Rome, Italy, May 2009 (cit. on p. 25).

[Chu+03] Brent Chun, David Culler, Timothy Roscoe, Andy Bavier, Larry Peterson, Mike Wawrzoniak, and Mic Bowman. "PlanetLab: An Overlay Testbed for Broad-Coverage Services". In: *ACM SIGCOMM Computer Communication Review* 33.3 (July 2003), pp. 3–12 (cit. on p. 25).

[Cap+09] Justin Cappos, Ivan Beschastnikh, Arvind Krishnamurthy, and Tom Anderson. "Seattle: a platform for educational cloud computing". In: *40th ACM Technical Symposium on Computer Science Education (SIGCSE '09)*. Chattanooga, USA, 2009, pp. 111–115 (cit. on pp. 25, 26).

[MB09] Alexandros Marinos and Gerard Briscoe. "Community Cloud Computing". In: *1st International Conference on Cloud Computing (CloudCom 2009)*. Ed. by MartinGilje Jaatun, Gansen Zhao, and Chunming Rong. Vol. 5931. LNCS. Beijing, China: Springer Berlin Heidelberg, Dec. 2009, pp. 472–484 (cit. on p. 25).

[DP12] Salvatore Distefano and Antonio Puliafito. "Cloud@Home: Toward a Volunteer Cloud". In: *IT Professional* 14.1 (Jan. 2012), pp. 27–31 (cit. on p. 25).

[Yi+11] Sangho Yi, E. Jeannot, D. Kondo, and D.P. Anderson. "Towards Real-Time, Volunteer Distributed Computing". In: *11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid'11)*. May 2011, pp. 154–163 (cit. on p. 26).

[BMT12] Ozalp Babaoglu, Moreno Marzolla, and Michele Tamburini. "Design and Implementation of a P2P Cloud System". In: *Proceedings of the 27th Annual ACM Symposium on Applied Computing*. SAC '12. Trento, Italy: ACM, 2012, pp. 412–417 (cit. on p. 26).

[Cha+12] Kyle Chard, Kris Bubendorfer, Simon Caton, and Omer F. Rana. "Social Cloud Computing: A Vision for Socially Motivated Resource Sharing". In: *IEEE Transactions on Services Computing* 5.4 (Jan. 2012), pp. 551–563 (cit. on p. 26).

[Pun+13] Magdalena Punceva, Ivan Rodero, Manish Parashar, Omer F. Rana, and Ioan Petri. "Incentivising resource sharing in social clouds". In: *Concurrency and Computation: Practice and Experience* (Mar. 2013) (cit. on p. 26).

[Cat+14] Simon Caton, Christian Haas, Kyle Chard, Kris Bubendorfer, and Omer F. Rana. "A Social Compute Cloud: Allocating and Sharing Infrastructure Resources via Social Networks". In: *IEEE Transactions on Services Computing* 7.3 (July 2014), pp. 359–372 (cit. on p. 26).

[GSF13] Mark Gall, Angelika Schneider, and Niels Fallenbeck. "An Architecture for Community Clouds Using Concepts of the Intercloud". In: *27th International Conference on Advanced Information Networking and Applications (AINA'13)*. Barcelona, Spain: IEEE, Mar. 2013, pp. 74–81 (cit. on p. 26).

[BRC10] Rajkumar Buyya, Rajiv Ranjan, and Rodrigo N Calheiros. "InterCloud: Utility-Oriented Federation of Cloud Computing Environments for Scaling of Application Services". In: *Algorithms and Architectures for Parallel Processing*. Lecture Notes in Computer Science 6081 (Mar. 2010), pp. 20–31 (cit. on p. 26).

[Esp+13] Christian Esposito, Massimo Ficco, Francesco Palmieri, and Aniello Castiglione. "Interconnecting Federated Clouds by Using Publish-Subscribe Service". In: *Cluster Computing* 16.4 (Apr. 2013), pp. 887–903 (cit. on p. 26).

[ZLL14] Han Zhao, Xinxin Liu, and Xiaolin Li. "Towards efficient and fair resource trading in community-based cloud computing". In: *Journal of Parallel and Distributed Computing* 74.11 (Aug. 2014), pp. 3087–3097 (cit. on p. 26).

[Jan+14] Minsung Jang, Karsten Schwan, Ketan Bhardwaj, Ada Gavrilovska, and Adhyas Avasthi. "Personal clouds: Sharing and integrating networked resources to enhance end user experiences". In: *33rd Annual IEEE International Conference on Computer Communications (INFOCOM'14)*. Toronto, Canada: IEEE, Apr. 2014, pp. 2220–2228 (cit. on p. 26).

