2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing

Practical Service Placement Approach for Microservices Architecture

Mennan Selimi*‡, Llorenç Cerdà-Alabern*, Marc Sánchez-Artigas†, Felix Freitag*, Luís Veiga ‡
* Universitat Politècnica de Catalunya, BarcelonaTech, Barcelona, Spain
{mselimi, llorenc, felix}@ac.upc.edu
† Universitat Rovira i Virgili, Tarragona, Spain
{marc.sanchez}@urv.cat
‡ Instituto Superior Técnico (IST), INESC-ID Lisboa, Lisbon, Portugal
{luis.veiga}@inesc-id.pt

***Abstract*—Community networks (CNs) have gained momentum in the last few years with the increasing number of spontaneously deployed WiFi hotspots and home networks. These networks, owned and managed by volunteers, offer various services to their members and to the public. To reduce the complexity of service deployment, community *micro-clouds* have recently emerged as a promising enabler for the delivery of cloud services to community users. By putting services closer to consumers, 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 micro-clouds and services over the network.**

To achieve this, this paper proposes 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. To evaluate its performance, we compare our heuristic with one based on random placement in `Guifi.net`, the biggest CN worldwide. Our experimental results show that our heuristic consistently outperforms random placement by 211% in terms of bandwidth gain. We quantify the benefits of our heuristic on a real live video-streaming service, and demonstrate that video chunk losses decrease significantly, attaining a 37% decrease in the loss packet rate. Further, using a popular Web 2.0 service, we demonstrate that the client response times decrease up to an order of magnitude when using our heuristic.

***Index Terms*—service placement; community networks; micro-clouds;**

I. Introduction

Since early 2000s, community networks (CNs) or "*Do-It-Yourself*" networks have gained momentum in response to the growing demands 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.

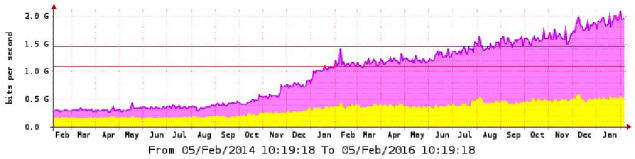

Figure 1. `Guifi.net` inbound and outbound traffic (Feb 2014 - Feb 2016).

These challenges have not precluded CNs from flourishing around. For instance, `Guifi.net`[1], 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, and until today, it has grown into a network of more than $32,000$ operational nodes. This makes it the largest CN worldwide [1]. Just to give some numbers, Figure 1 depicts the evolution of the total inbound (pink) and outbound (yellow) traffic to the Internet for the last two years. A mere inspection of this figure tells us that `Guifi.net` traffic has tripled. Traffic peaks correspond to the arrival of new users and deployment of bandwidth-hungry services in the network. Actually, a significant number of services, including GuifiTV, Graph servers, mail and game services, are running within `Guifi.net`. All these services have been provided by individuals, social groups, and small non-profit or commercial service providers.

`Guifi.net` ultimate aim is to create a full digital ecosystem that covers a highly localized area. But this mission is not so simple, because a quick glance at the type of services that users demand reveals that the percentage of Internet services (proxies and tunnel-based) is higher than 50%. This confirms that `Guifi.net` users are typically interested in mainstream Internet services [2], which imposes a heavy burden on the "thin" backbone links, with users experiencing high service variability.

Among other issues, this question spurred the invention of "alternative" service deployment models to cater for users in `Guifi.net`. One of these models was that based on *micro-clouds*. A micro-cloud is nothing but a platform to deliver services to a local community of citizens within the vast CN.

[1] http://guifi.net/

978-1-5090-6611-7/17 $31.00 © 2017 IEEE
DOI 10.1109/CCGRID.2017.28

Services can be of any type, ranging from personal storage to video streaming and P2P-TV [3]. Observe that this model is different from Fog computing, which extends cloud computing by introducing an intermediate layer between devices and datacenters. Micro-clouds take the opposite track, by putting services closer to consumers, so that no further or minimal action takes place in 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.

This approach, however, poses new challenges, such as that of *the optimal placement of micro-clouds within the CN* to overcome suboptimal performance. And `Guifi.net` is not an exception. 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 paper tries to answer the following two research questions:

1) First, given that sufficient state information is in place, is network-aware placement enough to deliver satisfactory performance to CN users?
2) Second, can the redundant placement of services further improve performance?

