# mMTC Deployment over Sliceable Infrastructure: the Megasense Scenario

Naser Hossein Motlagh[1], Ibrahim Afolabi[2], Matteo Pozza[1], Miloud Bagaa[2],
Tarik Taleb[2,3,4], Sasu Tarkoma[1,3], Hannu Flinck[5]
[1]Department of Computer Science, University of Helsinki, Helsinki, Finland
[2]Department of Communications and Networking, Aalto University, Espoo, Finland
[3]University of Oulu, Oulu, Finland; [4]Sejong University, Seoul, Korea
[5]Nokia Bell Labs, Espoo, Finland
Emails: [1]firstname.lastname@helsinki.fi; [2]firstname.lastname@aalto.fi; [5]hannu.flinck@nokia-bell-labs.com

***Abstract*—Massive Machine Type Communication (mMTC) has long been identified as a major vertical sector and enabler of the industry 4.0 technological evolution that will seamlessly ease the dynamics of machine-to-machine communications while leveraging the 5G technology. To advance this concept, we have developed and tested an mMTC network slice called Megasense. Megasense is a complete framework that consists of multiple software modules, which is used for collecting and analyzing air pollution data that emanates from a massive amount of air pollution sensors. Taking advantage of the 5G networks, the Megasense will significantly benefit from an underlying communication network that is traditionally elastic and can accommodate the on-demand changes in requirements of such a use case. As a result, deploying the sensor nodes over a sliceable 5G system is deemed the most appropriate in satisfying the resource requirements of such a use case scenario. In this light, in order to verify how 5G-ready our Megasense solution is, we deployed it over a network slice that is totally composed of virtual resources. We have also evaluated the impact of the network slicing platform on the Megasense in terms of bandwidth and resource utilization. We further tested the performances of the Megasense system and come up with different deployment recommendations based on which the Megasense system would function optimally.**

***Index Terms*—Air Quality Sensing, 5G Networks, mMTC, Sensors, NB-IoT, Network Slice.**

## I. Introduction

5G networks are expected to provide fundamental support for variant vertical industry applications and open the door to several others. In particular, 5G networks will provision the necessary communication service as well as the resources needed to enable the services towards powering up the different identified 5G network slice types. Typical examples of network slices are ultra-reliable low latency communication (uRLLC), enhanced/extreme mobile broadband (eMBB) and massive machine type communication (mMTC) slices [1]. mMTC is characterized as a major enabling backbone for smart cities applications. These applications are intended to provide improved services in areas such as healthcare, wearables and eCity/smart city, so as to impact the well-being and living conditions of the people in a positive way. Thanks to the capabilities of the next generation mobile technology and the developments in the field of sensors and actuators, this anticipated improved living condition is gradually becoming attainable [2].

One of the pillars of smart cities is sensing, i.e., collecting and analyzing data from a massive number of sensors that is densely deployed in the city. 5G networks with its network slicing enabling technology is able to support the requirements coming from the sensing use cases [3]. Fundamentally, network slicing allows the provisioning of network services from programmable virtual network functions (VNFs) that are running over a shared physical network infrastructure with logical isolation of the virtual resources in such a way that guarantees a certain level of security. Moreover, network slicing brings flexibility to network service deployment through the provisioning of virtual resources that dynamically scale on demand based on the amount of needed resources [4]. This essential feature makes 5G network slice a perfect choice to enable applications such as Megasense to adequately manage the collection and analysis of data streams from a deployment of a massive number of sensors within a city. Indeed, 5G networks with their network slicing features significantly supports new services and vertical applications (known as 5G verticals such as Megasense) in megacities [3].

Megasense is a project that is developed with the sole aim of providing real-time massive scale air quality sensing in urban areas by integrating a large number of air quality sensors at the scale of for example tens of thousands or even millions of air quality sensors [5]. Megasense enables air pollution hotspot detection outdoors; air quality sensing indoors; and offers creation of real-time air pollution map in urban areas [6]. To obtain real-time air quality data from local variations with high resolution at the resolution of few meters, Megasense uses any existing air quality infrastructure in the city such as city air quality stations, crowd-sensing approaches, and citizens personal air quality sensors willing to share their sensed data. The air quality sensor deployment within Megasense also includes NB-IoT enabled air quality sensor nodes that are installed in the parking lot of the Kumpula campus at the University of Helsinki (shown by red arrow) are depicted in Figure 1. Whereas, Figure 1(a) shows the locations of the sensor deployments. Figure 1(b) shows the parking and the


(a) Locations of deployed air quality sensors in Helsinki. The details of NB-IoT sensor are shown in sub-figures (b) and (c).

(b) NB-IoT enabled air quality sensor installed in the parking of Kumpula campus at the University of Helsinki.

(c) The electronics and sensing units of NB-IoT based air quality sensor.

Fig. 1. Air quality sensor deployment within Megasense in Helsinki with an example of NB-IoT enabled air quality sensor.

place where a NB-IoT sensor is installed and Figure 1(c) illustrates the electronics and sensing utilities of the air quality sensor, capable of measuring aerosol and gas pollutants.

