Bandwidth-aware Service Placement in Community Network Clouds

Mennan Selimi
UPC
Barcelona, Spain
mselimi@ac.upc.edu

Llorenç Cerdà-Alabern
UPC
Barcelona, Spain
llorenc@ac.upc.edu

Liang Wang
Cambridge University
Cambridge, UK
liang.wang@cl.cam.ac.uk

Arjuna Sathiaseelan
Cambridge University
Cambridge, UK
arjuna.sathiaseelan@cl.cam.ac.uk

Luís Veiga
INESC-ID / IST Lisboa
Lisbon, Portugal
luis.veiga@inesc-id.pt

Felix Freitag
UPC
Barcelona, Spain
felix@ac.upc.edu

***Abstract*—Seamless computing and service sharing in community networks (CNs) have gained momentum due to the emerging technology of community network micro-clouds (CNMCs). However, deploying and running services in CNMCs confront enormous challenges to cope with, such as the dynamic nature of micro-clouds, limited capacity of nodes and links, asymmetric quality of wireless links, geographic singularity based deployment model rather than network QoS based, etc. CNMCs have been increasingly used by network-intensive services which exchange significant amounts of data between nodes, therefore their performance heavily relies on the available bandwidth resource in a network. This paper proposes a novel bandwidth-aware service placement algorithm which aims to replace the current random placement adopted by Guifi.net. Our experimental results show that the proposed BASP algorithm consistently outperforms the random placement in Guifi.net by 35% regarding its bandwidth gain. More promisingly, as the number of services increases, the gain tends to increase accordingly. Furthermore, we quantify the performance and effects of our algorithm with a real service such as live video-streaming service, in a real production CN. Our real experimental results show that when using our BASP algorithm, the video chunk loss in the peer side decreases up to 3%.**

***Index Terms*—community network cloud; service placement;**

I. Introduction

Community networks (CNs) or Do-It-Yourself networks are built in a bottom-up and fully decentralized fashion, and are usually maintained by their own users. Early in the 2000s, community networks already gained momentum in response to the growing demands for network connectivity in rural and urban communities. One successful effort of such a network is Guifi.net[1], located in the Catalonia region of Spain. Guifi.net is defined as an open, free and neutral community network built by its members: citizens and organizations pool their resources and coordinate efforts to build and operate a local network infrastructure. Guifi.net was launched in 2004 and till today it has grown into a network of more than 30,000 operational nodes, which makes it the largest community network worldwide [1]. Figure 1 shows the evolution of total inbound and outbound Guifi.net traffic to the Internet for the last two years. Pink colour represents incoming traffic from Internet and yellow represents outgoing traffic. For two years, the traffic has tripled and peaks are as a result of a new users and bandwidth-hungry services in the network.

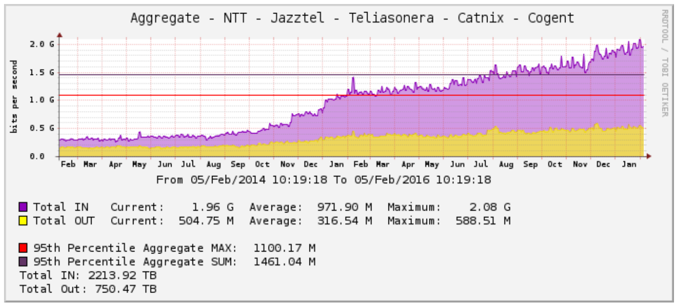

Figure 1. Guifi.net Traffic

Similar to other community networks, Guifi.net aims to create a highly localized digital ecosystem. However, the predominant usage we have observed, is to access cloud-based Internet services external to a community network. For instance, more than 50% of user-oriented services consumed in Guifi.net go through gateway proxies which provide Internet connectivity hence impose a heavy burden on the limit backbone links [2]. For a very long time in the past, user-oriented services had not been developed locally because of the lack of streamlined mechanisms to exploit all the available resources within a community network as well as other technological barriers. With the adoption of *community network micro-clouds*[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 homegrown services and using alternative open source software for many of today's Internet cloud services, e.g., data storage services, interactive applications such as Voice-over-IP (VoIP), video streaming, P2P-TV, and etc. In fact, a significant number of services were already locally deployed and run within Guifi.net including GuifiTV, Graph servers, mail servers, game servers [3]. All

[1]http://guifi.net/

[2]http://cloudy.community/

these services are provided by individuals, social groups, small non-profit or commercial service providers.

