Performance Evaluation of Federated Learning over Wireless Mesh Networks with Low-Capacity Devices

No Author Given

No Institute Given

Abstract. Federated learning is a distributed learning technique in which a machine learning model is trained collaboratively among several nodes. While the privacy preservation of the training data is one of the important promises of federated learning, there is also an opportunity to use low capacity devices for machine learning model training by taking advantage of the fact that the training effort is divided among many nodes. In this paper we conduct experiments with a federated learning network deployed on several low capacity devices connected to a wireless mesh network. The measurements show the hardware capacity and link bandwidth of the clients on the federated learning process. The results suggest that for heterogeneous networks the federated learning clients should be extended with more autonomous decision capacities according to the network and local conditions.

Keywords: edge computing, federated learning

1 Introduction

Federated learning (FL) is a recent approach to training machine learning models in a distributed fashion in which many client nodes participate in the model training [1]. Different to the centralized training approach requiring all training data to be available at a single location, in FL the client nodes train a machine learning model with their local data. This circumvents data needing to leave the node, which has important advantages for the privacy preservation of the data.

There is a strong tendency to run machine learning (ML) models on ever smaller computing devices. For the inference with trained models, simple recognition tasks can nowadays be performed even with devices as tiny as microcontroller boards [2]. The training of the machine learning models, however, is a more compute-intensive task, and imposes challenges with low capacity devices. GPUs instead of CPUs may be used for better performance. However, in low-capacity devices, such as mini-PCs or single-board-computers (SBCs), GPUs are not always available. As a consequence, there is a high load on the CPU during the training process, which also takes much longer time than on high-end devices.

Principally, the training of a neural network model with large datasets requires the availability of important computational resources during the training process. Federated learning, however, splits the training effort among several nodes. In the FL scenario, the local dataset at each client is considered to be smaller, since for instance the data may consist only of the information collected at this node. The lower amount of training data at each node opens the possibility to train models on low capacity devices, but it also introduces a new challenge to train models at different clients with non-independent and identically distributed data (non-IID), which adds complexity to the training process [3].

In this paper, we aim to study the federated learning process on computing devices consisting of mini-PCs or Single-Board-Computers (SBCs), such as those found in home environments. Typically, in user environments these computing devices run as home servers to manage several user-oriented services. Therefore, these devices are not dedicated exclusively to a machine learning application. A real scenario for this situation is the Guifi.net community network[1], where some users provide applications on such low-capacity nodes to other users [4]. From this scenario, an important research topic can be motivated: it is important to understand how federated learning consumes computing resources, since in non-dedicated home devices, i.e., those running multiple services, the resource consumption of federated learning must not affect the Quality of Experience (QoE) which the user perceives from other applications running simultaneously in the device.

In particular we analyze the resource usage of federated learning by means of experimentation on real devices connected to a wireless mesh network. We aim to understand better how the different phases of a federated learning round (i.e., the model exchange and local model training) affect the CPU and memory consumption of a low capacity device and the traffic in the network.

Therefore, the main contributions of the paper are:

1. We provide results on the computing resource usage of a federated learning process measured on real distributed low-capacity devices connected to a wireless mesh network.
2. The analysis of the results suggests the design of an adaptive client node enabling a context-aware federated learning in wireless mesh networks.

2 Related work

In this section, we review selected works related to the application of federated learning on low capacity devices and works which suggested mechanisms for the configuration or adaptability of federated learning.

In [5], an adaptive federated learning approach is proposed. The focus is on the global aggregation frequency parameter. Specifically, this work proposes a control algorithm to determine in real time after how many local training epochs at a node the model data is sent back to the aggregator node. This approach is different to the typically used fixed global aggregation frequency. The evaluation is performed by simulations and some experiments in real nodes consisting of

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

3 Raspberry Pi and 2 laptop computers. We note that the global aggregation frequency is determined by the aggregator node in a centralized fashion, while an alternative could be to determine at each node the individually most suitable number of training epochs.

In [6], a highly-efficient federated learning framework is presented. The heterogeneity of worker nodes given in the context of the IoT is addressed. Two measures are suggested, which are relaxed worker synchronization for tolerating dropouts of sporadic workers, and similarity-based worker selection, which aims to select a subset of the most efficient workers. By calculating the similarity among the received local models, the server can decide to exclude certain worker nodes for the next training round, e.g., those which may not contribute sufficiently to the global model. The principal idea is to empower the FL server to take smarter decisions on how to orchestrate the FL process over the worker nodes. The proposed system is evaluated in Google Cloud Platform focusing on the accuracy, but not on how the resource usage is affected.