To answer these questions, we contribute in this work a new placement heuristic called *BASP* (Bandwidth and Availability-aware Service Placement), 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 order of hours to complete, BASP runs much faster; it just takes a few seconds, while achieving equally good results.

Our results show that *BASP* consistently outperforms random placement, the existing in-place and naturally fast strategy in `Guifi.net`, by 211% with respect to end-to-end bandwidth. 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 according to BASP. Our experimental results demonstrate that with BASP, the video chunk loss in the peer side decreased up to a 3% point reduction, i.e., worth a 37% reduction in the loss packet rate, which is a significant improvement. Furthermore, when using the BASP with the Web 2.0 service (i.e., social networking service), the client response times decreased up to an order of magnitude.

The rest of the paper is organized as follows. In Section II we describe and characterize the performance of the QMP network. Section III defines our system model and presents our *BASP* heuristic. In Section IV we discuss the evaluation results. In Section V we present and discuss the real deployment experiments with a video-streaming and Web 2.0 service. Section VI describes related work and section VII concludes and discusses future research directions.

II. 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 (Quick Mesh Project) network over a five-month period. Our goal is to determine the key features of the network (e.g. bandwidth distribution) and its nodes (e.g. availability patterns) that could help us to design new heuristics for intelligent service placement in CNs.

Figure 2. QMP outdoor devices

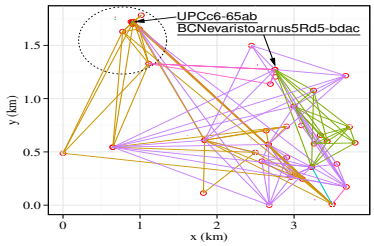

Figure 3. QMP network topology

A. QMP Network: A Brief Background

QMP network, began deployment in 2009 in a quarter of the city of Barcelona, Spain, called Sants, as part of the *Quick Mesh Project* (QMP)[2]. QMP is an urban mesh network and it is a subset of the `Guifi.net` CN sometimes called GuifiSants. At the time of writing, QMP has around 71 nodes. There are two gateways (proxies) distributed in the network that connect QMP to the rest of Guifi.net and Internet (see Figure 3). A detailed description of QMP can be found in [4].

Typically, QMP users have an outdoor router (OR) with a Wi-fi interface on the roof, connected through Ethernet to an indoor AP (access point) as a premises network. The most common OR in QMP is the NanoStation M5 as shown in Figure 2, 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 (NanoBridges). ORs in QMP are flashed with the Linux distribution which was developed inside the QMP project which is a branch of OpenWRT[3] and uses BMX6 as the mesh routing protocol [5].

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 operating system, which allows running services in a Docker containers.

Methodology and data collection: Measurements have been obtained by connecting via SSH to each QMP OR 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 have been taken hourly over a five-month period, starting from July 2016 to November 2016, and our live monitoring page and data is publicly available in the Internet[4]. We use this data to analyse main aspects of QMP network.

[2]http://qmp.cat/Home
[3]https://openwrt.org/
[4]http://dsg.ac.upc.edu/qmpsu/index.php

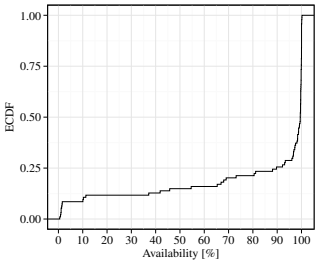

Figure 4. QMP node availability

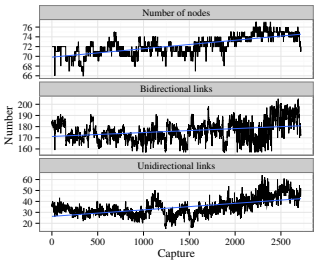

Figure 5. Number of nodes and links

B. 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 4 shows the Empirical Cumulative Distribution Function (ECDF) of the 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 (we took 2718 captures in total). Figure 4 reveals 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 [6], 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 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 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 micro-clouds.

C. Bandwidth characterization

A significant amount of services that run on QMP and `Guifi.net` network are network-intensive (bandwidth and delay sensitive), transferring large amounts of data between the network nodes [2]. 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 6 shows the average bandwidth distribution of all the links. The figure shows that the link throughput can be fitted with a mean of 21.8 Mbps. At the same time Figure 6 reveals that the 60% of the nodes have 10 Mbps or less throughput. 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 7 shows the bandwidth averages in both directions of the 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 (not shown in the graphs). For instance, node GSgranVia255, around 6am, when the user traffic is the lowest and equal in both directions, the asymmetry of the links bandwidth observed in Figure 7 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 present at each end, or different transmission powers.