Because Guifi.net nodes are geographically distributed, given this set of local services, we need to decide where these services should be placed in a network. Obviously, without taking into account the underlying network resources, a service may suffer from poor performance, e.g, by sending large amounts of data across slow wireless links while faster and more reliable links remain underutilized. Therefore, the key challenge in community network micro-clouds is to determine the location of deployment, i.e. servers at certain geographic points in the network, with the different services multiplexed on a shared infrastructure. Although conceptually straightforward, it is challenging to calculate an optimal decision due to the dynamic nature of community networks and usage patterns. In this work we aim to address the following question: *"Given a community network cloud infrastructure, what is an effective and low-complexity service placement solution that maximises end-to-end performance (e.g., bandwidth)?"* Our preliminary results show that the proposed algorithm consistently outperforms the current random placement adopted in Guifi.net by 35% regarding its bandwidth gain. More promisingly, as the number of services increases, the gain tends to increase accordingly. Furthermore, 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 real service such as live video-streaming service. Our real experimental results show that when using our BASP algorithm, the video chunk loss in the peer side decreases up to 3%.

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

II. Need for Localized Services

In this section, we characterize wireless community networks by presenting our experimental measurements in a production example over five months, which exposes the necessity of deploying localized services [4] and justifies our motivation of proposing an intelligent placement algorithm.

A. QMP Network: A Brief Background

The network we consider, began deployment in 2009 in a quarter of the city of Barcelona, Spain, called Sants, as part of the *Quick Mesh Project*[3] (QMP). In 2012, nodes from *Universitat Politècnica de Catalunya* (UPC) joined the network, supported by the EU CONFINE [4] project. We shall refer to this network as *QMPSU* (from Quick Mesh Project at Sants-UPC). QMPSU is part of the Guifi community network which has more than 30.000 operational nodes. At the time of writing, QMPSU has around 61 nodes, 16 at UPC and 45 at Sants. There are two gateways, one in UPC Campus and another in Sants, that connect QMPSU to the rest of Guifi.net (see Figure 2). A detailed description of QMPSU can be found in [5], and a live monitoring page updated hourly is available in the Internet [5].

Typically, QMPSU 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 QMPSU is the NanoStation M5, which 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 QMPSU are flashed with the Linux distribution which was developed inside the QMP project which is a branch of OpenWRT[6] and uses BMX6 as the mesh routing protocol [6].

B. Characterization: Bandwidth-Hungry

In the following, we characterize the network performance of QMP network. 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 intelligent service placement in community networks [7]. Measurements have been obtained by connecting via SSH to each QMPSU 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 the last 5 months, starting from October 2015 to February 2016. We use this data to analyse main aspects of QMP network.

Figure 3 shows the node and link presence. We define the presence as the percentage a given node or link is observed over the captures. A capture is an hourly network snapshot that we take from the QMP network. Overall, 90 different nodes were detected. From those, only 61 were alive during the entire measurement period, leading to a presence higher than 95%. Around 30 nodes were missed in the majority of the captures (i.e., presence less than 10%). These are temporarily working nodes from other mesh networks and laboratory devices used for various experiments. Figure 3 also reveals that 56% of links used between nodes are unidirectional and others are bidirectional.

Figure 4, depicts the Empirical Cumulative Distribution Function (ECDF) of the average traffic sent in each of the links in the busy hour. The overall average traffic observed is 70 kbps. Figure 5 shows the average traffic in both directions of the three busiest links.

