Gossip-based Service Monitoring Platform for Wireless Edge Cloud Computing

Nuno Apolónia*, Felix Freitag*, Leandro Navarro*, Sarunas Girdzijauskas†, Vladimir Vlassov†
*Universitat Politècnica de Catalunya. Barcelona, Spain
{apolonia, felix, leandro}@ac.upc.edu
†KTH Royal Institute of Technology. Stockholm, Sweden
{sarunasg, vladv}@kth.se

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

Keywords: monitoring, community networks, cloud services, computing constrained-devices, gossip overlay

I. Introduction

The edge environment we introduce in this paper is based on community network (CN) environments. CNs are large-scale, self-organised and decentralized communication infrastructures built and operated by the community itself. Hundreds of CNs are operating, and are geographically distributed in different parts of the world, without relying on any specific social or economic reasons. The larger networks have from 500 to 40,000 nodes, such as Guifi.net[1], Freifunk, or AWMN.

Participants in CNs can also share local cloud computing resources and provide local cloud services. In this way, the community creates its own edge cloud computing environment without relying on classic clouds from outside the network. In our case, Cloudy[2] is used to manage the edge cloud computing and services such as Peerstreamer [1], [2], an open source P2P Media streaming, or Tahoe-LAFS [3], an open source decentralized cloud storage system.

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

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

We developed the monitoring platform for Cloudy, in order to give community network clouds an efficient monitoring service. The platform considers the way to handle the dissemination of data, by using a gossip overlay to intercommunicate with the nodes. Also, the platform gathers distributed data, by using the gossip-enabled network through which it disseminates data. The gossiping properties are aligned towards its use on edge cloud computing, since it provides eventual consistency of the data without relying on a single entity, and can still be used when node churn is evident. Other methods, such as flooding, direct communication does not deal gracefully with the issues that arise from these types of environments.

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

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

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

The rest of the paper is organized as follows. Section II describes background knowledge on wireless mesh networks,

[1]http://guifi.net, https://freifunk.net/, http://www.awmn.gr
[2]http://cloudy.community

CNs clouds and gossip-enabled networks. Section III describes the monitoring platform. Section IV refers to the monitoring prototype as feasibility study. Section V presents the results done with emulation of nodes and simulation of network. Section VI presents the discussion on the monitoring platform. The related work is described and summarized in section VII. The last section concludes and specifies the direction of future work.

II. Background

A. Wireless mesh networks

Wireless mesh networks have emerged as a specific modality in networking. The use of wireless mesh networks started to generate new concepts and paradigms towards mobility in devices, such as the case of vehicular networks [4]. Our case study for wireless mesh networks is CNs, which is explained below in more detail.

CNs are a communication infrastructure model, in which local communities of citizens build, operate and own their own part of the network. CNs often originated to provide Internet access to the population of areas which are unattended by commercial telecommunication operators.

Guifi.net, with more than 32,000 nodes (2017), can be considered one of the largest CNs worldwide. The case we analyze is a real deployment of cloud computing infrastructure and services in Guifi.net: a subset of devices located around Barcelona in Spain. The resources available in the CNs cloud infrastructure are low-capacity devices shared by citizens.

CNs have characteristic properties, such as, varied latencies between nodes [5], dynamic routing changes and low-capacity devices used for node interconnection. Also, node connectivity is based on mesh routing protocols [6].

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

Furthermore, by understanding services and resources properties, and how users interact, we can improve the organization of the CNs cloud. The monitoring platform can help gather information on the network, services and its users to improve the edge cloud performance.

B. Edge Cloud Computing

Edge cloud computing is a case of cloud computing, moving computation to the resources at the network edge. Thus, fully utilizing the edge resources and relying less on the Internet cloud resources. In this way, edge cloud computing works to share its own services and resources without having to go outside of the local network (i.e. Internet) to utilize cloud services.

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

Our case study of edge cloud computing in CNs is based on Cloudy. Cloudy is a distribution based on Debian GNU/Linux, intended to be used by common users. Therefore, Cloudy's development was driven by important aspects, such as the ease of usage, deployment in low-capacity devices, automated service discovery and services pre-configuration.

Furthermore, community services are included in the distribution, in order to facilitate the process for edge cloud computing, e.g. Peerstreamer as a peer-to-peer based live streaming, Tahoe-LAFS as a decentralized storage service, Syncthing[3] as a data synchronization between various storage nodes, among others. Also, the shared services within Cloudy are expected to be announced to the network (published/unpublished) in an automated way, when initiated by the users.