In order to measure the link asymmetry, Figure 8 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 (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.

D. Discussion

Here are some observations (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 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

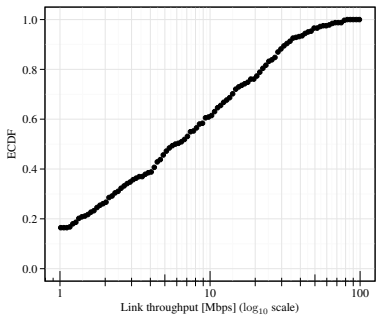

Figure 6. Bandwidth distribution

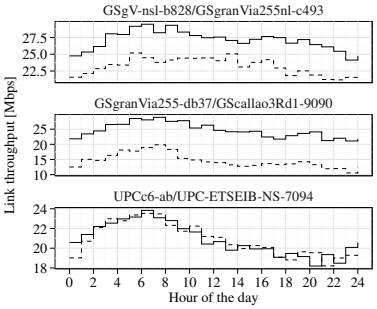

Figure 7. Bandwidth in the three busiest links

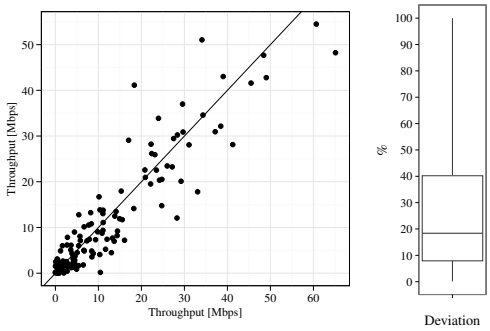

Figure 8. Bandwidth asymmetry

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 (25% of the links have a deviation higher than 40%). We observed a highly skewed traffic pattern and highly skewed bandwidth distribution (Figure 6).

Currently used organic (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 (service components) are placed close to each other in high capacity links [3]. 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) [7].

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.

III. 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 I.

A. Network Graph

The deployment and sharing of services in CNs is made available through *community network micro-clouds* (CNMCs). The idea of CNMC is to place the cloud at the edge closer to community end-users, so users can have fast and reliable access to the service. To reach its full potential, a CNMC needs to be carefully deployed in order to utilize the available bandwidth resources.

In a CNMC, a server or low-power device (i.e, home gateway) is directly connected to the wireless base-station (ORs) providing cloud services to users that are either within a reasonable distance or directly connected to base-station.

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.

B. Service Graph

The services aimed in this work are at infrastructure level (IaaS), as cloud services in current dedicated datacenters. 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[5]). The distributed services can be composite services (non-monolithic) built from simpler parts, e.g., video streaming (built from the source and peers component), web service (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.

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 (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]http://microservices.io/patterns/microservices.html

C. 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, how to place a service on a network as to maximize user 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:

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

with respect to the following constraints:

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

$$\forall e \in E : \sum_{s1,s_2 \in S} X_{s1,s2}(e) \times \beta_{s1,s2} \leq B_e \tag{2}$$

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

$$\forall n \in N : \sum_{n \in N} R_n \geq \lambda \tag{3}$$

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

$$|D| = k \tag{4}$$

D. Proposed Algorithm: BASP

Solving the problem stated in Equation 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* [8] 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.

Table I
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

Algorithm 1 B A S P

Require: $G(N,E)$ ▷ 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

```
procedure PERFORMKMEANS(G,k)
    if R_n ≥ λ then
        return C
    end if
end procedure
procedure FINDCLUSTERHEADS(C)
    clusterHeads ← list()
    for all k ∈ C do
        for all i ∈ C_k do
            B_i ← 0
            for all j ∈ setdiff(C,i) do
                B_i ← B_i + estimate.route.bandw(G,i,j)
            end for
            clusterHeads ← max B_i
        end for
    end for
    return clusterHeads
end procedure
procedure RECOMPUTECLUSTERS(clusterHeads,G)
    C' ← list()
    for all i ∈ clusterHeads do
        cluster_i ← list()
        for all j ∈ setdiff(G,i) do
            B_j ← estimate.route.bandw(G,j,i)
            if B_j is best from other nodes i then
                cluster_i ← j
            end if
            C' ← cluster_i
        end for
    end for
    return C'
end procedure
```

Our BASP algorithm (see Algorithm 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 (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 (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. [9].

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 9.
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 9.