Among the partners of the 5G FORCE project, Aalto University (AU) maintains the network slicing prototype, while UH operates the Megasense air quality sensing platform, which has been developed for the collection of air quality measurements coming from an arbitrary number of sensors. This paper has deployed the referred network slicing prototype over two different cloud facilities offered by AU and UH. While the sensor nodes are connected to the slice on the UH cloud facility, the core of the Megasense framework, which collects and processes the data, is deployed on the AU cloud facility, as shown in Figure 2. In addition, to transmit the sensor data from the UH cloud to AU cloud, the Finnish University and Research Network (FUNET) which is a backbone network providing Internet connections for research organizations in Finland was used. FUNET supports sending data files even at several gigabytes in sizes.

*The Objective* of our implementation and empirical evaluation is to measure the impact of a growth in the number of sensors on the performance of Megasense while running on a network slice. This reveals the most critical requirements coming from the sensing use case, which helps planning the sensing infrastructure of a smart city when deployed on a network slice. We also investigate the level of support that an underlying network slice and its resource can provide for a smooth operation of the Megasense platform. This determines the amount of virtual resources that would be needed to allocate to a network slice in ensuring an efficient operation of the sensing platform based on the number deployed sensors.

The contribution of this work consists of three folds:

- We have designed and developed a network slice platform that is able to run and manage multiple network slices simultaneously that have different objectives and service level agreements (SLAs). Different network slices run on top of a common physical infrastructure, each of which targets different use-cases including uRLLC, eMBB and mMTC, respectively.
- We have developed and tested an mMTC network slice (i.e, Megasense) that enables the sensing, measurement and processing of the air pollution and providing an accurate real-time air pollution map in urban areas.
- We have evaluated the impact of the network slicing platform on the Megasense in terms of bandwidth and resource utilization.

This paper is organized in the following fashion. Section II reviews some related works, while Section III describes the deployment of Megasense use case over a network slice. In Section IV, we present the testbed and the results of the performance evaluation. Finally, the paper concludes in Section V.

## II. Related Work

A significant portion of research works investigate the mMTC type of communication when it comes to the 5G network support. These studies have been carried out in order to facilitate the use of access technologies to support the deployment and operation of a massive number of sensor nodes and their networking. The research in [7] introduces a multi-site mMTC test network to investigate the long-term impact on communication quality and sensor data when the network is deployed over either Long Range (LoRa) or NB-IoT technology. This is essential in terms of separation of concerns with respect to identifying possible causes of degradation in communication quality whether from software or hardware failures. The research discovers that while NB-IoT presents better communication performance, measurements on the LoRaWAN networks show higher SNRs and RSSI.

Moreover, the research in [8] aims at actualizing energy efficiency in the deployment of NB-IoT and LoRa-equipped sensor devices using the method of multi-radio massive machine-type communication (MR-mMTC). Sensor devices that are integrated with MR-mMTC capabilities experienced an improved performance when using multiple access technologies. Another study [9] considers the energy consumption

and channel capacity utilization by wireless sensor networks. The study compares the efficiency of traditional base stations (BSs) against a high altitude platform (HAP). The results show that the energy consumption of WSN using HAPs is more efficient than WSN using BSss. In the same manner, the channel capacity of the WSN systems using HAPs is greater than the WSN systems using BSs. As a result, a gain in the performance efficiency of the sensor networks was found when using a HAP.

The research in [10] investigates harnessing the opportunities provided by satellite communication through the use of capillary networks, which is based on Fibre to the Cabinet technology incorporated with terrestrial radio stations in collecting data from sensor networks. This approach provisions support for multi-service smart city ecosystem that relies on machine-to-machine type communications. The work in [11] studies the uplink communication coverage of mMTC deployments while considering its operation in an ultra-dense network environment. This work reveals that the uplink network performance is independent of the maximum transmission power, which then afford the sensor devices the possibility of a prolonged battery life. The study in [12] uses a joint application admission control and network slicing in virtual sensor networks to manage a shared physical WSN. The study proposes a mathematical model to solve the problems of application admission control and wireless sensor network slicing resource allocation. The other study in [13] proposes a horizontal hierarchy slicing method that is based on mathematical morphology technology to compress data in WSNs. As a result, the method shows the effectiveness of data compression in WSN through the slicing.

These studies aimed to facilitate the use of different access technologies and network slicing to support the deployment and operation of a massive number of sensor nodes. These studies aim to investigate the communication quality [7]; actualize energy efficiency in the deployment of sensor devices [8]; study the channel capacity utilization [9]; harness the opportunities provided by satellite communication [10]; communication coverage of mMTC deployments [11]; manage a shared WSN using joint application admission control and network slicing [12]; and compress data in WSNs through horizontal hierarchy slicing [13]. The studies presented in this section have taken different progressive approaches that could be utilized towards enabling the mMTC for dense IoT devices deployment. These studies have made tangible contributions towards facilitating the development of the mMTC technology considering different perspectives towards fulfilling the requirements of the communication type.

However, in this work, we have developed and tested an mMTC network slice called Megasense, which is a complete framework designed for collecting and analyzing air pollution data that emanates from a massive amount of deployed air pollution sensors. Megasense supports enabling ultra-dense MTC in order to facilitate a smart city application. We evaluated the impact of the network slicing platform on the Megasense in terms of bandwidth and resource utilization. We also showed the performance of Megasense when the number of deployed sensors drastically increases within the network slice. In addition, we consider the real dense and ultra-dense air quality sensor deployments and show how feasible it is to deploy such a use case over a sliceable network infrastructure. We further provide recommendations for real sensor deployment in smart cities. To the best of authors knowledge, this is the first paper highlighting massive sensor deployment using network slices, which considers a realistic number of air quality sensors as a 5G sensing use case and evaluates the deployment through simulations.