We characterize the wireless links of the QMP network by studying their throughput. Figure 6 shows the average throughput 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. In order to see the variability of the throughput, Figure 7 shows the throughput averages in both directions of the three busiest links (same links as in Figure

[3]http://qmp.cat/Home
[4]https://confine-project.eu/
[5]http://dsg.ac.upc.edu/qmpsu/index.php
[6]https://openwrt.org/

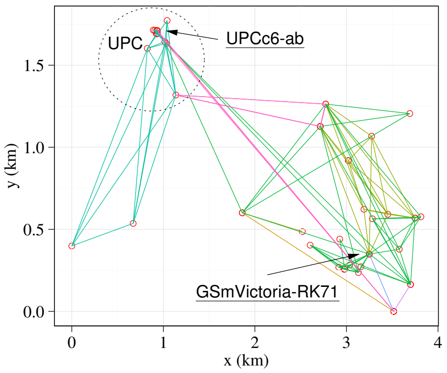

Figure 2. QMPSU network. Two main gateways are underlined.

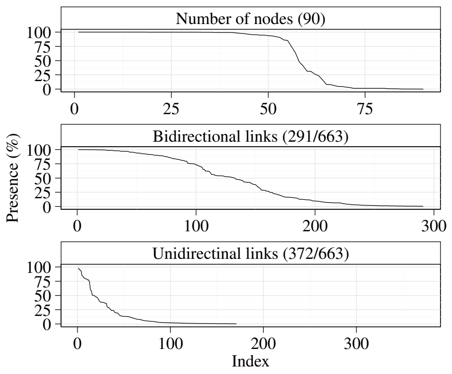

Figure 3. Node and link presence.

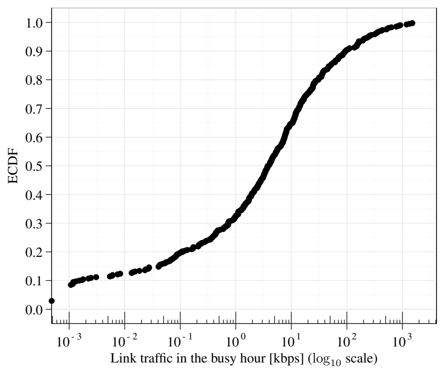

Figure 4. Link traffic in the busy hour ECDF.

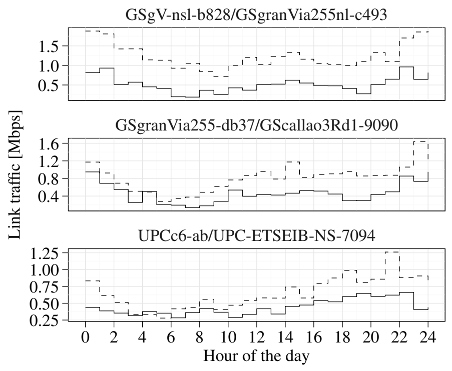

Figure 5. Traffic in the three busiest links.

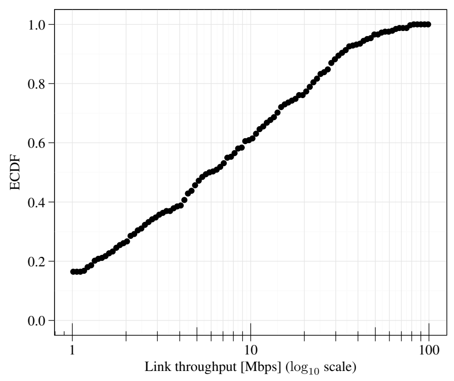

Figure 6. Throughput ECDF.

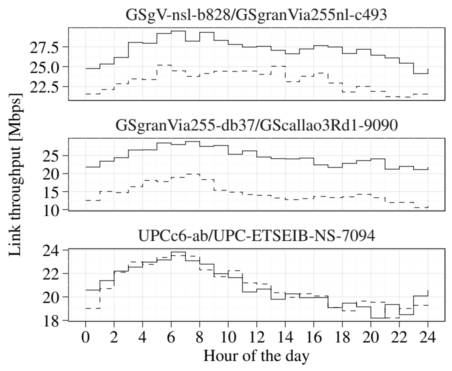

Figure 7. Throughput in the three busiest links.

5). The nodes of three busiest links are highlighted on the top of the figure. When we compare Figure 7 and Figure 5, we observe that the throughput is slightly affected by the traffic in the links. Solid and dashed lines are used to identify the measurements on each direction of the links. It is interesting to note that the asymmetry of the throughputs measured in both directions it not always due to the asymmetry of the user traffic. For instance (node GSgranVia255), around 6am, when the user traffic is the lowest and equal in both directions, the asymmetry of the links throughputs observed in Figure 4 remains the same. We thus conclude that this asymmetry must be due to the link characteristics, as level of interferences present at each end, or different transmission powers.