In Cloudy, the discovery service makes use of an overlay network created through existing technology, such as Serf and AVAHI[4], in order to specifically cluster nodes and manage service availability in the CNs clouds. The available shared services are then known to the other users by means of the GUI webpage, which informs about the situation of the services, and gives the ability to connect to the presented shared services.

C. Gossip-Enabled Networks

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

The purpose of having gossip overlays over networks is to overcome the issues of node discovery, detection or data dissemination [7]. In addition, gossip-enabled networks can scale with the network, since each node is only required to perform a fixed set of operations for dissemination; the network becomes resilient to node failures, node failure has little impact on the dissemination of data; avoids overloading the network with data, while ensuring all nodes eventually learn about shared information. Moreover, gossiping protocols rely on eventual consistency, where all nodes will have the data within a time-frame. Therefore, an issue on the gossip approach is that not all points of the network have the same information at the same time.

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

Our case study for gossip-enabled networks is Serf, a system that creates a gossip overlay between different members of a network [8]. Each node has a local agent that sends and receives messages from the other nodes. Each agent publishes its information to the members of the network, e.g. includes the nodes' name, number of members known, events queued to be processed and other tags with custom information. Thus, additional information can be shared between members, apart from the default information from Serf, by using custom tags. Furthermore, each interconnected node through Serf spreads the information to their neighbor nodes (T_{fanout}), 3 nodes by default. The gossip interval (T_{gossip}) to send data is also adjustable as a configuration option, with a default of 0.2 seconds.

III. Monitoring platform for Edge Clouds

We extended the Cloud distribution with a monitoring platform, towards enhancing the information gathered from

[3]https://www.syncthing.net/

[4]http://avahi.org

edge cloud services. The platform aims to gather raw data from the shared devices, and disseminate the relevant information to the community.

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

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

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

The monitoring platform is split in three stages: 1) Logging of raw data from services, resources and user interaction shared by the member nodes; 2) processing of the raw data, into a format that is user readable and can be shared among the community, such as a time-line of service usage; 3) showing the results to the users in the community, through the user interface, which is an additional service in Cloudy. These stages represent the major objectives on which we focus our attention, tailoring them towards an efficient monitoring service on the edge community cloud.

Figure 1 presents an overview of the modules in the monitoring platform integrated in Cloudy. The integration of the monitoring part is done in the middle layer, where a service is added to enable users to install and see the results of monitoring across the network. Additionally, the resource and service monitoring module is added in the communication path with other nodes and services, to gather the necessary information from the services and to disseminate it to other nodes using the gossip overlay. The monitoring module utilizes the already existing gossip overlay, created through Serf, to disseminate the relevant information to the other nodes. Meanwhile, the module also uses the same overlay to retrieve information from other nodes. Data dissemination and convergence in nodes happens at Serf level, since data is retrieved from the local Serf instance.

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

Additionally, the information gathered from the monitoring

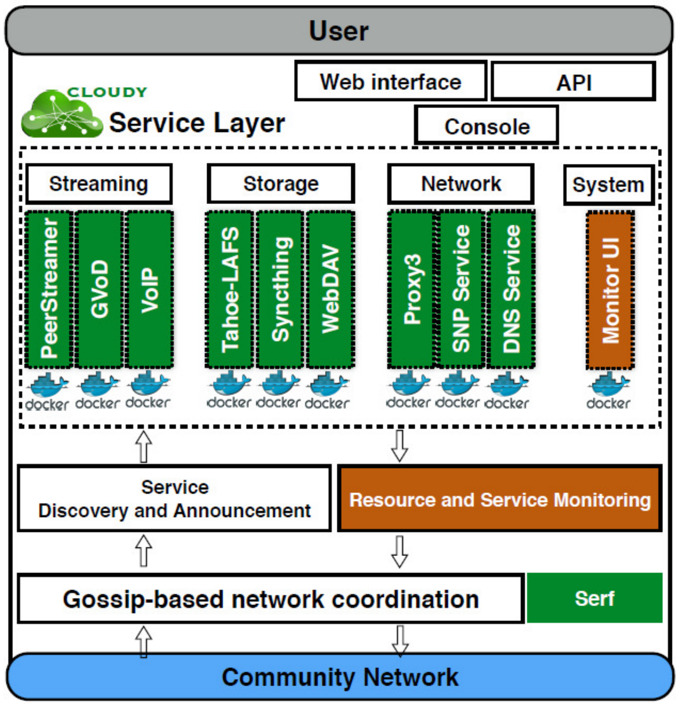

Fig. 1. Overview of Cloudy modules, showing the monitoring components and the integration with the gossip overlay.