## III. Deployment of Megasense over a Network Slice

### A. Megasense System

Megasense system [6] consists of three main parts, the Sensing System, Edge Layer and Cloud Layer as shown in Figure 2. The Sensing system includes the sensing sources deployed in urban areas as well as the radio communication interfaces. The sensing sources include data received from any type of air quality sensor devices, open street map data, city air pollution monitoring stations, crowd-sourced data, and many other sources. The Cloud layer offers a long-term and scalable storage system as well as processing and analytics services. This layer aggregates data received from different sensing sources and stores the cleaned data. The processed data is further used by decision makers for clean urban planning as well as the research communities for obtaining insights about the air pollution sources and suggesting solutions for mitigating the pollution. The Edge layer is responsible for reactively receiving air quality data from the deployed sensors within Megasense as well as the external sensors owned by third parties. The edge layer is also responsible for data pre-processing, data cleaning and aggregating, calibrating and providing in real-time (seconds) air quality information at the edge using pre-trained machine learning models [14]. Both processed and raw data is further relayed to the cloud for long-term storage for further sensor calibration and open data access based on request. At the edge layer, the input API is generic and can support all types of environmental data. Then the data is collected and stored as raw data in json format in order to enable easy readability and post-processing and analyses.

Megasense data management platform primarily relies on two main software frameworks, the NGINX[1] and Java Spring Framework (JSF)[2]. The NGINX is used for actualizing two main objectives, data routing and load-balancing respectively. The data collection application is designed as both a web app and mobile app using the JSF, which handles specification of endpoints, automatic parsing of data, data transport security and rate limiting. When the data are collected using the JSF, the application gives tags each data as either for storage or for processing. Data intended for **storage** are tagged as *write data* (using https.../write/) while those intended for **processing** and **visualization** are tagged as *read data* (using https.../read/) using the REST API endpoints. These data are then received

[1] https://www.nginx.com/

[2] https://spring.io/projects/spring-framework

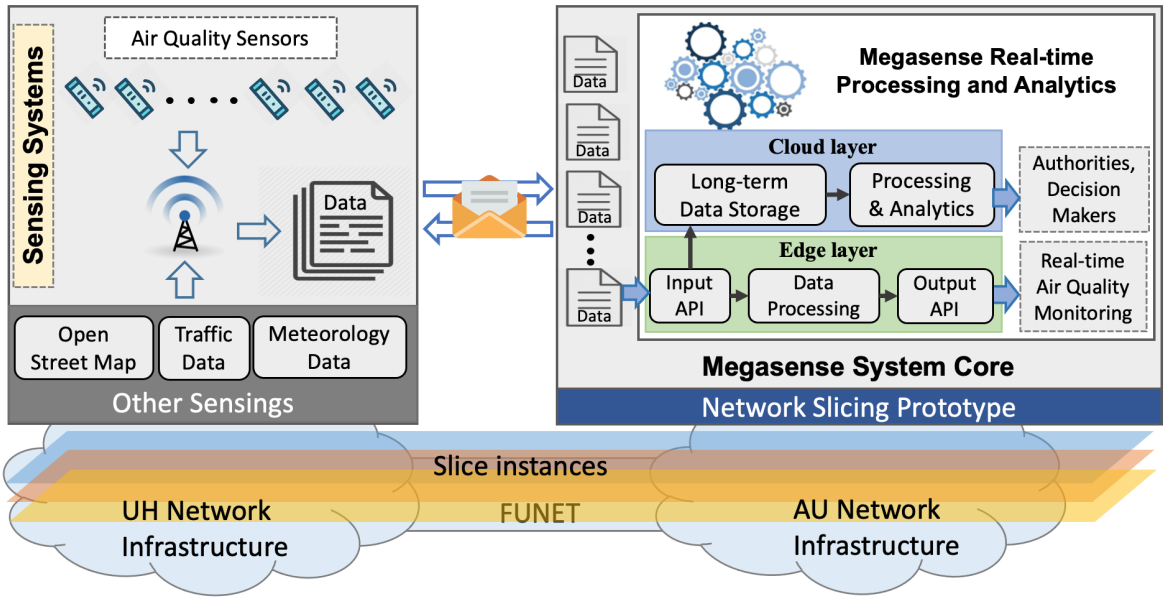



Fig. 2. Testbed architecture: Megasense system which is deployed on a network slice and runs in an end-to-end manner between UH and AU cloud facilities.

at the core of the Megasense platform through the Input API component where the NGINX application runs for correct routing of the incoming data to the appropriate component. Based on Figure 2, a *write data* is routed to the **long-term data storage** component while a *read data* is routed to the **data processing** unit. NGINX also does the load-balancing functionality by distributing the requests to be handled in a thread pool, which is dimensioned on the number of cores available in the system.