A significant amount of applications that run on Guifi.net and QMP network are network-intensive (bandwidth and delay sensitive), transferring large amounts of data between the network nodes [3]. The performance of such kind of applications depends not just on computational and disk resources but also on the network bandwidth between the nodes on which they are deployed. Therefore, the placement of such services in the network is of high importance. Here are some observations (features) that we captured from the measurements in QMP network:

- 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 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.
- The resources are not uniformly distributed in the network. Wireless links are with asymmetric quality for services (30% of the links have a deviation higher than 30%). We observed a highly skewed traffic pattern (Figure 4) and highly skewed bandwidth distribution (Figure 6).

Currently used *organic (random) placement scheme* in Guifi.net community network 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 [8]. 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). 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 community networks such as bandwidth.

III. Bandwidth-Aware Placement

The deployment and sharing of services in community networks is made available through *community network micro-clouds* (CNMCs). The idea of CNMC is to place the cloud 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.

A. Assumptions

In a CNMC, a server or low-power device is directly connected to the wireless base-station providing cloud services to users that are either within a reasonable distance or directly connected to base-station. These nodes are core-graph nodes what we call in Guifi.net. In Guifi.net core-graph is formed by removing the terminal nodes of the base graph (i.e. leaf nodes or clients).

It is important to remark that the services aimed in this work are at infrastructure level (IaaS), as cloud services in current dedicated datacenters (we assume QMP nodes are core-graph nodes). Therefore the services are deployed directly over the core resources of the network and accessed by base-graph clients. Services can be deployed by Guifi.net users or administrators.

The services we consider can be centralized or distributed. The distributed services can be composite services (non-monolithic) built from simpler parts, e.g., video streaming. These parts or components of service would create an overlay and interact with each other to offer more complex services. A service may or may not be tied to a specific node of the network. Each nodes can host one or more 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 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.

B. Formulation and Notations

We call the community network 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: $G \leftarrow (OR, L)$ where OR is the set of outdoor routers present in the CNs and L is the set of wireless links that connects them.

Let f_{ij} be the bandwidth of the path to go from node i to node j. We want a partition of k clusters: $S \leftarrow S_1, S_2, S_3, ..., S_k$ of the set of nodes in the mesh network. The cluster head i of cluster S_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:

$$\arg\max_S \sum_{i=1}^{k} \sum_{j \in Si} f_{ij} \tag{1}$$

Algorithm 1 Bandwidth-aware Service Placement (BASP)

```
Require: $G(V_n, E_n)$                                  ▷ Network graph
    $S \leftarrow S_1, S_2, S_3, ..., S_k$              ▷ $k$ partition of clusters
    $bw_i$                                              ▷ bandwidth of node $i$
 1: procedure PerformKmeans($G, k$)
 2:     return $S$
 3: end procedure
 4: procedure FindClusterHeads($S$)
 5:     $clusterHeads \leftarrow list()$
 6:     for all $k \in S$ do
 7:         for all $i \in S_k$ do
 8:             $bw_i \leftarrow 0$
 9:             for all $j \in setdiff(S, i)$ do
10:                 $bw_i \leftarrow bw + estimate.route.bandw(G, i, j)$
11:             end for
12:             $clusterHeads \leftarrow \max bw_i$
13:         end for
14:     end for
15:     return $clusterHeads$
16: end procedure
17: procedure RecomputeClusters($clusterHeads, G$)
18:     $S\prime \leftarrow list()$
19:     for all $i \in clusterHeads$ do
20:         $cluster_i \leftarrow list()$
21:         for all $j \in setdiff(G, i)$ do
22:             $bw_j \leftarrow estimate.route.bandw(G, j, i)$
23:             if $bw_j$ is best from other nodes $i$ then
24:                 $cluster_i \leftarrow j$
25:             end if
26:             $S\prime \leftarrow cluster_i$
27:         end for
28:     end for
29:     return $S\prime$
30: end procedure
```

C. Proposed Algorithm: BASP

We designed a bandwidth-aware algorithm that allocates services taking into account the bandwidth of the network. We take a network snapshot (capture) from QMP network regarding the bandwidth of the links [7]. Our bandwidth-aware service placement algorithm BASP (see Algorithm 1) runs in three phases.

