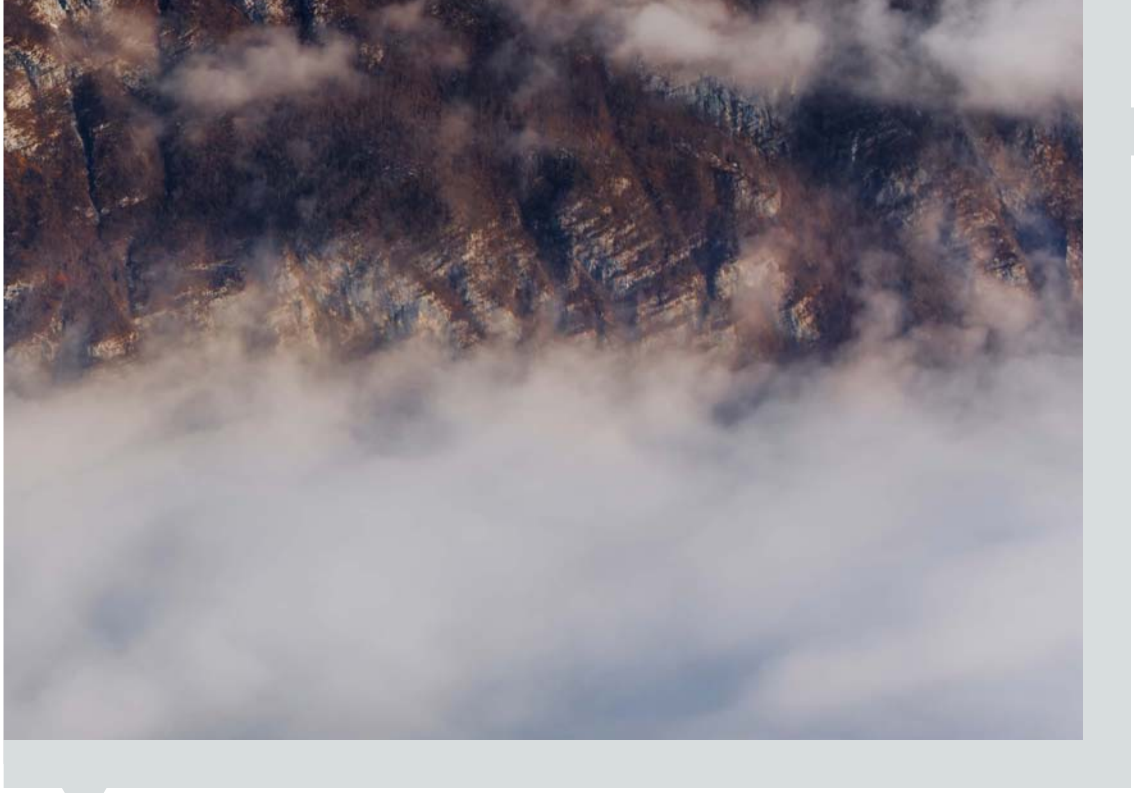

A Cooperative Fog Approach for Effective Workload Balancing

Andreas Kapsalis, Panagiotis Kasnesis, Iakovos S. Venieris, and Dimitra I. Kaklamani, National Technical University of Athens
Charalampos Z. Patrikakis, Piraeus University of Applied Science

Fog computing is an emerging paradigm, suitable to serve the particular needs of IoT networks. We present a fog architecture, which diverges from the traditional hierarchical and centralized fog model, and adopts a cooperative model, which allows for a federation of edge networks.

Smart sensor networks are considered characteristic examples, where the Internet of Things (IoT) can act as an enabling framework, allowing the efficient exchange of information between sensors and controllers. On the other hand, cloud computing platforms and infrastructures provide the ideal virtual environment for analysis of the collected data, giving the ability to provide computational resources. This appoints cloud computing platforms as the perfect environment to support IoT computational needs.

The explosive rate of the evolution of IoT, creates increasing demands for processing capabilities, which is translated in a need for deploying a large number of devices, in order to serve tasks cognitively for strict computational purposes, improving QoS, latency and efficiency. This new paradigm often referred to as fog computing, enables low-latency computation of tasks in close range networks (edge), which incorporate various types of devices. Consequently, the core concept of fog computing is in complete accordance to the concept of IoT, enabling smart, connected devices not only to operate inside a collective and collaborative ecosystem, but also to be a structural part of it.

The traditional idea of fog computing was first introduced by Bonomi et al.[1] and identifies the need for low latency edge networks. These networks assume the role of executing short computational tasks for various IoT use cases. Tasks and data are handled in the intermediate fog layer. Moreover, the characteristics of fog computing[1] are:

- low latency and location awareness
- wide-spread geographical distribution
- mobility
- very large number of nodes
- predominant role of wireless access
- strong presence of streaming and real-time applications
- heterogeneity

These works introduce the fog computing paradigm and it's potential. However, fog hosts are not as computationally powerful as the ones offered by massively distributed cloud infrastructures. Load balancing techniques, including resource management methods, messaging architectures and smart task profiling for different scenarios, are still open issues for fog computing. In this article, we focus on these issues, and present a novel architecture of a fog-enabled platform that can be used by IoT networks to perform complex and demanding computational tasks. Smart devices are connected to specific message topics, via the publish-subscribe (Pub/Sub) pattern and the Message Queue Telemetry Transport (MQTT)[2] connectivity protocol, which enables them to send and receive MQTT compliant messages asynchronously. These smart devices also have the ability to send a small piece of code that implements the set of collected data. When received by the platform, this small piece of code is forwarded to the most suitable host for execution, whether is a VM on the Cloud or a device on the Edge of the network.