### B. Sensor Deployment

Currently, the Megasense platform supports a real deployment of more than 100 mobile air quality sensors, 20 fixed sensors and several test sensors. The measurement accuracy of these sensors which are tested at laboratory environments [6] as well as through co-locations located at city air quality monitoring stations, all operate in Helsinki, Finland [5]. The Megasense mobile sensors which their design is shown in [6] are portable sensor devices that can be carried by citizens. These sensors are designed to offer more accurate information about personal exposure to air pollutants. These sensor devices are capable of measuring meteorological variables including relative humidity, temperature and wind speed; particulate matter concentrations $PM_{2.5}$ and $PM_{10}$, and gaseous pollutants $NO_2$, CO and $O_3$. For data transmission, the sensor devices can connect to smartphones. This allows determining the location of the measurement and whether the sensor was used indoors or outdoors. The test sensors also are designed so that in addition to the particulate matters, they can measure volatile organic compound (VOC). The test sensors are designed so that they can be used for indoor measurements such as inside buildings and transportation systems.

The fixed sensors which are deployed in fixed locations at bus stops and roadsides are equipped with NB-IoT for data transmission. These sensors capture meteorological variables, particulate matter concentrations, and various gaseous pollutants. The NB-IoT sensors use Quectel BC95-B20 chip based modems, where sim cards from the public network of Elisa (one of the main network operators in Finland) are used for NB-IoT connections. The sensors periodically send their measurements over HTTP to Megasense data management platform. Due to the power requirements of various sensors and the environments in which the sensors are deployed, the frequencies of data transmissions of the sensors are different. Indeed, NB-IoT sensors installed in outdoor areas, Wi-Fi based sensors are located indoors, while mobile sensors carried by citizens which use Bluetooth to connect to the user's mobile phones, while users' mobile phones transmit the measurements to Megasense backend server through cellular networks. However, users' mobile phones can connect Internet for transmitting their data whenever needed and when cellular network is in outage or not available. In addition, smartphone application (Android and iOS) is also needed to be installed on users' mobile phones. This application is responsible for recording sensor measurements locally and transmitting them to the Mgasense backend server for analytics. Currently, the iOS version of the application is in private beta, while application is publicly available on Google Play for Android devices.

Hence, the different sensor devices, based on the location which they are deployed and according to their sensing capabilities and their access to the power resources can transmit their measured data at different intervals. Whereas, NB-IoT sensors send their measurements every 30 seconds; Wi-Fi based sensors send their measurements every 60 seconds; and the mobile sensors transmit their measurements every 30 seconds (due to their limited power sources). Moreover, as for physical real-life implication, for example, our sensor deployments may stimulate the development of various mobile apps such as personal exposure apps or green path applications presented in [6] to be used by users.

### C. Slicing Prototype

The network slice orchestration system enables the deployment and run of multiple network slices on top of the same physical infrastructure while ensuring security and resource isolation between the network slices. One of the network slices runs on top of our shared physical infrastructure is Megasense, which is an mMTC network slice. The Megasense platform is designed for collecting, processing and analysing data sourced from these sensors in order to offer real-time, accurate and fine-grained information about the air quality and enable new applications [6]. Therefore, in order to demonstrate the performance of a massive number of air quality sensors over network slices, we use the Megasense framework of UH and deploy them on a Slicing framework that is developed at AU, both in Finland. The overall architecture of Megasense deployment over a network slice in an end-to-end manner is shown in Figure 2.

As depicted in Figure 2, both the sensor nodes that are deployed to sense the atmospheric air pollution and the Megasense application designed to collect and analyse the data are hosted on virtual resources between the cloud facilities of UH and AU in an end-to-end way. Despite the fact that the number of air quality sensor units are continuously increasing within Megasense, with current deployments around hundreds of sensors in Helsinki. Yet, this number does not translate to a massive scale. Hence, in order to test the tolerability of the network slicing prototype, we simulate an increasing number of sensors in Megasense. To this end, we developed a script that simulates an arbitrary number of sensors and executes it from the UH end of the slicing framework.

While the script needed to simulate the deployment of a massive number of sensor nodes, e.g., 10k, 20k or more was deployed on virtual resources consisting of 2 CPU cores and 2 GB memory. The resources necessary to operate the Megasense application that is provided from the network slicing framework was initially equipped with 4 cores of CPU and 4 GBs of memory. Depending on the deployment need, resources for the Megasense platform may be scaled either vertically up/down or horizontally up/down.

We developed the script such that each sensor sends it's measurements over HTTP every 30 seconds. We also implemented the script in Erlang and we modelled each sensor as an independent process, i.e., an actor in the actor concurrency model. In addition, the script uses non persistent TCP connections. The choice of using TCP connections is because $i)$ in reality, the sensor devices which are deployed in different places in a city do not use a shared TCP connection, and $ii)$ it is to avoid running out of file descriptors in the VM when the number of simulated sensors is high.

The slicing prototype is one which consists of multiple clusters of Infrastructure as a Service (IaaS) cloud facilities deployed on a datacentre, wherein, there are two controller nodes and three compute nodes each attached to of each controller nodes. In order to enforce quality of service for different slices, we have integrated the ONOS SDN controller in the setup which has SDN enabled switches. This way, the network layer 2 packet forwarding could be under the absolute control of the SDN controller. This SDN support brings flexibility and dynamicity to traffic steering for the network slices.

Next, in order to determine how well the end-to-end setup provides operational support for the smooth running of the Megasense platform in terms of its requirements, we focus on assessing the capabilities of a single instance of Megasense while running on a network slice and we took note of the slices virtual resources.