[Che+13] Y. F. Chen, S. Daniels, M. Hadjieleftheriou, P. Liu, C. Tian, and V. Vaishampayan. "Distributed storage evaluation on a three-wide inter-data center deployment". In: *2013 IEEE International Conference on Big Data*. Oct. 2013, pp. 17–22 (cit. on p. 27).

[16d] *QFS-Quantcast File System*. 2016. URL: `https://quantcast.github.io/qfs/` (cit. on p. 27).

[16e] *OpenStack Swift*. 2016. URL: `https://www.swiftstack.com/product/openstack-swift` (cit. on p. 27).

[Gra+13] R. Gracia-Tinedo, M. S. Artigas, A. Moreno-Martínez, C. Cotes, and P. G. López. "Actively Measuring Personal Cloud Storage". In: *2013 IEEE Sixth International Conference on Cloud Computing*. June 2013, pp. 301–308 (cit. on p. 27).

[Tse+12] Fan-Hsun Tseng, Chi-Yuan Chen, Li-Der Chou, and Han-Chieh Chao. "Implement a reliable and secure cloud distributed file system". In: *Intelligent Signal Processing and Communications Systems (ISPACS), 2012 International Symposium on*. 2012, pp. 227–232 (cit. on p. 27).

[SD12] K. Shima and N. Dang. *Indexes for Distributed File/Storage Systems as a Large Scale Virtual Machine Disk Image Storage in a Wide Area Network*. 2012 (cit. on p. 27).

[16f] *XtreemFS - Fault-Tolerant Distributed File System*. 2016. URL: http://www.xtreemfs.org/ (cit. on p. 28).

[16g] *Ceph - distributed object store*. 2016. URL: http://ceph.com/ (cit. on p. 28).

[16h] *Gluster - scalable network file system*. 2016. URL: https://www.gluster.org/ (cit. on p. 28).

[16i] *Sheepdog project*. 2016. URL: https://sheepdog.github.io/sheepdog/ (cit. on p. 28).

[10] "Towards Transactional Load over XtreemFS". In: *CoRR* abs/1001.2931 (2010). Withdrawn. (cit. on p. 28).

[BML14] 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 (cit. on p. 28).

[BMC15] Luca Baldesi, Leonardo Maccari, and Renato Lo Cigno. "Improving {P2P} streaming in Wireless Community Networks". In: *Computer Networks* 93, Part 2 (2015). Community Networks, pp. 389–403 (cit. on p. 28).

[16j] *CONFINE Project: Community Networks Testbed for the Future Internet*. 2016. URL: http://confine-project.eu/ (cit. on p. 28).

[Ala13] Amr Alasaad. "Content sharing and distribution in wireless community networks". English. In: Univeristy of British Columbia, PhD Thesis, 2013 (cit. on p. 28).

[Tra+12] 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)*. Sept. 2012, pp. 13–24 (cit. on pp. 28, 50).

[RC13] 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)*. Mar. 2013 (cit. on p. 29).

[Oli+13] João Oliveira et al. "Can Peer-to-Peer live streaming systems coexist with free riders?" In: *P2P'13*. 2013, pp. 1–5 (cit. on p. 29).

[Cou+11] A.P. Couto da Silva, E. Leonardi, M. Mellia, and M MEO. "Exploiting Heterogeneity in P2P Video Streaming". In: *IEEE Transactions on Computers* 60 (May 2011), 667–679 (cit. on p. 29).

[DS10] A Dittrich and Felix Salfner. "Experimental responsiveness evaluation of decentralized service discovery". In: *Parallel Distributed Processing, Workshops and Phd Forum (IPDPSW), 2010 IEEE International Symposium on*. Apr. 2010, pp. 1–7 (cit. on p. 29).

[DMQ07] C. Dabrowski, K. Mills, and S. Quirolgico. "Understanding failure response in service discovery systems". In: *Journal of Systems and Software* 80.6 (2007), pp. 896–917 (cit. on p. 29).

[Lee+07] Jae Woo Lee et al. "z2z: Discovering Zeroconf Services Beyond Local Link". In: *Globecom Workshops, 2007 IEEE*. Nov. 2007, pp. 1–7 (cit. on p. 29).

[Oh+04] Chang-Seok Oh et al. "A Hybrid Service Discovery for Improving Robustness in Mobile Ad Hoc Networks". In: *IEEE International Conference on Dependable Systems and Networks (DSN 2004)*. 2004 (cit. on p. 29).