Complexity:

The complexity of the *BASP* is as follows: for *BASP*, finding the optimal solution to the K-means (i.e., phase one) clustering problem if k and d (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.

IV. Evaluation

A. Setup

We take a network snapshot (capture) from 71 physical nodes of the QMP network regarding the bandwidth of the links[6] and node availability. The node and bandwidth 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. Additionally, we have discarded some disconnected clusters. The links are bidirectional and unidirectional, thus we we use a directed graph. The nodes of QMP consists of Intel Atom N2600 CPU, 4GB 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.

B. Comparison

To emphasise the importance of the different phases of Algorithm 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 1. The service is placed on the node having the maximum bandwidth on the initial clusters formed by 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 1. The service is placed on the node having the maximum bandwidth after the clusters are re-computed.

C. Results

Figure 9 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 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 (*Naive K-Means*) to 27.7 Mbps (*BASP*), which represents a 50% improvement. The biggest 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 9, 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 9, 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 for all the combinations of size k from a set of size n. This is a combinatorial problem that becomes intractable even for small

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

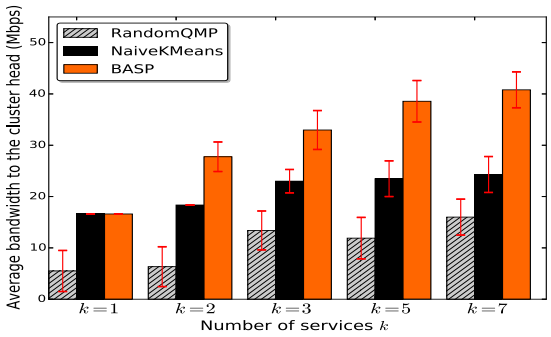

Figure 9. Average bandwidth to the cluster heads

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 (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 II 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 II shows some centrality measures and some graph properties obtained for each cluster head. Further, Figure 10 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 (for $k=2$ 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 paper and will be reserved as our future work.

V. Experimental Evaluation

A. Cloudy: A Service Hub for the Micro-Clouds

In order to foster the adoption and transition of the community micro-cloud environment, we provide a community cloud distribution, codenamed Cloudy[7]. This distribution contains the platform and application services of the community cloud system. 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. Cloudy is open-source and can be downloaded from public repositories[8].

[7]http://cloudy.community/
[8]http://repo.clommunity-project.eu/images/

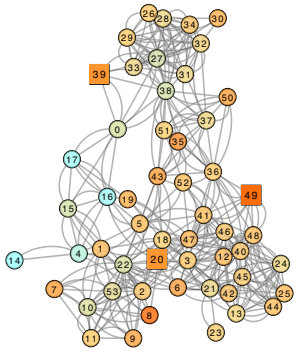

Figure 10. Neighborhood connectivity graph of the QMP network

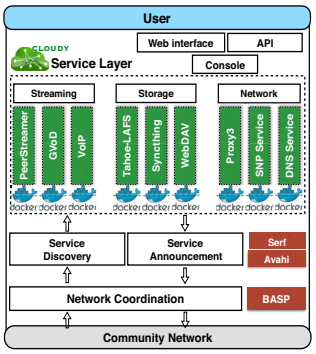

Figure 11. Cloudy architecture

Cloudy's main components can be considered a layered stack, with services residing both inside the kernel and at the user level. Figure 11 reports some of the available services running on Docker containers. 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 as shown in Figure 11. Thus, the service can be discovered by the other users.

B. Evaluation 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, located in the city of Barcelona. We concentrate on benchmarking two of the most popular network-intensive applications: *Live-video streaming service*, and *Web 2.0 Service* performed by the most popular websites.

1) Live-video streaming service: PeerStreamer[9], 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 (by default, this is 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 (as home gateways, set-top-boxes, etc.) or within other infrastructures distributed around the city of Barcelona. They run the Cloudy operating system. As the controller node, we leverage the experimental infrastructure of Community-Lab[10]. Community-Lab provides a central coordination entity that has knowledge about the network topology in real time and

[9]http://peerstreamer.org/
[10]https://community-lab.net/

Table II
CENTRALITY MEASURES FOR THE 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
Random QMP - 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 [seconds]	7 sec	15 sec		23 sec			30 sec				