## IV. Simulation and Results

In our study, we aim to evaluate the performance of the Megasense platform when deployed over a network slice and understand its tolerance when increasing the number of air quality sensors that is connected to it. Megasense is designed to send an acknowledgement/reply with HTTP 200 status code when a measurement is received. The HTTP 200 status code refers to success status response code and indicates that the request has succeeded. In Megasense, when big number of measurements (i.e., it is overloaded) are received, the platform either replies $i)$ with a different HTTP status code or $ii)$ it does not send a reply at all, because the upload request might have been dropped.

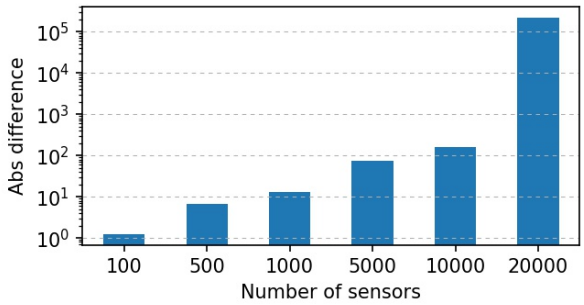
To evaluate the tolerance and performance of Megasense, we considered the following primary metrics that are needed to determine if Megasense is being overloaded or not: $i)$ the difference between the number of measurements sent by the sensors based on simulation and the number of measurements for which the script receives a reply (either positive or negative), and $ii)$ the percentage of measurements that is successfully collected by the Megasense platform, i.e., measurements for which the script has received a positive reply (200 as HTTP status code) when operating over a network slice.

In order to properly identify the cause of an overload on Megasense, we investigate its resource consumption by collecting the metrics of CPU usage, memory usage, disk utilization and bandwidth usage. We utilize the SAR tool[3] to investigate the Megasense resource consumption. In our measurements, we simulated an increasing number of sensors from $100, 500, 1000, 5000, 10K,$ all the way to $20K$ sensor nodes. The measurements pertaining to each increase in the number of sensors i.e., 100, 500 and so on, is carried out for a total duration 10min for each test and repeated for a total of 20 times, while taking the average values. In the following subsections, we explain the results of our performance evaluation.

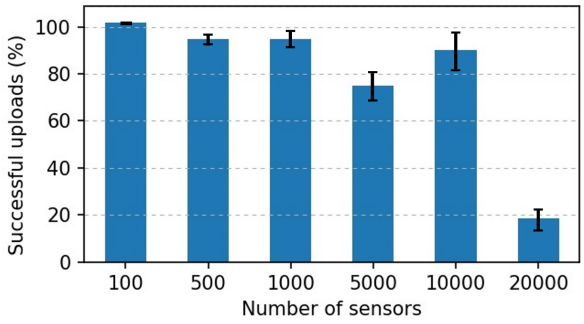
### A. Identifying overload

To study the overloading of Megasense, we increased the number of sensors in our simulations. Figure 3 illustrates these results. Whereas, Figure 3(a) shows the absolute value of the difference between the expected number of replies from Megasense and the average number of received replies (i.e.,

[3]https://linux.die.net/man/1/sar




(a) The absolute difference between the number of expected and the average number of received replies.




(b) The percentage of successfully received replies.

Fig. 3. Megasense performance when increasing the number of sensors.

$Expected_{Received-Replies} - Average_{Received-Replies}$). The number of expected replies is computed when each sensor sends a measurement every 30 seconds (i.e., 2 replies in 1min) and a session lasts 10 minutes. For example, when we consider $10K$ sensors, the expected number of replies is $10K \times 2 \times 10 = 200K$ replies per test.

In Figure 3(a), we observe that the difference in magnitude at each increase in the number of sensors is almost similar. This result shows that the performance of Megasense is good until the number of sensors increases to $10K$. As illustrated in Figure 3(a), the difference in magnitude between $10K$ and $20K$ sensors is much higher than others i.e., between $10K$ and $5K$, $5K$ and $1K$, $1K$ and 500, and 500 and 100. This result indicates that when the number of sensors exceeds $10K$, Megasense exhibit overload behaviour since it is not able to send replies to a significant number of upload requests coming from the sensors.

Beside the difference in the number of received replies, we also measured the percentage of successful received replies. As shown in Figure 3(b), a reduction happens in the percentage of successful replies when the number of sensors increases. This is due to the sudden and massive increase of the requests which puts pressure on the Megasense platform.

As illustrated in the figure, the performance of Megasense significantly deteriorates when the number of sensors exceeds $10K$. The number of successful replies drastically decreases when the number of sensors reaches $20K$ (a massive $10K$ increase) with our current settings in the testbed. However, with the limits that we have imposed on our testbed by means of using a single VM, data transmission frequency (i.e., transmission per 30 seconds) and considering the available resources such as CPU, memory, disk and network interface, we can conclude that $10K$ sensors is an acceptable number for Megasense that shows a high performance.

### B. Resource consumption

In addition to overload of Megasense, we evaluate the resource consumption of Megasense by means of CPU usage, memory usage, Megasense disk utilization and the bandwidth consumption when increasing the number of sensors. The results of our performance evaluation are shown in Figure 4.