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

IV. Monitoring Platform Prototype

The prototype for the monitoring platform was developed using the available low-capacity devices and connected to the community network (Guifi.net). The devices used are equivalent to those deployed by the Clommunity project[5] in the community network at users' homes.

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

In our prototype, we gather information and process it on one node, to be shown as a time-line graph of service usage, as seen in Fig. 2. The figure depicts the time of actions across three nodes, such as publishing and unpublishing of services, where each bar represents time-wise the service usage for a given node. Moreover, information relevant to the users, that comes directly from the nodes in the network, can be displayed in the GUI. Furthermore, since data travels through the gossip overlay, the whole network information (services from all nodes) becomes more accurate over time, when all nodes' data converges, as

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

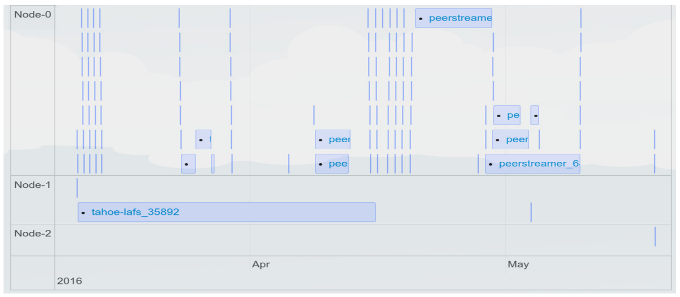

Fig. 2. GUI of the prototype for the monitoring of Cloudy services, using three devices interconnected.

low as one second in Serf[6].

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

V. Experimental Results

We conduct an evaluation by emulation of the monitoring platform, as the means to understand the characteristics and properties relevant to a real deployment. The simulation of network data gathered gives us insight on the best practices for an efficient way of dealing with dissemination of data across a wireless mesh network. Whereas, the emulation of nodes provides details about the scalability of the monitoring platform for edge cloud computing. In this way, we can understand the properties and issues that arise when dealing with higher number of nodes, higher latencies and the use of low-capacity devices.

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

For our evaluation, we emulate nodes and simulate the network, giving us the average of time for dissemination and convergence data rates, while also being able to tune certain aspects (such as gossip interval, gossip fan-out) of the overlay created through Serf. Therefore, we used the Mininet simulator [9], merged with Mininet-Wifi [10] and Mininet ContainerNet[7]. Mininet-Wifi adds to Mininet the ability of simulating wireless links. The capability of simulating wireless links is then more attuned with the environment created with community network clouds. The ContainerNet project enables each of the emulated nodes to run as Docker containers, guaranteeing execution isolation for each of the emulated nodes. Furthermore, we are able to run different executions of the same applications, such as Serf, without interference among each emulated node.

In the experiments done, we used as network topology a random geo-positioning of the nodes, where each of the nodes positions itself within a maximum range of 100 meters of another node. The characteristics of the topology are drawn from the CNs, where each person connects to their nearest neighbors to join the network. Therefore, each experiment run uses a randomly created topology, in an attempt to not be influenced by a given network topology.

Moreover, the positioning of the nodes influence the overall latencies in the network. The nodes average latencies can be as high as 800 milliseconds. Also noting that the simulated network shows as a worst case scenario, in fact other experiments done on CNs [11] tend to experience, on average, lower latencies and higher bandwidth between nodes, on normal usage of the network. However, our evaluation comprises the worst cases, to infer on the monitoring activity when high latencies are dominant in the network.

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

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

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

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

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

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

[6]https://www.Serfdom.io/docs/internals/simulator.html
[7]https://github.com/mpeuster/containernet

be unable to reflecting realistic conditions of resource usage.

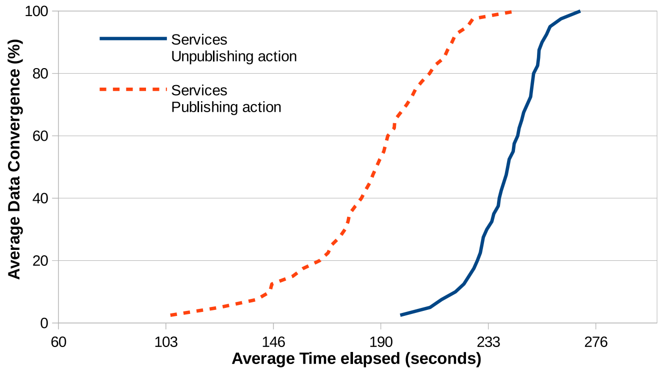

Fig. 3. Averaged data convergence in the time elapsed for the actions of publishing and unpublishing services.