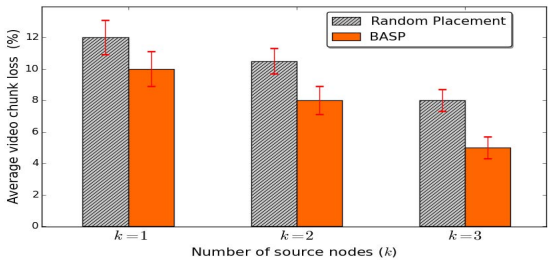

Figure 12. Average video chunk loss in QMP

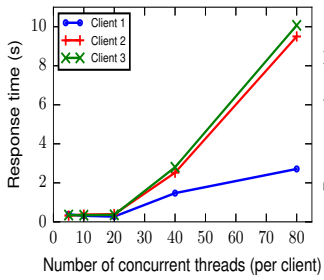

Figure 13. UpdateActivity-Random

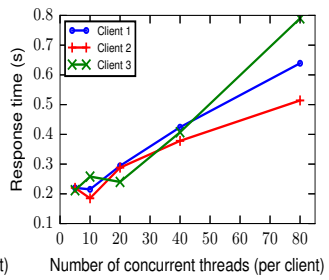

Figure 14. UpdateActivity-BASP

allows researchers to deploy experimental services and perform experiments in a production CN. The nodes of QMP that are running the live video streaming service are part of CommunityLab. In our experiments, we connect a live streaming camera (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 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 12 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 Figure 12, 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.

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 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 (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 [10]. 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[11] 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 (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 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 (marked as F in Table III). The location of the web server,

[11]https://elgg.org/

Table III
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	0.02s	0.03s	0.01	0.01	0.02	0.02	0.01s	0.03s
Improvement	0.1s	0.2s	1.8s	6.7s	0.1s	0.1s	1.2s	4.2s

database server and memcached server has a direct impact on the client response time.

Results: Figure 13 and Figure 14 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 13 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 (300-350 ms). Response time increases more than one order of magnitude in Client 2 and Client 3, and an order of magnitude in Client 1 when performing 160 operations (i.e., 80 threads).

Figure 14 depicts that, the client response times for higher workloads *decreases an order of magnitude* when using our *BASP* heuristic compared to *Random* approach shown in the Figure 13. For up to 120 operations per client, the response times that three clients perceive is slightly better (200-280 ms) than the response time when the web server is deployed with the *Random* approach. Furthermore, Table III 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 when using *BASP* heuristic over *Random* approach. We can notice that the gain brought by the *BASP* heuristic is higher for more intensive workloads.

VI. RELATED WORK

Service placement is a key function of the 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.

Data centers: Choreo [11] 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. Volley [12] 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. A large body of work of service placement in data centres has been devoted to finding heuristic solutions [13].

Most of the work in the data center 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 network makes the service placement a very different problem than in a mostly homogeneous cloud datacenter. Our measurement results demonstrate that 25% of the links have a symmetry deviation higher than 40%.

Distributed Clouds: 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). The work in [14] 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 [15] authors propose a selection algorithm to allocate resources for service-oriented applications and the work in [16] focuses on resource allocation in distributed small datacenters. Another example of a network-aware approach is the work from Moens in [17] which employs a Service Oriented Architecture (SOA), where applications are constructed as a collection of services. Their approach performs node and link mapping simultaneously. The work in [18] extends the work of Moens in wireless settings taking into account the IoT. Mycocloud [19] is another work, which provides elasticity through self-organized service placement in decentralized clouds. The work of Elmroth [20] takes into account rapid user mobility and resource cost when placing applications in Mobile Cloud Networks (MCN). A recent work of Tantawi [21] uses biased statistical sampling methods for cloud workload placement. Regarding the service placement through migration, the authors in [22] and [23] 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

Most of the work in the distributed clouds consider micro-datacenters, where in our case the CN micro-clouds consist of constraint/low-power devices such us home gateways. Furthermore, in our case we have a partial information regarding the computational devices, so their approaches are not fully applicable to our environment.

Wireless Environment: In [24] the authors propose an optimal allocation solution for ambient intelligence environments using tasks replication to avoid network performance degradation. Some other works done in wireless settings are the work of Davide [25] and our recent work [7] which proposes several placement algorithms that minimize the coordination and overlay cost along a CN. The focus of the work in this paper is to design a low-complexity service placement heuristic for CN micro-clouds in order to maximise bandwidth and improve user QoS and QoE.

VII. Conclusion

In this paper, 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 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 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 a CN micro-clouds. 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%. 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.

As a future work, we plan to look into service migration, i.e., the controller needs to decide which micro-cloud should perform the computation for a particular user, with the presence of user mobility and other dynamic changes in the network.