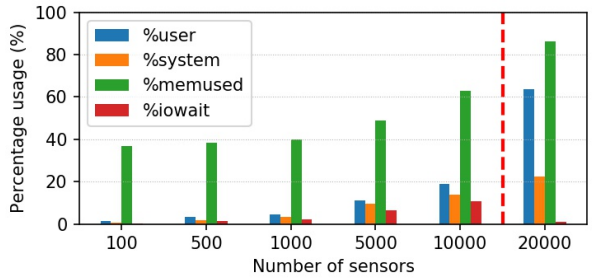
In principle, the CPU usage includes the user applications usage (%user) and system operations usage (%system). Whereas, the sum of %user and %system defines the CPU load on the system. The Megasense system is prepared to handle a lot of incoming network connections and process a massive amount of disk I/O operations (from the connected sensor nodes), hence, it is natural for the system to be heavy on CPU usage.

As depicted in Figure 4(a), when the number of sensors increase up to $10K$, both %user and %system usages show slight and normal increase of a little below 20%. We can also observe that, as expected the memory consumption (%memused) shows a logical increase of slightly above 60% with the increase of the number of sensors. In Figure 4(a), we observe that the I/O waiting time (%iowait) steadily increases with the increasing number of sensors. %iowait indicates the percentage of time that a CPU is busy during which the system has an outstanding disk I/O request.

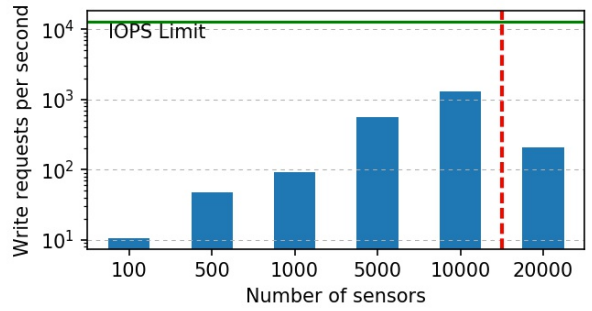
The (%iowait) exhibit slightly steady increase when the number of sensors increases from 500 to $10K$. However, when the number of sensors increased from $10k$ to $20k$, the %iowait almost vanished. This happens because Megasense fails to store the sensor measurements, due to the massive increase in the amount of upload request coming from the sensor nodes, thereby resulting in much lower %iowait and the high rate of unsuccessful uploads. The sudden surge in the %user also corroborates this behavior.

When the number of sensors exceeds $10K$, both CPU and memory usage demonstrate significant increase. Since the Megasense system deals with disk read/write and network operations, the CPU usage is high. However, to further realize the reason behind this increase, we investigated the logs of Megasense and found out that the problem is sourced from the lack of enough memory. That was reported by *"the application throws OutOfMemory exceptions"*. Actually, the reason for high CPU usage would be $i$) that the VM does not have swap space, i.e., the system is forced to discard memory pages when Megasense is approaching the memory limit, thus deteriorating the performance; or $ii$) since the *"OutOfMemory"* exceptions are uncaught but the system keeps running anyway.

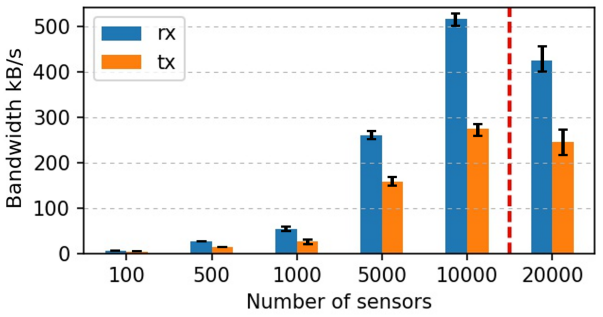
Moreover, the results in Figure 4(b) show that the number of write requests per second is relatively low compared to what the disk in the VM can perform. The disk limit in VM is shown with green line in Figure 4(b). The write request indicates the request for storing the data disk of Megasene




(a) The percentage consumption of different system metrics when increasing number of sensors.




(b) The number of write requests on the disk per second.




(c) The bandwidth consumption.

Fig. 4. Megasense performance results.

system. To investigate the capabilities of the Megasense disk, we performed tests with the fio utility[4]. The results show that for each number of sensors the write requests are much below the limit line. This proves that the disk in VM has capacity to host big number of I/O operations. This result enables us to conclude that disk capacity is not the factor that limits the performance of Megasense when running on this amount of virtual resources.

In addition, when the number of sensors increases to $20K$, the I/O operations considerably decreases. This is shown by a dashed red line separating this result with others in Figure 4(b). This translates to the overload situation of Megasense and it's inability to cope with the massive workload when the number of sensors exceeds $10K$. Figure 4(c) depicts that in Megasense, the bandwidth consumption is not a critical concern in air sensing use case. The reason would rely on the small data file sizes of sensing. Indeed, when the number of sensors increases to $10K$, the maximum utilized bandwidth reaches to around $500KB/s$ in download (shown by rx). Similar to the disk usage, when the number of sensors increases to $20K$, due to the overload situation of Megasense the bandwidth consumption decreases compared to $10K$ sensors.

[4]https://linux.die.net/man/1/fio

### C. Real sensor deployment in smart cities

Our performance evaluation results showed that Megasense when running in a VM with $4$ cores and $4$ GB of memory supports around $10K$ sensors ($\mathcal{S}$). In reality, to deploy the sensors in smart cities, here, we consider two scenarios of Dense network and Ultra dense network deployment. We use parameters and values presented in [3], as presented in Table I to roughly estimate the capabilities of the Megasense when used for real massive sensor deployments. Based on [3], a real-life mega-city coarse-grained sensors deployment are usually characterized with the placement of a few dozen expensive measurement towers for carrying out accurate and effective air quality measurement and hundreds of other less expensive ones.

