Towards Network-Aware Service Placement in Community Network Micro-Clouds

Mennan Selimi[1,3(✉)], Davide Vega[2], Felix Freitag[1], and Luís Veiga[3]

[1] Universitat Politècnica de Catalunya, Barcelonatech, Barcelona, Spain
{mselimi,felix}@ac.upc.edu
[2] University of Bologna, Bologna, Italy
davide.vegadaurelio@unibo.it
[3] INESC-ID Lisboa/Instituto Superior Técnico, University of Lisbon, Lisbon, Portugal
luis.veiga@inesc-id.pt

Abstract. Cloud services in community networks have been enabled by micro-cloud providers. They form community network micro-clouds (CNMCs), which grow organically, i.e. without being planned and optimized beforehand. Services running in community networks face specific challenges intrinsic to these infrastructures, such as the limited capacity of nodes and links, their dynamics and geographic distribution. CNMCs are used to deploy distributed applications, such as streaming and storage services, which transfer significant amounts of data between the nodes on which they run. Currently there is no support given to users for enabling them to chose better or the best option for specific service deployments. This paper looks at the next step in community network cloud service deployments, by taking network characteristics into account when deciding placement of service instances. We propose a service placement algorithm (PASP) that minimizes the service overlay diameter, while fulfilling service specific criteria. First, we characterize with simulations the potential performance gains of our approach. Secondly, we apply our algorithm to deploy a distributed storage service currently used in Guifi.net, and evaluate it in the real production network, assessing the performance and effects of our algorithm. We find that our PASP algorithm reduces the client reading times by an average of 16 % (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.

Keywords: Community network micro-clouds · Service placement

1 Introduction

Community networks or Do-It-Yourself networks (DIYs) are bottom-up built decentralized networks, deployed and maintained by their own users. One successful effort of such a network is Guifi.net[1], located in the Catalonia region of

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

DOI: 10.1007/978-3-319-43659-3_28

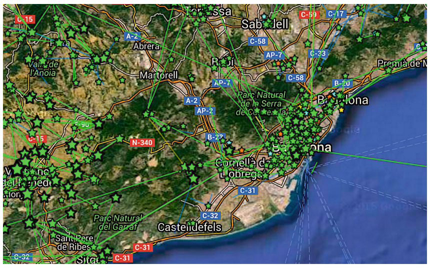

Fig. 1. Guifi.net nodes and links in Barcelona

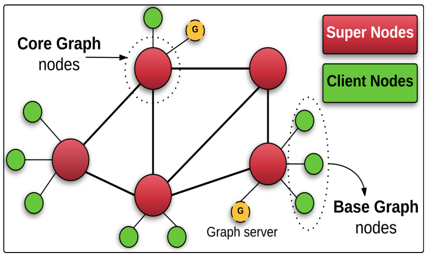

Fig. 2. Guifi.net topology

Spain. Guifi.net is defined as an open, free and neutral community network built by its members: citizens and organizations pooling their resources and coordinating efforts to build and operate a local network infrastructure. Guifi.net network started in 2004 and today it has more than 30.000 operational nodes, which makes it the largest community network worldwide [1]. Figure 1 shows as example the nodes and links of Guifi.net in the city of Barcelona. Figure 2 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.

Until recently, user-oriented local services were not much deployed because of the lack of easy to use mechanisms to exploit the available resources within the community network and due to other technological barriers. Early services include GuifiTV, Graph servers, mail servers, game servers [1]. With the adoption of *community network micro-clouds* (CNMC)[2], i.e. the platform that enables cloud-based services in community networks, local user-oriented services gained a huge momentum. Community network users started creating their own home-grown services and using alternative open source software for some of today's Internet cloud services, e.g., data storage services, interactive applications such as Voice-over-IP (VoIP), video streaming, P2P-TV, [2,3]. In a CNMC, a server (i.e. a low-power device such as a enhanced home gateway or mini-PC) is connected to a node of the community network.

Since Guifi.net nodes and the connected servers are geographically distributed, it needs to be decided where services should be placed in a network. If the underlying network resources are not taken into account, 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. Therefore, a key challenge in community network micro-clouds is to determine the location of the service deployments, i.e. which servers at a certain geographic points in the network. Due to the dynamic nature of community networks and usage patterns, it is challenging to calculate an optimal placement.

[2] http://cloudy.community/.

In this paper we aim at understanding the impact of network-aware service placement decisions on end-to-end client performance. The main contributions of this paper can be summarized as follows:

- We introduce a service placement algorithm that provides optimal service overlay selection without the need to verify the whole solution space. The algorithm finds the minimum possible distance in terms of the number of hops between two furthest selected resources, and at the same time fulfil different service type quality criteria.
- We extensively study the effectiveness of our approach in simulations using real-world node and usage traces from 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 community network and quantify the performance and effects of our algorithm with a distributed storage service.