(i) 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 locations 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. Grouping nodes based on geo-location is in line with how Guifi.net is organized. The nodes

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

in Guifi.net are organized into a tree hierarchy of *zones* [9]. 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.

(ii) The second phase of the algorithm is based on the concept of finding the cluster head maximizing the bandwidth between the head and member nodes of the cluster, formed in the first phase of the algorithm. The cluster heads computed in this phase are the ones having the maximum bandwidth to the other nodes in the cluster S_k. The cluster heads are node candidates for service placement.

(iii) The third and last phase of the algorithm includes reassigning the nodes to the selected cluster heads having the maximum bandwidth.

Regarding computational complexity, the naive brute force method can be estimated by calculating the *Stirling number of the second kind* [10] which counts the number of ways to partition a set of n elements into k nonempty subsets, i.e., $\frac{1}{k!}\sum_{j=0}^{k}(-1)^{j-k}\binom{n}{k}j^n \Rightarrow \mathcal{O}(n^k k^n)$. However, for BASP, finding the optimal solution to the k-means clustering problem if k and d (the dimension) are fixed (e.g., in our case $n = 54$, 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 $\mathcal{O}(n^{2k+1}\log n)$, which is significantly smaller than the brute force method.

IV. Algorithmic Behaviour & Performance

Solving the problem stated in Equation 1 in brute force for any number of N and k is NP-hard. For this reason we came up with our heuristic. Initially we used k-means algorithm for a first selection of the clusters. Then, we limit the choice of the cluster heads to be inside the sets of clusters obtained using k-means. Inside these clusters we computed the cluster heads having the maximum bandwidth to the other nodes. To emphasise the importance of phase two and three, in this section we compare *BASP* to *Naive K-Means* which partitions the nodes into k groups such that the sum of squares from nodes to the assigned cluster heads (centroids) is minimized. At the minimum, all cluster heads in *Naive K-Means* are at the mean of their Voronoi sets (the set of nodes which are nearest to the cluster heads).

Our experiment is comprised of 5 runs and the presented results are averaged over all the successful runs. Each run consists of 15 repetitions. Figure 8 depicts the average bandwidth to the cluster heads obtained with *Naive K-Means* algorithm and our *BASP* algorithm. Figure reveals that for any number of k, our *BASP* algorithm outperforms the *Naive K-Means* algorithm. For k=2 the average bandwidth to the cluster head is increased from 18.3 Mbps (obtained with naive k-means) to 27.7 Mbps (obtained with our BASP algorithm) i.e., 40% increase. The biggest increase of 50% is when k=7. Based on the observations from the Figure 8, the gap between two algorithms is growing as k increases. K increases as network grows.