TABLE I
BACK-OF-THE-ENVELOPE COMPUTATIONS FOR MEGASENSE CAPABILITIES.

| Considered factor | Dense network | Ultra dense network |
|---|---|---|
| Density ($\mathcal{S}$/km$^2$) | 400 $\mathcal{S}$/km$^2$ | 10K $\mathcal{S}$/km$^2$ |
| Coverage single VM (km$^2$) | 25 km$^2$ | 1 km$^2$ |
| Helsinki area (213.8 km$^2$) | 9 VMs | 214 VMs |

As presented in Table I, the dense network scenario is defined to have $400$ sensors per km$^2$, while the ultra-dense network scenario has $10K$ sensors per km$^2$. If we consider Megasense which runs on a single VM similar to our experiment, a single instance of Megasense is enough to cover 25 km$^2$ in the dense network scenario and 1 km$^2$ in the ultra dense network scenario.

Considering the case of dense network (dense deployment of sensors) in the city of Helsinki, which is 213.8 km$^2$, 9 instance of Megasense (9 VMs) will be needed. While, in the case of ultra-dense network scenario, 214 VMs is needed. Fortunately, in both scenarios, obtaining these numbers of VMs is fairly easy from any datacenter or any edge computing resource [15]. As a result, even though the memory was the main concern limiting the performance of Megasense, yet it supports a high number of sensors when running in a VM with $4$ cores and $4$ GB of memory based on our system simulation setup. Therefore, we can assume that Megasense is an efficient platform which promotes and facilitates the way towards 5G-enabled smart cities. Although, the system deployment and results presented in this work has been largely motivated by the Nokia white paper [3], which anticipates a real sensor network deployment for a specific sensing use case (air quality sensing system), we are satisfied with the realised results. As a future work, we are interested in studying the impact of a massive deployment of sensors in an extreme case considering the 5G requirement which indicates 1M devices/km$^2$ in a more generic perspective of sensors deployment.

## V. Conclusion

In this work, we have presented the feasibility of deploying a smart city platform application on network slices. Most importantly, owing to the peculiarity of the sensors' platform communications pattern, we have shown that deploying tens of thousands of such IoT device nodes for data collection and analysis through the Megasense platform is very possible over network slices. Based on our performance evaluation results, it is observed that the system requirements of sensors platforms such as the Megasense can be adequately achieved while operating on network slices. Also, deploying such a system over network slices ensures an optimal resource utilization at every point in time throughout the operation life cycle of the system due to the flexibility and elasticity of network slices in resource utilization.

## Acknowledgement

This work is supported by the Finnish funding agency for research, Business Finland under the 5G Finnish Open Research Collaboration Ecosystem (5GFORCE) project as part of the 5G Test Network Finland (5GTNF). The research was supported in part by Nokia Center for Advanced Research (NCAR) and Healthy Outdoor Premises for Everyone project (UIA03-240). It was also supported in part by Megasense project with grant number 324576, and the Academy of Finland projects: 6Genesis under grant number 318927 and IDEA-MILL with grant numbers 335934 and 335936.

## References

[1] 3GPP, "Feasibility Study on New Services and Markets Technology Enablers," 3rd Generation Partnership Project, Tech. Rep., September 2016.

[2] T. Taleb, I. Afolabi, and M. Bagaa, "Orchestrating 5G Network Slices to Support Industrial Internet and to Shape Next-Generation Smart Factories," *IEEE Network*, vol. 33, no. 4, pp. 146–154, 2019.

[3] Nokia, "The Outlook for Intelligent Air Quality Monitoring in Megacities as a 5G Service," Nokia Corporation, Tech. Rep., 2018. [Online]. Available: https://onestore.nokia.com/asset/202208

[4] I. Afolabi, T. Taleb, K. Samdanis, A. Ksentini, and H. Flinck, "Network Slicing and Softwarization: A Survey on Principles, Enabling Technologies, and Solutions," *IEEE Communications Surveys Tutorials*, vol. 20, no. 3, pp. 2429–2453, 2018.

[5] N. H. Motlagh, E. Lagerspetz, P. Nurmi, X. Li, S. Varjonen, J. Mineraud, M. Siekkinen, A. Rebeiro-Hargrave, T. Hussein, T. Petäjä, M. Kulmala, and S. Tarkoma, "Toward Massive Scale Air Quality Monitoring," *IEEE Communications Magazine*, vol. 58, no. 2, pp. 54–59, 2020.

[6] A. Rebeiro-Hargrave, N. H. Motlagh, S. Varjonen, E. Lagerspetz, P. Nurmi, and S. Tarkoma, "MegaSense: Cyber-Physical System for Real-time Urban Air Quality Monitoring," in *2020 15th IEEE Conference on Industrial Electronics and Applications (ICIEA)*, 2020, pp. 1–6.

[7] S. Horsmanheimo, J. Säe, T. Jokela, L. Tuomimäki, E. Nigussie, A. Hjelt, S. Huilla, T. Dönmez, N. L. Bail, and M. Valkama, "Remote Monitoring of IoT Sensors and Communication Link Quality in Multisite mMTC Testbed," in *2019 IEEE 30th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, 2019, pp. 1–7.