2 System Model

2.1 Network Structure

The Guifi.net community network consists of a set of *nodes* interconnected through mostly wireless equipment that users, companies, administrations must 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.

We have collected network description data through CNML files (obtained January 2016)[3]. CNML (Community Networks Markup Language) is an XML-based language used to describe community networks. 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 community network 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.

The CNML information obtained 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 terminal nodes of the base graph (i.e., client nodes). Table 1 summarizes

[3] https://guifi.net/en/guifi/cnml/2413.

Table 1. Summary of the used network graphs

	Nodes / edges	Node degree max/ mean/ min	Diameter	Zones
Base-graph	**13636 / 13940**	537 / 2.04 / 1	35	129
Core-graph	**687 / 991**	20 / 2.88 / 1	32	85

the main properties of base and core graphs that we use in our study e.g., number of nodes, node degree, diameter (number of max hops in the sub-graph) and number of zones traversed in core and base-graph.

2.2 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 (with different roles or not) that interact or exchange information through the community network. The service can also be a composite service (e.g., three-tier service) built from simpler 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 should be at infrastructure level (IaaS), as cloud services in current dedicated datacenters. Therefore the services are deployed directly over the core resources of the network (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 community network 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.

2.3 Service Quality Parameters

Resource dispersion in a community network scenario can be a drawback or an advantage, as the Nebula [4] authors claim in their research. The overlay created by composite services abstracts from actual underlying network connections. Based on that, services that require intensive inter-component communication (e.g. streaming service), can perform better if the replicas (service components) are placed close to each other in high capacity links [2]. 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 (e.g. overall reduction of bandwidth for service provisioning).

If we have some information about the application SLAs in community networks and node behaviour from the underlying network, decisions can be made accordingly, in order to promote that certain types of applications are executed in certain type of nodes with better QoS.

Our algorithm considers the following network and graph metrics as shown in Table 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 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 (%).
- **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 CNMCs. It is measured in milliseconds (ms).

Table 2. Service-specific quality parameters

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

– **Closeness:** The closeness is defined as the average distance (number of hops) from the solution obtained from 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.

3 Service Placement Algorithm

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. Algorithm 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 2. In the case of equal diameter allocations, the mean out-degree (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. The node with high degree centrality is initially chosen as 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 (e.g., latency or availability 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.

Algorithm 1. Policy-Aware Service Placement (PASP)

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

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

4 Experimental Results

4.1 Network Behaviour and Algorithmic Performance

Our service placement algorithm proposed in Sect. 3 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 CNMCs.

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 3 shows that 40 % of the base-graph nodes are reachable from the network 90 % or less of the time. The situation seems to be even worse with the core-graph nodes, which are supposed to be the most stable part of the network (20 % of the core-graphs have availability of 90 % or less). Base-graph nodes have higher availability because they are closer to users, and is of high interest to users to take care of them. It is interesting to note that 20 % of the core-graph nodes have availability between 98–100%, and those are most probably the nodes that comprise the backbone of the network and connect different administrative zones. Since the service placement is done on the core-graph nodes, selecting the nodes with higher availability (e.g., 90–100%) is of high importance.

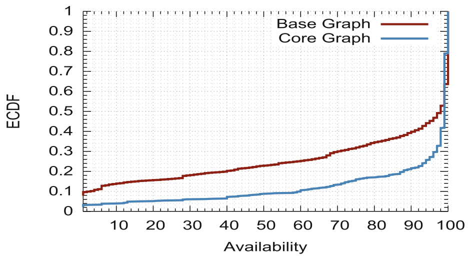

Fig. 3. ECDF of node availability

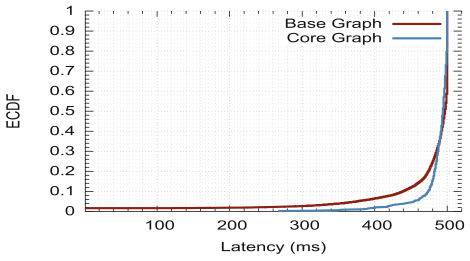

Fig. 4. ECDF of node latency

Figure 4 depicts the Empirical Cumulative Distribution Function (ECDF) plot of the node latency. Similar to the availability case, the latency of base-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 CNMCs. By not taking the performance of 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 (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. Figures 5 and 6 reveal that nodes obtained in the solutions with *PASP* have higher average availability and lower latency than with *Baseline*, with minimum service overlay diameter.