[CG06] Celeste Campo and Carlos García-Rubio. "DNS-Based Service Discovery in Ad Hoc Networks: Evaluation and Improvements". English. In: *Personal Wireless Communications*. Ed. by Pedro Cuenca and Luiz Orozco-Barbosa. Vol. 4217. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2006, pp. 111–122 (cit. on p. 30).

[Wir+12] Hanno Wirtz, Tobias Heer, Martin Serror, and Klaus Wehrle. "DHT-based localized service discovery in wireless mesh networks." In: *MASS*. 2012, pp. 19–28 (cit. on p. 30).

[Dit+14] Andreas Dittrich, Björn Lichtblau, Rafael Rezende, and Miroslaw Malek. "Modeling Responsiveness of Decentralized Service Discovery in Wireless Mesh Networks". English. In: *Measurement, Modelling, and Evaluation of Computing Systems and Dependability and Fault Tolerance*. Ed. by Kai Fischbach and UdoR. Krieger. Vol. 8376. Lecture Notes in Computer Science. Springer International Publishing, 2014, pp. 88–102 (cit. on p. 30).

[Her10] Klaus Herrmann. "Self-organized Service Placement in Ambient Intelligence Environments". In: *ACM Trans. Auton. Adapt. Syst.* 5.2 (May 2010), 6:1–6:39 (cit. on p. 31).

[Aga+10] Sharad Agarwal et al. "Volley: Automated Data Placement for Geo-distributed Cloud Services". In: *Proceedings of the 7th USENIX Conference on Networked Systems Design and Implementation*. NSDI'10. San Jose, California: USENIX Association, 2010, pp. 2–2 (cit. on p. 31).

[Gha+14] H. Ghanbari, M. Litoiu, P. Pawluk, and C. Barna. "Replica Placement in Cloud through Simple Stochastic Model Predictive Control". In: *2014 IEEE 7th International Conference on Cloud Computing*. June 2014, pp. 80–87 (cit. on p. 31).

[SG12] Moritz Steiner and Bob et al. Gaglianello. "Network-aware Service Placement in a Distributed Cloud Environment". In: *Proceedings of the ACM SIGCOMM 2012 Conference*. SIGCOMM '12. Helsinki, Finland: ACM, 2012, pp. 73–74 (cit. on p. 31).

[KIH12] Adrian Klein, Fuyuki Ishikawa, and Shinichi Honiden. "Towards Network-aware Service Composition in the Cloud". In: *Proceedings of the 21st International Conference on World Wide Web*. WWW '12. Lyon, France: ACM, 2012, pp. 959–968 (cit. on p. 31).

[AL12] M. Alicherry and T.V. Lakshman. "Network aware resource allocation in distributed clouds". In: *Proceedings of INFOCOM, IEEE*. Mar. 2012, pp. 963–971 (cit. on p. 31).

[Moe+14] H. Moens, B. Hanssens, B. Dhoedt, and F. De Turck. "Hierarchical network-aware placement of service oriented applications in Clouds". In: *2014 IEEE Network Operations and Management Symposium (NOMS)*. May 2014, pp. 1–8 (cit. on p. 31).

[SBL15] B. Spinnewyn, B. Braem, and S. Latré. "Fault-tolerant application placement in heterogeneous cloud environments". In: *Network and Service Management (CNSM)*. Nov. 2015, pp. 192–200 (cit. on p. 31).

[Dub+15] D. J. Dubois, G. Valetto, D. Lucia, and E. Di Nitto. "Mycocloud: Elasticity through Self-Organized Service Placement in Decentralized Clouds". In: *2015 IEEE 8th International Conference on Cloud Computing*. June 2015, pp. 629–636 (cit. on p. 32).

[Tär+16] William Tärneberg, Amardeep Mehta, Eddie Wadbro, Johan Tordsson, Johan Eker, Maria Kihl, and Erik Elmroth. "Dynamic application placement in the Mobile Cloud Network". In: *Future Generation Computer Systems* (2016) (cit. on p. 32).

[Tan16] A. N. Tantawi. "Solution Biasing for Optimized Cloud Workload Placement". In: *2016 IEEE International Conference on Autonomic Computing (ICAC)*. July 2016, pp. 105–110 (cit. on p. 32).

[Urg+15a] Rahul Urgaonkar, Shiqiang Wang, Ting He, Murtaza Zafer, Kevin Chan, and Kin K. Leung. "Dynamic service migration and workload scheduling in edge-clouds". In: *Performance Evaluation* 91 (2015), pp. 205–228 (cit. on p. 32).

[Wan+15a] Shiqiang Wang et al. "Dynamic service placement for mobile micro-clouds with predicted future costs". In: *IEEE International Conference on Communications (ICC)*. June 2015, pp. 5504–5510 (cit. on p. 32).