In [7], a federated learning framework for the IoT is proposed. The specific application is to detect anomalies in the network traffic of IoT devices. The scenario is motivated by cybersecurity requirements, in which the communication overhead of a centralized over-the-cloud approach is unfeasible. For the anomaly detection, a deep autoencoder model is trained in a federated learning fashion on each node. The evaluation is performed on real devices, namely Raspberry Pi model 4 and NVIDIA Jetson Nano. While the evaluation focuses on the accuracy of the detection, it was also stated that only a small fraction of the 4GB memory of the Raspberry Pi was used.

The work of Y. Gao et al [8] performs an empirical evaluation of two different state-of-art machine learning techniques, namely split neural networks (SplitNN) and federated learning. For an end-to-end evaluation, a variety of datasets, different model architectures, multiple clients and various performance metrics were considered. The learning performance was assessed for two types of distributed data, imbalanced and non-IID data. Model training was done on Raspberry Pi devices, where the CPU consumption, memory usage, communication overhead and training time was measured. From the experiments, the authors conclude that FL overall perform better in comparison with SplitNN, because of the lower communication overhead.

From the reviewed related works, it can be seen that there are several approaches proposed for reducing the computational resource consumption of federated learning, ranging from changes of the machine leaning model training up to off-loading of specific services to other platforms. However, there is still a lack of results on the performance of FL in real environments. Our work aims to provide practical results by running federated machine learning on low-capacity devices, giving input on how to design the federated learning process for different end user environments.

3 Federated learning implementation

3.1 Federated learning architecture

We use a federated learning architecture consisting of a server as global aggregator and distributed client nodes which train locally an instance of the global model. Figure 1 shows the federated learning components and illustrates the principal idea: The aim is to train a global ML model hosted by the federated learning server. Training data is available locally at the client nodes. For a new training round, the server sends the current version of the global model to the client nodes. They train this model for a predetermined number of epochs with their local data. After the training, the updated model parameters are sent back to the server. The server then generates a new global model by averaging the model parameters received from the different clients nodes. The training phase may have several rounds initiated by the server.

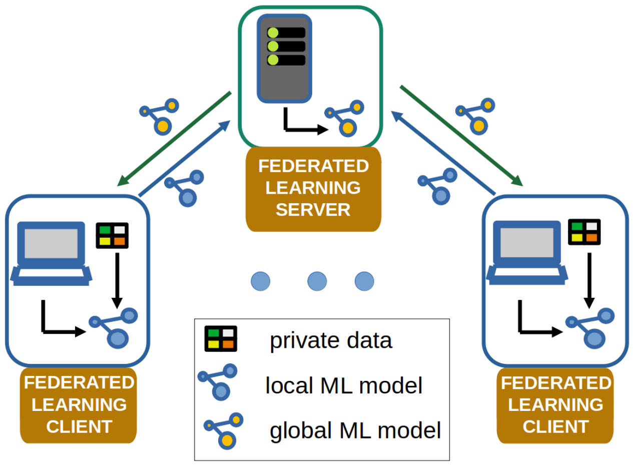

FEDERATED LEARNING SERVER
FEDERATED LEARNING CLIENT
FEDERATED LEARNING CLIENT
private data
local ML model
global ML model

Fig. 1: Federated learning architecture with interaction between the server and clients.