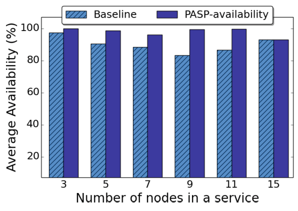

Fig. 5. PASP-availability

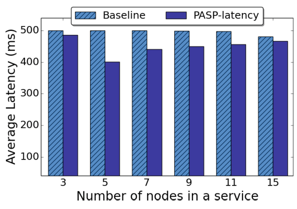

Fig. 6. PASP-latency

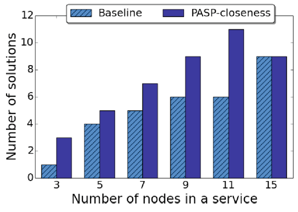

Fig. 7. PASP-closeness

On average, the gain of $PASP$ over $Baseline$ is 8 % for the availability, and 45 ms for the latency (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 or might not be very close (in terms of number of hops) to base-graph clients. Because of this we also developed another flavour of PASP algorithm called $PASP - closeness$. Figure 7 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).

Moreover, from working with the Guifi.net data, we observed some patterns in the node features that conforms optimal allocations. 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 (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 (for bandwidth-intensive services) and betweenness centrality (for latency-sensitive services) is specially useful to obtain more optimal overlays.

4.2 Deployment in a Real Production Community Network

In order to understand the gains of our network-aware service placement algorithm in a real production community network, we deploy our algorithm in real hardware connected to the nodes of the QMP[4] network, which is a subset of Guifi.net located in the city of Barcelona. Figure 8 depicts the topoloqy of the QMP network. Furthermore, a live QMP monitoring page updated hourly is available in the Internet[5].

We use 16 servers connected to the wireless nodes of QMP. 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[6] operating system. Cloudy contains some pre-integrated distributed applications, which the community network user can

[4] http://qmp.cat/.
[5] http://dsg.ac.upc.edu/qmpsu/index.php.
[6] http://cloudy.community/.

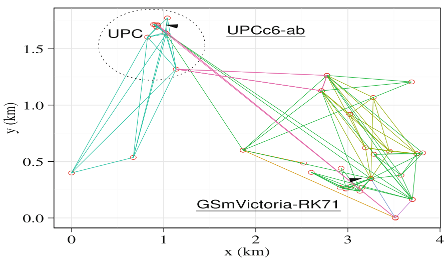

Fig. 8. QMP topology

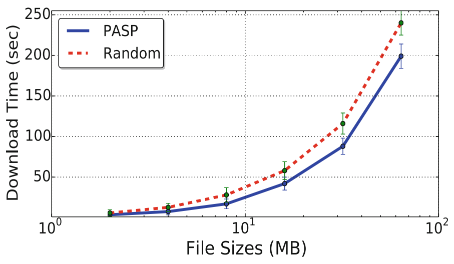

Fig. 9. Average reading time on clients side

activate to enable services inside the network. Services include a streaming service, a storage service and a folder synchronizing service, among others. For our experiments, we use the storage service, which is based on Tahoe-LAFS[7]. Tahoe-LAFS is an open-source distributed storage system with enforced security and fault-tolerance features, such as data encryption at the client side, coded transmission and data dispersion among a set of storage nodes.

As the controller node we leverage the experimental infrastructure of Community-Lab[8]. Community-Lab provides a central coordination entity that has knowledge about the network topology in real time. Out of the 16 devices used, three of them are storage nodes and 13 of them are clients (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 three 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 9 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. The experiment is composed of 20 runs, where each run has 10 repetitions, and averaged over all the successful runs. Standard deviation error bars are also shown.

Regarding the network interferences that may be caused by other users concurrent activities which can impact the results of our experiments, we reference to our earlier work [5] which investigated these issues.

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 s for the clients on average to read a 8 MB file. In the random case, the time is almost doubled, reaching 28 s for reading a file from the clients side. We observed therefore that when allocating services, taking into account the

[7] https://tahoe-lafs.org/trac/tahoe-lafs.
[8] https://community-lab.net/.

closeness and availability parameters in the allocation decision, on average (for all clients) our algorithm reduces the client reading times for 16 %. Maximum improvement (around 31 %) has been achieved when reading larger files (64 MB). When reading larger files client needs to contact many nodes in order to complete the reading of the file.

5 Related Work

Service placement is a key function of cloud management systems. Typically, it is responsible for monitoring all the physical and virtual resources on a system and balance their load through the allocation, migration and replication of tasks.

Data Centers: Choreo [6] 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 [7] the authors proposed an optimal allocation solution for ambient intelligence environments using tasks replication to avoid network performance degradation. Volley [8] is a system that performs automatic data placement across geographically distributed datacenters of Microsoft. 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.

Distributed Clouds: There are few works that provides service placement in distributed clouds with network-aware capabilities. The work in [9] proposes efficient algorithms for the placement of services in distributed cloud environment. Their algorithms need input on the status of the network, computational resources and data resources which are matched to application requirements. In [10] authors propose a selection algorithm to allocate resources for service-oriented applications and the work in [11] focuses on resource allocation in distributed small datacenters.