[Urg+15b] Rahul Urgaonkar et al. "Dynamic service migration and workload scheduling in edge-clouds". In: *Performance Evaluation* 91 (2015). Special Issue: Performance 2015, pp. 205–228 (cit. on pp. 32, 109).

[Mac+16] Andrew Machen, Shiqiang Wang, Kin K. Leung, Bong Jun Ko, and Theodoros Salonidis. "Migrating Running Applications Across Mobile Edge Clouds: Poster". In: *Proceedings of the 22Nd Annual International Conference on Mobile Computing and Networking*. MobiCom '16. New York City, New York: ACM, 2016, pp. 435–436 (cit. on p. 32).

[Cab14] Guillem Cabrera Añon. "Service allocation methodologies for contributory computing environments". PhD thesis. Barcelona, Spain: Universitat Oberta de Catalunya. Escola de Doctorat, 2014 (cit. on p. 33).

[Nov+15] P. Novotny, R. Urgaonkar, A. L. Wolf, and B. Ko. "Dynamic placement of composite software services in hybrid wireless networks". In: *IEEE Military Communications Conference (MILCOM 2015)*. Oct. 2015, pp. 1052–1057 (cit. on p. 33).

[Veg+14] D. Vega, R. Meseguer, G. Cabrera, and J.M. Marques. "Exploring local service allocation in Community Networks". In: *10th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob'14), IEEE*. Oct. 2014, pp. 273–280 (cit. on p. 33).

[Sel+16b] Mennan Selimi, Llorenç Cerdà-Alabern, Liang Wang, Arjuna Sathiaseelan, Luis Veiga, and Felix Freitag. "Bandwidth-aware Service Placement in Community Network Clouds". In: *CoRR* abs/1603.08119 (2016) (cit. on p. 35).

[Sel+17] Mennan Selimi, Llorenç Cerdà-Alabern, Marc Sanchez-Artigas, Felix Freitag, and Luís Veiga. "Practical Service Placement Approach for Microservices Architecture". In: *17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid 2017)*. Madrid, Spain, June 2017 (cit. on p. 35).

[16k] *Guifi.net: Services of Catalunya (by zone)*. 2016. URL: `https://guifi.net/en/node/2413/view/services` (cit. on p. 38).

[Eli+09] Fotios A. Elianos, Georgia Plakia, Pantelis A. Frangoudis, and George C. Polyzos. "Structure and evolution of a large-scale Wireless Community Network". In: *2009 IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks*. IEEE, June 2009 (cit. on p. 39).

[16l] *AWMN - Athens Wireless Metropolitan Network*. 2016. URL: `http://www.awmn.net/` (cit. on p. 39).

[16m] *Proxmox: Server-Virtualization with KVM and Containers*. 2016. URL: `https://www.proxmox.com/` (cit. on p. 40).

[16n] *Guinux*. 2016. URL: `https://guifi.net/en/node/29320` (cit. on p. 40).

[Sel+14] M. Selimi, F. Freitag, R. P. Centelles, and A. Moll. "Distributed Storage and Service Discovery for Heterogeneous Community Network Clouds". In: *2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing (UCC 2014)*. Dec. 2014, pp. 204–212 (cit. on p. 40).

[160] *Guifi TV*. 2016. URL: `http://project.Guifi.net/projects/Guifitv` (cit. on p. 40).

[17k] *Community-Lab: Community Networks Testbed by the CONFINE Project*. 2017. URL: `http://community-lab.net/` (cit. on p. 41).

[Bra+13] Bart Braem et al. "A case for research with and on community networks". In: *ACM SIGCOMM Computer Communication Review* 43.3 (July 2013), pp. 68–73 (cit. on p. 41).

[17l] *FEDERICA: Federated E-infrastructure Dedicated to European Researchers Innovating in Computing network Architectures*. 2017. URL: http://www.fp7-federica.eu/ (cit. on p. 41).

[C+11] Brad Calder, Ju Wang, et al. "Windows Azure Storage: A Highly Available Cloud Storage Service with Strong Consistency". In: *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles*. SOSP '11. Cascais, Portugal: ACM, 2011, pp. 143–157 (cit. on p. 42).

[16p] *IOzone: a filesystem benchmark tool*. 2016. URL: http://www.iozone.org/ (cit. on p. 44).

[17m] *Microsoft Azure: Cloud Computing Platform and Services*. 2017. URL: https://azure.microsoft.com/en-us/ (cit. on p. 45).