3.2 Implementation

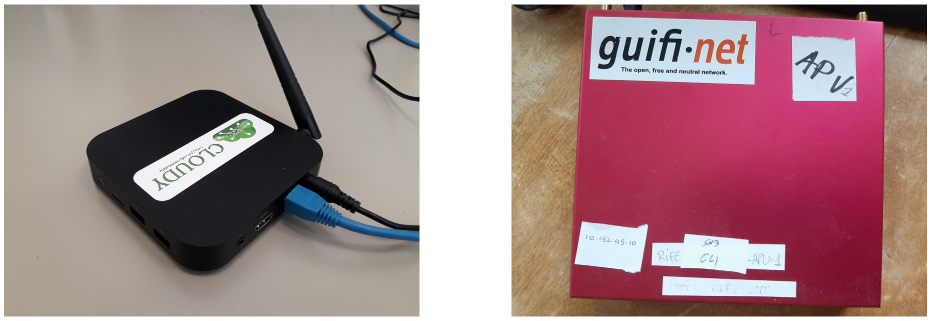
The federated learning network we use for the experimentation is implemented in Python language. The system is composed of two major components, i.e., the code for the client and the server. In our implementation, the server sends both the model parameters and the hyperparameters, which relate to the control of how the training is done on the clients. These hyperparameters assigned are the learning rate, number of local training epochs, and batch size. This data is sent between server and clients in JSON format over http POST messages, where both the server and the client implement a REST API. The server does not make a distinction between different client nodes, i.e., all clients receive the same value of the hyperparameters. For both the federated learning server and the client, we create Docker images in order to instantiate them with Docker containers on the different devices. The source code of the federated learning network is available on Github[2]. Additional information on the code design can be found in [9].

4 Experimental evaluation of the federated learning network

4.1 Experimental environment and testbed

We deploy the previously introduced federated learning network on a testbed of low-capacity computing nodes connected to a wireless mesh network called GuifiSants[3]. GuifiSants is part of the Guifi.net community network. The Guifi.net communication network is an infrastructure of more then 30.000 interconnected heterogeneous network devices (wired and wireless), belonging to the thousands of community network members [10].

Within the Guifi.net communication infrastructure, edge computing services started around 2015 [4]. These edge devices located at the premises of the community network member typically host the owner's specific application services but also host some community-oriented service for helping to manage the network, such as contributing to network monitoring [11]. In order to have a low energy consumption, these edge devices are often mini-PCs.



(a) Minix mini-PC.

(b) PC Engines APU2.

Fig. 2: Computing nodes of the testbed used for federated learning experiments.

[2] `https://github.com/eyp/federated-learning-network`

[3] Live monitor of GuifiSants. `http://dsg.ac.upc.edu/qmpsu/index.php`

In order to build a testbed for the federated learning experimentation, we have connected several Minix mini-PCs[4] (Figure 2a) and PC Engines APU2[5] (Figure 2b), both type of devices with Debian 10 Buster installed, to the GuifiSants wireless mesh network. As such, they form a testbed which is part of the production network and allow to experiment under real conditions (Figure 3).

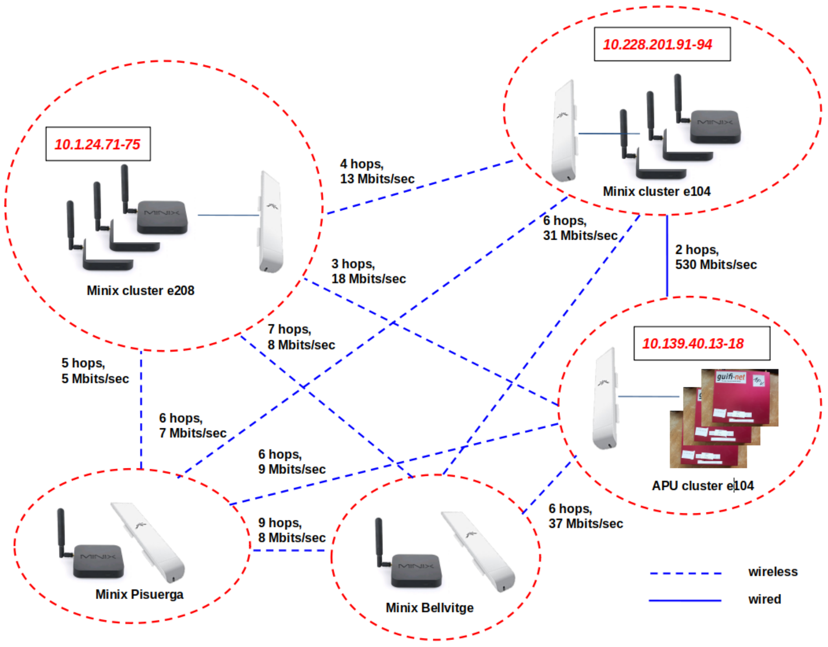


Fig. 3: Testbed infrastructure and approximate bandwidth between the locations.

4.2 Experimentation

The objective of the experimentation is to measure the resource consumption of federated learning devices interconnected over the wireless mesh network.