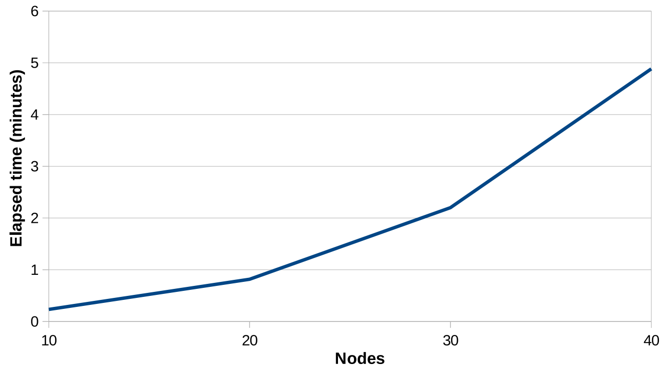

Fig. 4. Scalability results, between number of nodes and time elapsed until service is unpublished.

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

Fig. 3 shows the percentage for average monitoring data convergence for all nodes, after the services were published (dashed line) and unpublished (continuous line), averaged from all the experiments done. The actions are not immediately propagated to the network, therefore the dissemination occurs some time after starting. Also, we can see that the convergence of data in both actions happens within 1 minutes in high latency conditions. In this figure we observe the convergence is done faster for the unpublish action than for publish, this is because the nodes already know about others in the network, and thus it only requires to update the current information. The figure also demonstrates for each action the data convergence on all nodes, on average, is 130 and 80 seconds in each action respectively.

Fig. 4 demonstrates the scalability of having the monitoring data exchange with a gossip overlay. The figure shows the time elapsed of the combined actions for starting and stopping a service and the fully convergence of information within a node. Furthermore, we can say that the results obtained imply a linear evolution of the time elapsed when adding more nodes to the network with high latency connections, taking into account the overall latency (edge to edge) increase when adding more nodes. The results show that in the community network scenario and with the high latency conditions, the nodes can still gather information quickly enough to have the knowledge of the network without issues.

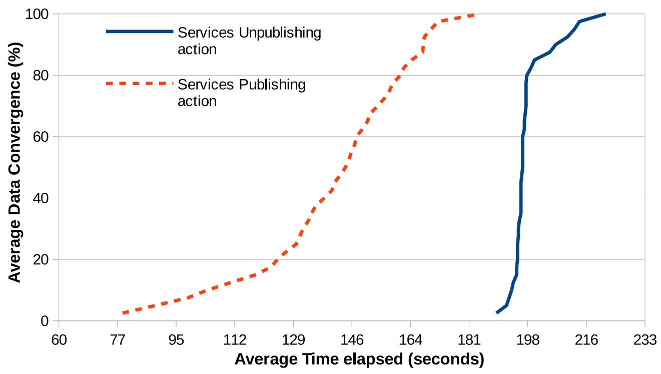

Fig. 5. Average data convergence in time elapsed for the actions of publish and unpublish services, using the tuned gossip properties for wireless devices.

Fig. 5 depicts the same actions (Publish and Unpublish of services) as previously, however the gossip properties (Gossip interval and Gossip fan-out) were adjusted for wireless environments. The figure shows that the optimization done has an effect on the data dissemination and convergence on the nodes. Furthermore, the figure also demonstrates that the convergence of data for the nodes, on average, is 87 and 38 seconds in each action respectively. Therefore, each of the actions shown, on average, are faster (around 25%) than with the previous gossip properties.

VI. Discussion

The evaluation provides relevant characteristics of monitoring in community network clouds. Therefore, we discuss about the usage of a gossip overlay to disseminate information, and monitoring non-critical information (information that is not necessary in real-time). However, we can also be assured that the dissemination is done quickly (within 1 minute under high latency situations), depending on other factors such as number of services running, size of shared information, bandwidth to each node, offline nodes or failures in the network.

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

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

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

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

VII. Related Work

Cluster monitoring and management, in the work of [12], is done through a hierarchical overlay network of the available resources. This work focuses on monitoring resources for management of clusters, and does not account for the actual information shared or the amount exchanged. The dissemination of data is done in a hierarchical manner, between nodes and masters. Also, the information sharing is done as a push based system in order for the masters to obtain the monitoring data from the nodes.