Acknowledgement

This work was supported by the European H2020 framework programme projects netCommons (H2020-688768) and LightKone (H2020-732505), and by the Spanish government under contracts TIN2013-47245-C2-1-R and TIN2016-77836-C2-2-R. This work was also supported by the national funds through Fundação para a Ciência e a Tecnologia in project ContexTWA with reference PTDC/EEI-SCR/6945/2014. The authors would like to thank Besim Bilalli (UPC) and Rana Faisal Munir (TU Dresden) for their helpful and constructive comments.

References

[1] D. Vega, R. Baig, L. Cerdà-Alabern, E. Medina, R. Meseguer, and L. Navarro, "A technological overview of the guifi.net community network," *Computer Networks*, vol. 93, Part 2, pp. 260 – 278, 2015.

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

[3] M. Selimi *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, Nov. 2015.

[4] L. Cerdà-Alabern, A. Neumann, and P. 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*, ser. MSWiM '13. New York, NY, USA: ACM, 2013, pp. 23–30.

[5] 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.

[6] 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.

[7] M. Selimi, D. Vega, F. Freitag, and L. Veiga, "Towards network-aware service placement in community network micro-clouds," in *22nd International Conference on Parallel and Distributed Computing (Euro-Par 2016)*. Springer International Publishing, 2016, pp. 376–388.

[8] "Stirling Number of the Second Kind," http://mathworld.wolfram.com/StirlingNumberoftheSecondKind.html.

[9] A. Lancichinetti and S. Fortunato, "Community detection algorithms: A comparative analysis," *Phys. Rev. E*, vol. 80, p. 056117, Nov 2009.

[10] T. Palit, Y. Shen, and M. Ferdman, "Demystifying cloud benchmarking," in *2016 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, April 2016, pp. 122–132.

[11] K. LaCurts *et al.*, "Choreo: Network-aware task placement for cloud applications," in *Proceedings of the 2013 Conference on Internet Measurement Conference*, ser. IMC '13. New York, NY, USA: ACM, 2013, pp. 191–204.

[12] S. 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*, ser. NSDI'10. Berkeley, CA, USA: USENIX Association, 2010, pp. 2–2.

[13] H. Ghanbari *et al.*, "Replica placement in cloud through simple stochastic model predictive control," in *2014 IEEE 7th International Conference on Cloud Computing*, June 2014, pp. 80–87.

[14] M. Steiner *et al.*, "Network-aware service placement in a distributed cloud environment," in *Proceedings of the ACM SIGCOMM 2012 Conference*, ser. SIGCOMM '12. New York, NY, USA: ACM, 2012, pp. 73–74.

[15] A. Klein, F. Ishikawa, and S. Honiden, "Towards network-aware service composition in the cloud," in *Proceedings of the 21st International Conference on World Wide Web*, ser. WWW '12. New York, NY, USA: ACM, 2012, pp. 959–968.

[16] M. Alicherry and T. Lakshman, "Network aware resource allocation in distributed clouds," in *Proceedings of INFOCOM, IEEE*, March 2012, pp. 963–971.

[17] H. Moens *et al.*, "Hierarchical network-aware placement of service oriented applications in clouds," in *2014 IEEE Network Operations and Management Symposium (NOMS)*, May 2014, pp. 1–8.

[18] 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.

[19] D. J. Dubois, G. Valetto, D. Lucia, and E. D. 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.

[20] W. Tärneberg, A. Mehta, E. Wadbro, J. Tordsson, J. Eker, M. Kihl, and E. Elmroth, "Dynamic application placement in the mobile cloud network," *Future Generation Computer Systems*, vol. 70, pp. 163 – 177, 2017.

[21] A. N. Tantawi, "Solution biasing for optimized cloud workload placement," in *2016 IEEE International Conference on Autonomic Computing (ICAC)*, July 2016, pp. 105–110.

[22] R. Urgaonkar, S. Wang, T. He, M. Zafer, K. Chan, and K. K. Leung, "Dynamic service migration and workload scheduling in edge-clouds," *Performance Evaluation*, vol. 91, pp. 205 – 228, 2015.

[23] S. 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.

[24] K. Herrmann, "Self-organized service placement in ambient intelligence environments," *ACM Trans. Auton. Adapt. Syst.*, vol. 5, no. 2, pp. 6:1–6:39, May 2010.

[25] D. Vega, R. Meseguer, G. Cabrera, and J. 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.