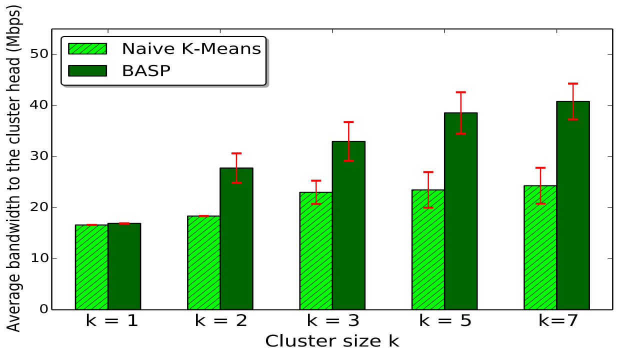

Figure 8. Average bandwidth to the cluster heads

Note that our heuristics enables us to select nodes (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 end up to be an NP-hard. Finding the solution is NP-hard, because finding the optimum 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 sizes of k or n (e.g., $k = 5$, $n = 54$). For instance, if we would like 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 the set of size 54. That is for 54 nodes we would end up having 25K combinations ($choose(54,3)$), or 25K possible nodes to start with. We managed to do this and the optimum average bandwidth obtained was 62.7 Mbps. The optimum bandwidth obtained for $k = 2$ was 49.1 Mbps, and for $k = 1$ was 16.9 Mbps. However the computation time took very long (65 hours for $k = 3$, 30 minutes for $k = 2$ etc.), comparing to BASP where it took 23 seconds for $k = 3$ and 15 seconds for $k = 2$. Table I shows the BASP improvement over Naive K-Means algorithm. Furthermore, Table I shows some centrality measures and some graph properties obtained for each cluster head. To summarize, BASP is able to achieve good bandwidth performance with very low computation complexity.

Correlation with centrality metrics: Figure 9 shows the neighborhood connectivity graph of the QMP network.The neighborhood connectivity of a node n is defined as the average connectivity of all neighbors of n. 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 the nodes with the highest neighborhood connectivity are the the cluster heads obtained with our BASP algorithm. 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. Cloudy: Integrating BASP with a Real Service Hub for Community Networks

In order to foster the adoption and transition of the community network cloud environment, we provide a community

Table I
CENTRALITY MEASURES FOR CLUSTER HEADS

	k=1	k=2		k=3			k=5				
Clusters [node id]	C1 [27]	C1 [20]	C2 [39]	C1 [20]	C2 [39]	C3 [49]	C1 [20]	C2 [4]	C3 [49]	C4 [51]	C5 [39]
Head degree	20	6	6	6	6	10	6	10	10	12	6
Neighborhood Connectivity	7.7	9.6	9.6	**9.6**	**9.6**	**10.8**	9.6	8.7	10.8	8.1	9.6
Diameter	6	5	3	4	3	5	4	2	3	1	3
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	7 sec	15 sec		23 sec			30 sec				

cloud GNU/Linux distribution, codenamed Cloudy [8]. 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. Each community cloud user who contributes infrastructure to the cloud is encouraged to install the Cloudy distribution on his/her on-premise device at home. Cloudy is open-source and can be downloaded from public repositories [9]. Cloudy is meant to be useful and usable for the end-user, to be installed on any kind of on-premise devices, which then can become part of community network cloud. Therefore, Cloudy has been installed on desktop PCs, but also on low-resource single-board-computers, such as RaspberryPI and Beagle Board Black4.

Cloudy's main components can be considered a layered stack, with services residing both inside the kernel and the user-level. Figure 10 indicates some of the already integrated types of services on the Cloudy CN distribution. An example of these services are the ones we consider in this paper, the video streaming service such as PeerStreamer.

Cloudy includes a tool for users, to announce and discover services in the CNMCs based on Serf, which is a decentralized solution for cluster membership and orchestration. On the network coordination layer the BASP, having knowledge about the underlying network topology, decides about the placement of the service announced via Serf (see Figure 10). After that, the service can be discovered by the other users.