In [13] monitoring tools are used for workstations in clusters. Information sharing is done through a communication interface between nodes and monitoring proxies. An important lesson learned is that the behavior of monitoring clusters needs to be open environments, flexible and scalable. The behavior of the current monitoring tools only account for the system they are based on, and may not be flexible enough for handling different loads or resource types. This work uses workstations as clusters, and the inter-connectivity in the infrastructure is intended to be as the Internet infrastructure, which does not account for issues from wireless connectivity.

Monitoring Large-Scale Cloud Systems with Layered Gossip Protocols [14] presents an alternative to monitoring services through additional infrastructures in cloud systems. Their work is focused on the gossiping communication for data collection and monitoring on large scale cloud systems, aggregating data from the virtual machines deployed with the services, for a self monitoring process among the grouped virtual machines.

In summary, the works about monitoring tools and systems are focused on either grid and clusters or cloud environments. Thus, they do not account for changes in the network, as part of the infrastructure itself, and only node failures are detected. The bandwidth and latencies between nodes are not given as requirements for environments such as clusters or current cloud systems. The gossip-enabled solutions enhance communication towards handling information across several nodes, despite the network characteristics are not accounted.

VIII. Conclusion and future work

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

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

The directions for our future research include considering the levels of information that is disseminated, which can include social, service, resource and network aspects. Also, to improve the monitoring system with the addition of a decentralized solution through community detection and partitioning of the network. Furthermore, by deploying the monitoring platform in real usage scenarios we can enhance the knowledge and applicability of monitoring services on community network clouds.

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 contract TIN2016-77836-C2-2-R.

References

[1] L. Baldesi, L. Maccari, and R. Lo Cigno, “Improving P2P Streaming in Community-Lab Through Local Strategies,” in *10th IEEE Int. Conference on Wireless and Mobile Computing, Networking and Communications*, Larnaca, Cyprus, October 2014, pp. 33–39.

[2] R. Birke *et al.*, “Architecture of a network-aware p2p-tv application: The napa-wine approach,” *IEEE Communications Magazine*, vol. 49, pp. 154–163, 06/2011 2011.

[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] F. A. Silva *et al.*, “Vehicular networks: A new challenge for content-delivery-based applications,” *ACM Comput. Surv.*, vol. 49, no. 1, pp. 11:1–11:29, Jun. 2016.

[5] 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, Dec. 2015.

[6] A. Neumann, E. López, and L. Navarro, “Evaluation of mesh routing protocols for wireless community networks,” *Computer Networks*, vol. 93, Part 2, no. P2, pp. 308–323, Dec. 2015.

[7] R. Friedman *et al.*, “Gossiping on manets: the beauty and the beast,” *ACM SIGOPS Operating Systems Review*, vol. 41, no. 5, pp. 67–74, 2007.

[8] R. Baig, F. Freitag, and L. Navarro, “On the sustainability of community clouds in guifi.net,” in *Economics of Grids, Clouds, Systems, and Services*, ser. Lecture Notes in Computer Science, J. Altmann, G. C. Silaghi, and O. F. Rana, Eds. Springer International Publishing, 15 Sep. 2015, pp. 265–278.

[9] B. Lantz *et al.*, “A network in a laptop: rapid prototyping for software-defined networks,” in *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*. ACM, 2010, p. 19.

[10] R. R. Fontes *et al.*, “Mininet-wifi: Emulating software-defined wireless networks,” in *Network and Service Management (CNSM), 2015 11th International Conference on*. IEEE, 2015, pp. 384–389.

[11] M. Selimi, N. Apolonia *et al.*, “Integration of assisted p2p live streaming service in community network clouds,” in *Proceedings of the IEEE 7th International Conference on Cloud Computing Technology and Science (CloudCom 2015)*. IEEE, Nov. 2015.

[12] D. Park, S.-M. Lee, and C. Lee, “The cluster monitoring & controlling method with scalable communication framework,” in *Proceedings of the Eighth International Conference on High-Performance Computing in Asia-Pacific Region*, ser. HPCASIA ’05. Washington, DC, USA: IEEE Computer Society, 2005, pp. 387–.

[13] Z. Liang *et al.*, “Clusterprobe: An open, flexible and scalable cluster monitoring tool,” in *Proceedings of the 1st IEEE Computer Society International Workshop on Cluster Computing*, ser. IWCC ’99. Washington, DC, USA: IEEE Computer Society, 1999, pp. 261–.

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