[8] K. Mikhaylov, V. Petrov, R. Gupta, M. A. Lema, O. Galinina, S. Andreev, Y. Koucheryavy, M. Valkama, A. Pouttu, and M. Dohler, "Energy Efficiency of Multi-Radio Massive Machine-Type Communication (MR-MMTC): Applications, Challenges, and Solutions," *IEEE Communications Magazine*, vol. 57, no. 6, pp. 100–106, 2019.

[9] V. Windha M., Iskandar, Hendrawan, and M. S. Arifianto, "Wireless Sensor Network on 5G Network," in *2018 4th International Conference on Wireless and Telematics (ICWT)*, 2018, pp. 1–5.

[10] R. Giuliano, F. Mazzenga, and A. Vizzarri, "Satellite-Based Capillary 5G-mMTC Networks for Environmental Applications," *IEEE Aerospace and Electronic Systems Magazine*, vol. 34, no. 10, pp. 40–48, 2019.

[11] M. Kamel, W. Hamouda, and A. Youssef, "Uplink Coverage and Capacity Analysis of mMTC in Ultra-Dense Networks," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 1, pp. 746–759, 2020.

[12] C. Delgado, M. Canales, J. Ortín, J. R. Gállego, A. Redondi, S. Bousnina, and M. Cesana, "Joint Application Admission Control and Network Slicing in Virtual Sensor Networks," *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 28–43, 2018.

[13] P. Jaszkowski, P. Sienkowski, and K. Iwanicki, "Decentralized Slicing in Mobile Low-Power Wireless Networks," in *2016 International Conference on Distributed Computing in Sensor Systems (DCOSS)*, 2016, pp. 177–186.

[14] Y. Lin, W. Dong, and Y. Chen, "Calibrating Low-Cost Sensors by a Two-Phase Learning Approach for Urban Air Quality Measurement," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 2, no. 1, pp. 1–18, 2018.

[15] X. Su, X. Liu, N. H. Motlagh, J. Cao, P. Su, P. Pellikka, Y. Liu, T. Petäjä, M. Kulmala, P. Hui, and S. Tarkoma, "Intelligent and Scalable Air Quality Monitoring with 5G Edge," *IEEE Internet Computing*, vol. 25, no. 2, pp. 35–44, 2021.

## Biographies

**Naser Hossein Motlagh** is a Postdoctoral Researcher at the Department of Computer Science, University of Helsinki. He received his D.Sc. in Networking Technology from the School of Electrical Engineering, Aalto University, Finland in 2018. His research interests include Internet of Things, wireless sensor networks, smart buildings, unmanned aerial and unmanned underwater vehicles (UAVs & AUVs). Contact him at naser.motlagh@helsinki.fi

**Ibrahim Afolabi** obtained his Master's degree from the School of Electrical Engineering, Aalto University, Finland in 2017. He is presently pursuing his doctoral degree at the same university. His research interests include network slicing, cloud computing, machine learning, MEC, network softwerization, NFV, SDN, and dynamic network resource allocation. Contact him at ibrahim.afolabi@aalto.fi

**Matteo Pozza** received the bachelor's and master's degrees in computer science from the University of Padua, Italy, in 2014 and 2016, respectively, and the Ph.D. degree from the University of Helsinki, in 2020. His research interests include theoretical and practical problems in networked systems, especially mobile networks. Contact him at matteo.pozza@helsinki.fi

**Miloud Bagaa** received his Ph.D. degree from the University of Science and Technology, Houari Boumediene Algiers, Algeria, in 2014. He is currently a senior researcher with the Communications and Networking Department, Aalto University. His research interests include wireless sensor networks, the Internet of Things, 5G wireless communication, security, and networking modeling. Contact him at miloud.bagaa@aalto.fi

**Tarik Taleb** is professor at Aalto University and the University of Oulu. He received his Ph.D. degree in information sciences from Tohoku University, Japan in 2005, where he worked as an assistant professor. He is the founder and director of the MOSA!C Lab (www.mosaic-lab.org). He was also a senior researcher and 3GPP standards expert at NEC Europe Ltd., Germany. His research interests include network function virtualization, network softwarization and software defined networking. Contact him at tarik.taleb@aalto.fi

**Sasu Tarkoma** is a Professor of computer science with the University of Helsinki and Head of the Department of Computer Science. He is a visiting professor with the 6G Flagship at the University of Oulu. He completed his Ph.D. in Computer Science at the University of Helsinki in 2006. He has authored four textbooks and has published over 250 scientific articles. He holds ten granted U.S. patents. His research interests include Internet technology, distributed systems, data analytics, and mobile and ubiquitous computing. Contact him at sasu.tarkoma@helsinki.fi

**Hannu Flinck** received the M.Sc. and Lic.Tech. degrees in Computer Science and Communication Systems from Aalto University in 1986 and 1993, respectively. He was with the Nokia Research Center and the Technology and Innovation Unit of Nokia Networks. Currently, he is a research manager with Nokia Bell Labs, Espoo, Finland. His research interests include mobile edge computing, SDN, and content delivery in mobile networks, particularly in 5G networks. Contact him at hannu.flinck@nokia-bell-labs.com