[Dim+17] Emmanouil Dimogerontakis, Joao Neto, Roc Meseguer, and Leandro Navarro. "Client-Side Routing-Agnostic Gateway Selection for heterogeneous Wireless Mesh Networks". In: *IFIP/IEEE International Symposium on Integrated Network Management (IM)*. 2017 (cit. on p. 49).

[16q] *Sopcast: P2P Internet TV*. 2016. URL: http://www.sopcast.com/ (cit. on p. 49).

[B+11] R. Birke, E. Leonardi, M. Mellia, et al. "Architecture of a network-aware P2P-TV application: the NAPA-WINE approach". In: *Communications Magazine, IEEE* 49.6 (June 2011), pp. 154–163 (cit. on pp. 51, 55).

[15] *Zero Configuration Networking (Zeroconf)*. 2015. URL: http://www.zeroconf.org/ (cit. on p. 60).

[Sel+15c] M. Selimi, F. Freitag, R. P. Centelles, A. Moll, and L. Veiga. "TROBADOR: Service Discovery for Distributed Community Network Micro-Clouds". In: *2015 IEEE 29th International Conference on Advanced Information Networking and Applications (AINA 2015)*. Mar. 2015, pp. 642–649 (cit. on p. 66).

[Ryd+14] M. Ryden et al. "Nebula: Distributed Edge Cloud for Data Intensive Computing". In: *IEEE International Conference on Cloud Engineering (IC2E), 2014*. Mar. 2014, pp. 57–66 (cit. on p. 72).

[Wan+15b] Liang Wang, Suzan Bayhan, Jörg Ott, Jussi Kangasharju, Arjuna Sathiaseelan, and Jon Crowcroft. "Pro-Diluvian: Understanding Scoped-Flooding for Content Discovery in Information-Centric Networking". In: *Proceedings of the 2nd International Conference on Information-Centric Networking*. ICN '15. San Francisco, California, USA: ACM, 2015, pp. 9–18 (cit. on p. 80).

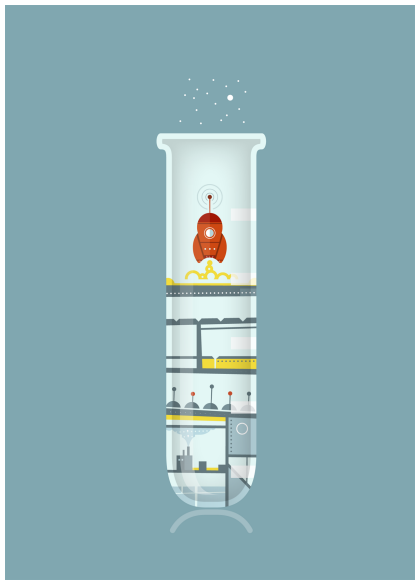
[VP11] H. Verespej and J. Pasquale. "A Characterization of Node Uptime Distributions in the PlanetLab Test Bed". In: *2011 IEEE 30th International Symposium on Reliable Distributed Systems*. Oct. 2011, pp. 203–208 (cit. on p. 85).

[16r] *Stirling Number of the Second Kind*. 2016. URL: `http://mathworld.wolfram.com/StirlingNumberoftheSecondKind.html` (cit. on p. 92).

[LF09] Andrea Lancichinetti and Santo Fortunato. "Community detection algorithms: A comparative analysis". In: *Phys. Rev. E* 80 (5 Nov. 2009), p. 056117 (cit. on p. 93).

[PSF16] Tapti Palit, Yongming Shen, and Michael Ferdman. "Demystifying Cloud Benchmarking". In: *2016 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*. Apr. 2016, pp. 122–132 (cit. on p. 101).

[Nee16] M. J. Neely. "Distributed Stochastic Optimization via Correlated Scheduling". In: *IEEE/ACM Transactions on Networking* 24.2 (Apr. 2016), pp. 759–772 (cit. on p. 108).



This thesis was typeset using LaTeX, originally developed by Leslie Lamport and based on Donald Knuth's TeX. The body text is set in 11 point Minion Pro, designed by Robert Slimbach in 1990 inspired by late Renaissance-era type and issued by Adobe in 2000. The headlines and captions are set in variations of Myriad Pro, a humanist sans-serif typeface designed by Robert Slimbach and Carol Twombly in 1990 and issued by Adobe in 2000. The above illustration was created by Ben Schlitter and released under CC BY-NC-ND 4.0. A template that can be used to format a PhD dissertation with this look & feel has been released under the permissive AGPL license, and can be found online at github.com/aminmkhan/Dissertate or from its lead author, Jordan Suchow, at suchow@post.harvard.edu.