For the experiment, the federated learning task to be executed is to train a CNN (Convolutional Neural Network) model with the Chest_X_ray dataset[6]. Since the testbed nodes are of a low computing capacity, the clients are configured to train 1 epoch in each round and the number of images for training and testing are reduced to 200 and 100, respectively. The machine learning model

[4] Minix NEO Z83-4 with Intel Atom x5-Z8350 processor and 4GB DDR3 RAM. `https://minix.com.hk/products/neo-z83-4-pro`

[5] PC Engines APU2 with AMD Embedded G series GX-412TC processor and 4 GB DDR3 RAM. `https://pcengines.ch/apu2e4.htm`

[6] Chest X-Ray Images. `https://www.kaggle.com/paultimothymooney/chest-xray-pneumonia`

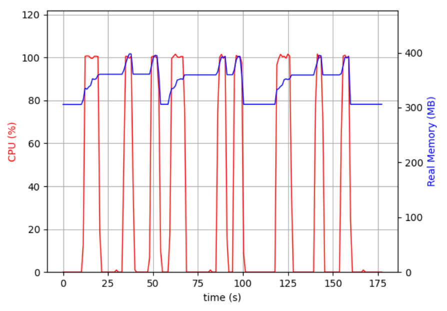
to be trained is a 6-layer CNN, which has around 420.000 parameters. Three rounds are trained in both experiments.

Experiment 1: Federated learning clients on different hardware. The objective of this experiment is to observe the behaviour and resource consumption of FL clients *when run on different hardware*. For this experiment we run one client in a device of the APU2 cluster e104, and the other client in a device of the Minix cluster e104 (see Figure 3). The server is deployed on another device in the PC Engines APU2 cluster e104. Figure 4 shows the results. Comparing the times in Figure 4d to 4f, when the model is exchanged with the server, it can be seen that the client in the Minix device replies quicker to the server with the trained model than the client in the APU2.

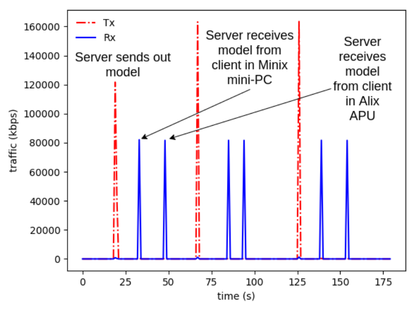
Experiment 2: Federated learning clients with different link bandwidth. In this experiment we aim to observe *the effect of different link bandwidth available at the clients*. We chose one device from each of the 5 locations of the testbed. The FL server is installed in one of the APU2 devices from its cluster e104. The FL clients are installed on Minix Pisuerga device, Minix Bellvitge device, a device from the Minix cluster e208, and one from the Minix cluster e104, in total 4 clients, all on Minix devices.

Figure 5 shows the measured resource consumption. With regards to the CPU and memory consumption of the clients (Figures 5c, e), the three training rounds done by each client can be clearly observed by the peaks in the CPU consumption. For the training, almost the complete CPU capacity (the four cores) are used. The memory consumption is moderate, as being below 1 GB and taking into account that the devices have 4 GB RAM available. It can be observed that the FL client Minix Pisuerga (Figure 5c) started with a higher memory consumption compared to the other client (Figure 5e). This is due to the fact that the client Minix Pisuerga participated already previously in a federated learning round with the server, while the other clients joined the federated learning network later. The traffic produced during the federated learning rounds is shown in Figures 5d, f. It reflects the available bandwidth between the locations. For instance, in the low bandwidth link to the Minix Pisuerga client, the traffic produced by sending the ML model between the client and server is lower and takes longer, while in the faster link of the client in the Minix cluster e208 the traffic due to the model exchange has higher peaks.

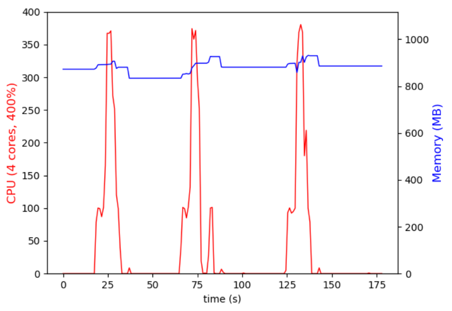
Figure 5a, b shows the resource consumption of the FL server. The CPU consumption in the server is clearly lower than in the clients, where the server uses approximately 1 core (Figure 5a). The peaks of the CPU consumption seem to correspond to the communication phases with the clients, in which the model exchange takes place. The memory consumption of the server with around 0.5 GB is low and lower than the approx. 1 GB memory used by the FL clients. The traffic shown in Figure 5b represents the model exchanges with the four clients during the three training rounds. The peaks correspond to the communication with the high capacity links. It can be observed how the low capacity link (in the testbed, the link to the Minix Pisuerga client) delays the finalization of the federated learning rounds.