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

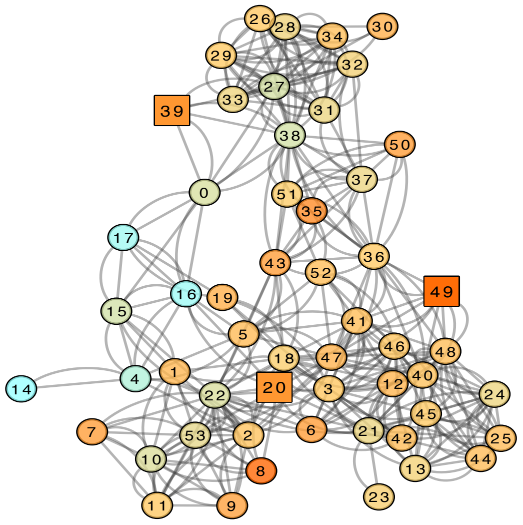

Figure 9. Neighborhood Connectivity in QMP network

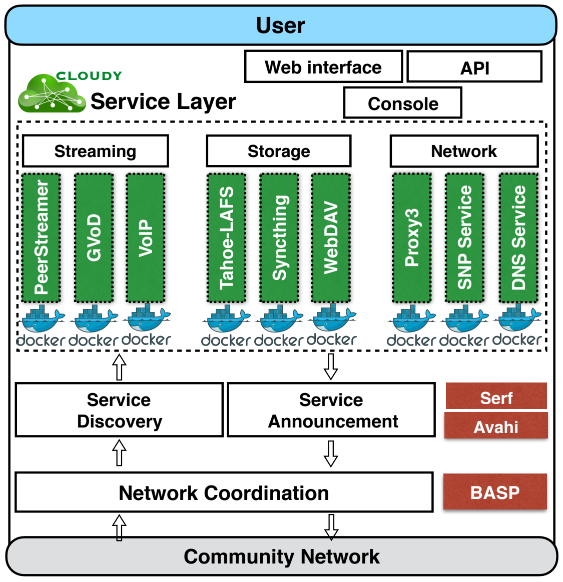

Figure 10. Cloudy architecture and its integration with BASP

A. UseCase: Live video-streaming service

PeerStreamer[10] is an open source live P2P video streaming service and mainly used in our Cloudy distribution as the live streaming service example. This service is built on a chunk based stream diffusion, where peers offer a selection of the chunks that they own to some peers in their neighborhood. The receiving peer acknowledges the chunks it is interested in, thus minimizing multiple transmissions of the same chunk to the same peer. Chunks consist of parts of the video to be streamed (by default, this is one frame of the video). PeerStreamer differentiates between a source node and peer node. Source node is responsible for sending the video chunk data to the peers in the network. In our case, both the source and peers are located in Docker containers in the nodes distributed over the QMP network [8].

B. Evaluation in 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 network, located in the city of Barcelona.

[10]http://peerstreamer.org/

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. Nodes are deployed to use the wireless links of each community network that operate in the ISM frequency bands at 2.4 GHz and 5 GHz. The hardware of the nodes 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 Cloudy, which allows running several services on one node simultaneously implemented as Docker or Linux containers (LXC). Containers of Cloudy contains some of pre-integrated distributed applications, which the community network user can activate to enable services inside the network. For our experiments, we use the live video streaming service, which is based on PeerStreamer.

As the controller node we leverage the experimental infrastructure of Community-Lab[11]. Community-Lab provides a central coordination entity that has knowledge about the network topology in real time and allows researchers to deploy experimental services and perform experiments in a real and production community network. The nodes of QMP where we are running the live video streaming service are part of Community-Lab.

In our experiments we connect a live streaming camera (maximum 512 kbps bitrate, 30 frame-per-second) to a local PeerStreamer instance which acts as the *source* for the P2P streaming. The source is running in a Docker container. The source is responsible for converting the video stream into chunk data that is sent to the peers. In the default configurations of PeerStreamer a single chunk is comprised of one frame of the streaming video. We choose as a source a stable node with good connectivity and bandwidth to the camera in order to minimize the video frame loss from the networked camera. The location of the source in such a dynamic network is of high importance. Placing the source across slow wireless links impacts the QoS and QoE that peers will perceive.

In order to determine the impact of our algorithm when placing the source node, we measure the average chunk loss percentage at the peer side. Chunk loss metric defines the percentage of chunks that are lost and not arrived in time. This way helps us to understand the impact of the network on the reliable operation of live-video streaming service.

Our experiment is composed of 20 runs, where each run has 10 repetitions, and is averaged over all the successful runs (90% of the runs were successful). In the 10% of the runs the source was not able to get the stream from the camera, so peers did not receive the data. The measurements we present consists of 2 weeks of experiments, with roughly 100 hours of actual live video distribution and several MBytes of logged data. The presented results are from one hour of continuous live streaming from the PeerStreamer source.