Service Migration: Regarding the service migration in distributed clouds, few works came out recently. The authors in [12,13] 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 [14] studies when services should be migrated in response to user mobility and demand variation.

Our focus is to perform service placements in community network clouds, which are peer-to-peer clouds formed from low-resource machines and very dynamic and diverse network. Another work in the community network context related to ours is [15] where the authors propose service allocation algorithms that minimize the coordination and overlay cost along the network.

6 Conclusion and Future Work

We addressed the need for network-aware service placement in community network micro-cloud infrastructures. We looked at a specific case of improving the deployment of service instance on micro-servers for enabling an improved distributed storage service in a community network.

As services become more network intensive, the bandwidth, latency etc., between the used nodes becomes the bottleneck for improving performance. In community networks, the limited capacity of nodes and links and an unpredictable network performance becomes a problem for service performance. Network awareness in placing services allows to chose more reliable and faster paths over poorer ones.

In this work we introduced a service placement algorithm that provides improved overlay service selection for distributed services considering service quality parameters, without the need for exploring the whole solution space. 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, a production community network, and quantified the performance and effects of our algorithm. We conducted our study on the case of a distributed storage service. In experiments we showed that by using our service placement algorithm, we were able to improve the total file reading time comparing to the currently used random placement.

In next steps we plan to develop and implement a decentralized version of our investigated service placement algorithm. Service migration should also be addressed to support performance objectives in the case of user mobility and within dynamic changes in the network.

Acknowledgements. This work was supported by the EU Horizon 2020 Framework Program project netCommons (H2020-688768), by the EMJD-DC program and by the Spanish Government under contract TIN2013-47245-C2-1-R. This work was also supported by the national funds through Fundação para a Ciência e a Tecnologia with reference UID/CEC/50021/2013.

References

1. Selimi, M., et al.: Cloud services in the Guifi.net community network. Comput. Netw. **93**, 373–388 (2015). Part 2
2. Selimi, M., et al.: Integration of an assisted p2p live streaming service in community network clouds. In: Proceedings of the IEEE 7th International Conference on Cloud Computing Technology and Science, CloudCom 2015. IEEE, November 2015
3. Selimi, M., et al.: Performance evaluation of a distributed storage service in community network clouds. Concurrency and Computation: Practice and Experience n/a-n/a cpe.3658 (2015)
4. Ryden, M., et al.: Nebula: distributed edge cloud for data intensive computing. In: IEEE International Conference on Cloud Engineering, IC2E 2014, pp. 57–66, March 2014

5. Cerdà-Alabern, L., Neumann, A., Escrich, P.: 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 2013, pp. 23–30. ACM, New York (2013)
6. LaCurts, K., Deng, S., Goyal, A.: Choreo: network-aware task placement for cloud applications. In: Proceedings of the 2013 Conference on Internet Measurement-Conference, IMC 2013, pp. 191–204. ACM, New York (2013)
7. Herrmann, K.: Self-organized service placement in ambient intelligence environments. ACM Trans. Auton. Adapt. Syst. **5**(2), 6:1–6:39 (2010)
8. Agarwal, S., 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 2010, Berkeley, CA, USA, USENIX Association 2–2 (2010)
9. Steiner, M., Gaglianello, B.G., Gurbani, V., Hilt, V., Roome, W. D., Scharf, M., Voith, T.: Network-aware service placement in a distributed cloud environment. In: Proceedings of the ACM SIGCOMM 2012 Conference, SIGCOMM 2012, pp. 73–74. ACM, NewYork (2012)
10. Klein, A., Ishikawa, F., Honiden, S.: Towards network-aware service composition in the cloud. In: Proceedings of the 21st International Conference on World Wide Web, WWW 2012, pp. 959–968. ACM, New York (2012)
11. Alicherry, M., Lakshman, T.: Network aware resource allocation in distributed clouds. In: 2012 Proceedings of INFOCOM, pp. 963–971. IEEE, March 2012
12. Wang, S., Urgaonkar, R., Zafer, M., He, T., Chan, K., Leung, K.K.: Dynamic service migration in mobile edge-clouds. CoRR abs/1506.05261 (2015)
13. Wang, S., et al.: Dynamic service placement for mobile micro-clouds with predicted future costs. In: IEEE International Conference on Communications, ICC 2015, pp. 5504–5510, June 2015
14. Urgaonkar, R., et al.: Dynamic service migration and workload scheduling in edge-clouds. Perform. Eval. **91**, 205–228 (2015). Special Issue: Performance 2015
15. Vega, D., Meseguer, R., Cabrera, G., Marques, J.: Exploring local service allocation in community networks. In: 10th International Conference on Wireless and Mobile Computing, Networking and Communications, WiMob 2014, pp. 273–280. IEEE, October 2014