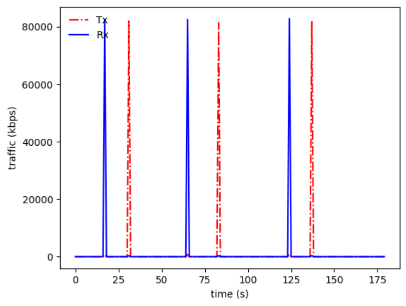
(a) FL server in APU2: CPU and memory consumption.



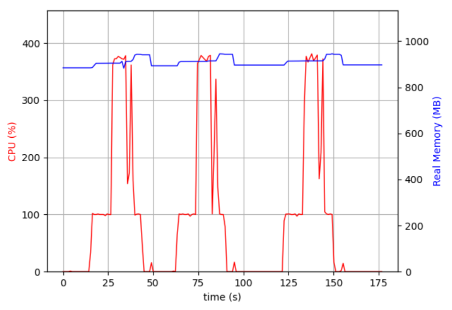
(b) FL server in APU2: bandwidth consumption.



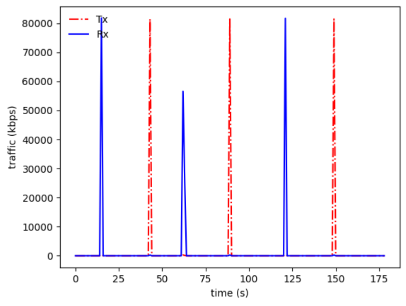
(c) FL client in Minix mini-PC: CPU and memory consumption.



(d) FL client in Minix mini-PC: bandwidth consumption

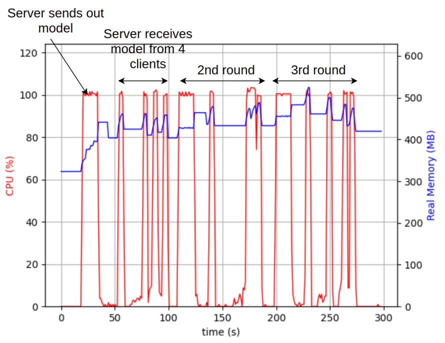


(e) FL client in APU2: CPU and memory consumption.

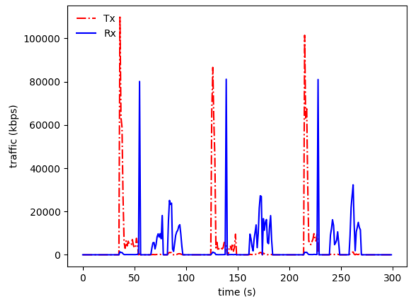


(f) FL client in APU2: bandwidth consumption

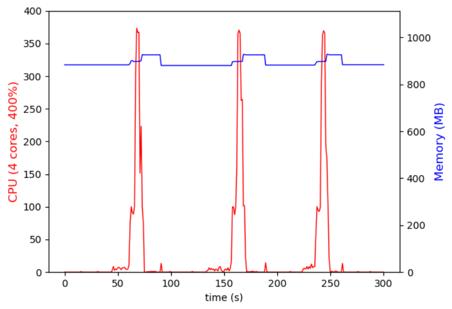
Fig. 4: Federated learning clients in different hardware: Resource consumption in three training rounds.



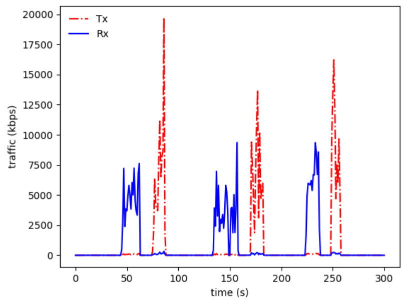
(a) FL server in APU2 cluster e104: CPU and memory consumption.