Figure 11 shows the average chunk loss for different cluster sizes. The data reveals that for any number of cluster *k*, our BASP algorithm outperforms the currently adopted random placement in Guifi.net. For k=1, using our BASP algorithm

[11]https://community-lab.net/

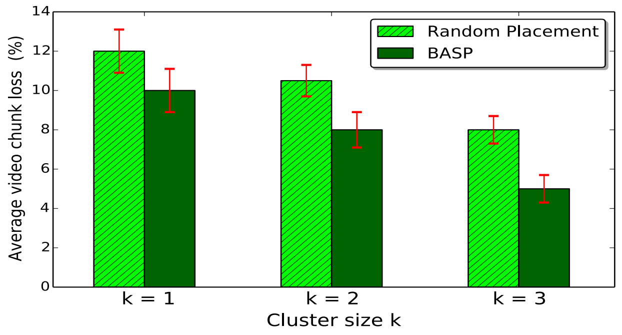

Figure 11. Average video chunk loss in QMP

the average chunk loss is decreased from 12% to 10%. This is the case when we have one source node streaming to the 20 peers in the network. Based on the observations from Figure 11, the gap between the two algorithms is growing as k increases. For instance, when k=3, we get 3% of improvement regarding chunk loss.

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.

VI. Related Work

Service placement is a key function of cloud management systems. Typically, by monitoring all the physical and virtual resources on a system, service placement aims to balance load through the allocation, migration and replication of tasks.

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, which in practice is much more efficient than finding an optimal solution. In [12] the authors proposed an optimal allocation solution for ambient intelligence environments using tasks replication to avoid network performance degradation. Volley [13] 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 [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.

Service Migration: Regarding the service migration in distributed clouds, few works came out recently. The authors in [17] and [18] 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 [19] studies when services should be migrated in response to user mobility and demand variation.

Another related work is [20] which proposed several algorithms that minimize the coordination and overlay cost along a network. The focus of our work in this paper however is to design a low-complexity service placement heuristic for community network clouds to maximise bandwidth.

VII. Conclusion

In this paper, we motivated the need for bandwidth-aware service placement on community network micro-cloud infrastructures. Community networks 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 related 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 community network 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 in deploying a CNMC. 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 35%. 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 community network, and quantified the performance and effects of our algorithm. We conducted our study on the case of a live video streaming service PeerStreamer 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%.

As a future work, we plan to look into service migration, i.e, the controller needs to decide which CNMC should perform the computation for a particular user, with the presence of user mobility and other dynamic changes in the network. In this problem, the user may switch between CNMCs thus another question is whether we should migrate the service from one CNMC to another cloud when the user location or network condition changes.

References

[1] B. Braem *et al.*, "A case for research with and on community networks," *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 3, pp. 68–73, Jul. 2013.

[2] "Guifi.net: Services of Catalunya (by zone)," https://guifi.net/en/node/2413/view/services.

[3] M. Selimi *et al.*, "Cloud services in the guifi.net community network," *Computer Networks*, vol. 93, Part 2, pp. 373 – 388, 2015, community Networks.

[4] L. Wang, S. Bayhan, J. Ott, J. Kangasharju, A. Sathiaseelan, and J. Crowcroft, "Pro-diluvian: Understanding scoped-flooding for content discovery in information-centric networking," in *Proceedings of the 2nd International Conference on Information-Centric Networking*, ser. ICN '15. New York, NY, USA: ACM, 2015, pp. 9–18.

[5] 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.

[6] 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.

[7] A. Lertsinsrubtavee, L. Wang, A. Sathiaseelan, J. Crowcroft, N. Weshsuwannarugs, A. Tunpan, and K. Kanchanasut, "Understanding internet usage and network locality in a rural community wireless mesh network," in *Proceedings of the Asian Internet Engineering Conference*, ser. AINTEC '15. New York, NY, USA: ACM, 2015, pp. 17–24.

[8] 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.

[9] D. V. et al., "A technological overview of the guifi.net community network," *Computer Networks*, vol. 93, Part 2, pp. 260 – 278, 2015, community Networks.

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

[11] K. e. a. LaCurts, "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] 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.

[13] 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.

[14] M. Steiner and B. e. a. Gaglianello, "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] S. Wang, R. Urgaonkar, M. Zafer, T. He, K. Chan, and K. K. Leung, "Dynamic service migration in mobile edge-clouds," *CoRR*, vol. abs/1506.05261, 2015.

[18] 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.

[19] R. Urgaonkar *et al.*, "Dynamic service migration and workload scheduling in edge-clouds," *Performance Evaluation*, vol. 91, pp. 205 – 228, 2015, special Issue: Performance 2015.

[20] 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.