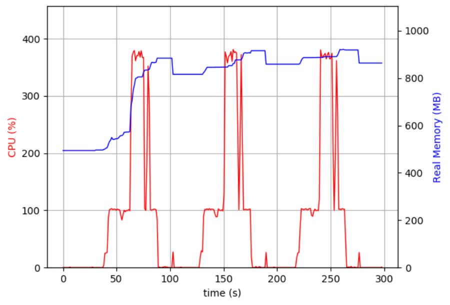
(b) FL server in APU2 cluster e104: bandwidth consumption.



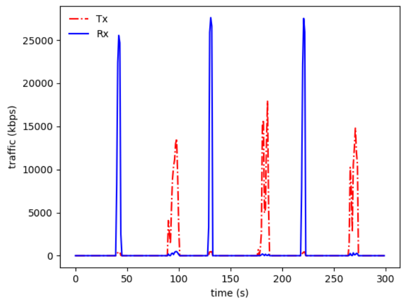
(c) FL client Minix Pisuerga: CPU and memory consumption.



(d) FL client Minix Pisuerga: bandwidth consumption.



(e) FL client in Minix cluster e208: CPU and memory consumption.



(f) FL client in Minix cluster e208: bandwidth consumption

Fig. 5: Federated learning clients with different link bandwidth: Resource consumption in three training rounds.

5 Conclusions

This paper presented a federated learning deployment using low capacity devices in a wireless mesh network. The resource consumption of the clients and server in terms of CPU, memory, and bandwidth consumption were measured. During the model training at the clients a high CPU consumption which uses the four cores of the processor was observed, which could be a problem if other applications run simultaneously on the device for which certain service levels must be guaranteed to end users. The experiments furthermore showed how a low bandwidth link of clients delay the model exchange and thus the finalization of a of federated learning round, leading to an overall slower model training. Therefore, for improved performance in such environments with heterogeneous bandwidth and hardware of the clients, the obtained results suggest to design federated learning clients which can dynamically adapt the training parameters to enable a context-aware federated learning.

References

1. Yang, Q., Liu, Y., Chen, T., Tong, Y.: Federated machine learning: Concept and applications. ACM Trans. Intell. Syst. Technol. **10**(2) (January 2019)
2. Sakr, F., Bellotti, F., Berta, R., De Gloria, A.: Machine learning on mainstream microcontrollers. Sensors **20**(9) (May 2020) 2638
3. Li, T., Sahu, A.K., Talwalkar, A., Smith, V.: Federated Learning: Challenges, Methods, and Future Directions. IEEE Signal Processing Magazine **37**(3) (May 2020) 50–60
4. Baig, R., Freitag, F., Navarro, L.: Cloudy in guifi.net: Establishing and sustaining a community cloud as open commons. Future Generation Computer Systems **87** (2018) 868–887
5. Wang, S., Tuor, T., Salonidis, T., Leung, K.K., Makaya, C., He, T., Chan, K.: Adaptive federated learning in resource constrained edge computing systems. IEEE Journal on Selected Areas in Communications **37**(6) (2019) 1205–1221
6. Xu, H., Li, J., Xiong, H., Lu, H.: Fedmax: Enabling a highly-efficient federated learning framework. In: 2020 IEEE 13th International Conference on Cloud Computing (CLOUD). (2020) 426–434
7. Zhang, T., He, C., Ma, T., Ma, M., Avestimehr, S.: Federated learning for internet of things: A federated learning framework for on-device anomaly data detection (2021)
8. Gao, Y., Kim, M., Abuadbba, S., Kim, Y., Thapa, C., Kim, K., Camtep, S.A., Kim, H., Nepal, S.: End-to-end evaluation of federated learning and split learning for internet of things. In: 2020 International Symposium on Reliable Distributed Systems (SRDS). (2020) 91–100
9. Parareda, E.Y.: Federated learning network: Training distributed machine learning models with the federated learning paradigm. (2021)
10. Baig, R., Roca, R., Freitag, F., Navarro, L.: Guifi.net, a crowdsourced network infrastructure held in common. Comput. Netw. **90**(C) (October 2015) 150–165
11. Centelles, R., Selimi, M., Freitag, F., Navarro, L.: Redemon: Resilient decentralized monitoring system for edge infrastructures. In: 2020 20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (CCGRID), Los Alamitos, CA, USA, IEEE Computer Society (may 2020) 91–100