Our platform diverges from the traditional cloud and fog approach in a number of ways. A Cloud Platform plays the part of a centralized point of access for IoT platforms. Thus, resource management is targeted towards ensuring that multiple data streams from several IoT networks are satisfied, which can lead to lower QoS in terms of how efficiently IoT workloads are processed. The fog computing paradigm came as a solution to that problem, by providing on the Edge computation capabilities. Our platform tries to improve that approach as well by utilizing distributed communication models (Pub/Sub) and standardizing messages in order to create a cooperative fog layer, where multiple devices and networks can be involved. To ensure that QoS is maintained at optimal levels, our platform uses an allocation mechanism that captures the characteristics of the system and selects the most appropriate host in a near greedy fashion. First, we introduce a fog architecture, capable of serving the computational needs of tasks by utilizing both stable and mobile devices. Next, we propose fog message, a specific MQTT message format, which contains all the necessary information for task allocation. Finally, we evaluate the Workload Balancer module and the incorporated cost function, by taking into consideration different scenarios.

Related Work

Salman et al., describes a fog computing architecture for integrating Mobile Edge Computing (MEC) with

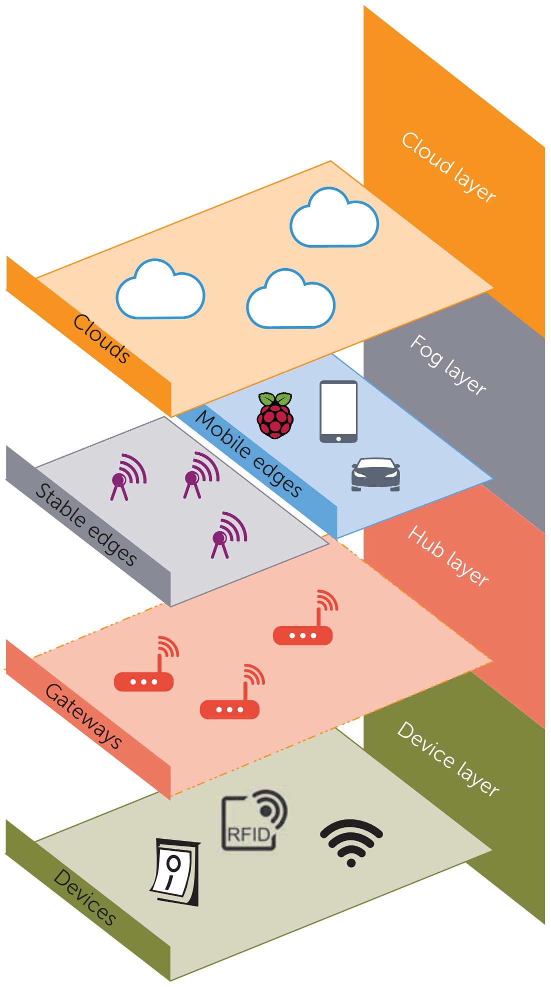

FIGURE 1. Fog platform architecture.

IoT and SDNs (Software Defined Networks)[3]. The examined architecture highlights Device, Network, Control and Application Layers that undertake different tasks for the orchestration of the offered fog services. However, this work only presents some conceptual information regarding fog architectures and for a particular use case. Another SDN-based fog computing approach is proposed, a "fog node", which is an edge switch integrated with the SDN-Controller and is capable of acting as an MQTT broker, serving as a platform for data analytics and communicating with other "fog nodes" or with the end-host broker server[4].

Aazam et al. provide a detailed model for resource management in the fog[5]. They focus on resource prediction and reservation by providing a complete algorithm model in the fog layer (Fog Micro Datacenter). The proposed resource management is mainly based on prediction by analyzing the user behavior, without classifying tasks and hosts depending on their network and computational characteristics.

Another self-organized fog platform,[6] has the authors introducing the term Fog of Things (FoT). FoT defines IoT services in the network edge and distributes them through a message and service oriented middleware. This method is empowered with the FoT-Profile, which defines what type of IoT service is offered by a FoT node. Despite the fact that FoT is a distributed paradigm, it is a "fixed" approach. According to our fog platform the edge devices do not have service profiles, but they can offer any service as long as they have the processing capability to do that and they have informed the mediator edge device.

A similar approach to ours is proposed for IoT resource management and domain management as described by Li et al,.[7] however it is cloud based and does not utilize Edge Networks. Moreover, Aazam et al.[8] use a Smart Gateway for the data processing management. In contrast with the fog computing architecture presented in this article, the gateway is a mandatory component responsible for making decisions (i.e. sending data on cloud or at the edge device) and preprocessing the IoT data.

Another fog platform, capable of serving heterogeneous IoT devices, is introduced in *Fog computing: A platform for internet of things and analytics*.[9] A special software agent, called *Foglet* agent, is responsible for orchestrating the services to be executed. However, the use of mobile devices in this platform remains unclear. This issue has been investigated in detail, and an Ad Hoc Cloudlet is introduced. Chen et al. focuses in the network constraints that mobile devices face and not on the interoperability and service allocation issues that are produced when dealing with IoT service orchestration.[10]

Fog Platform Architecture

The proposed fog computing platform enables the allocation and management on the set of computational resources for executing effectively IoT tasks. It consists of a *Device Layer*, *Hub Layer*, *Fog Layer*, and *Cloud Layer*. Figure 1 shows the layers and the entities of the proposed fog platform architecture.

Device Layer

The lowest layer of our platform architecture is the *Device Layer* (*DL*). It contains physical devices that have low computational power and storage memory. Such devices can be sensors, actuators or electronic tags, which are incapable of executing complicated tasks, thus they are connected to the upper layers of the proposed platform in order to get their tasks executed or their data stored. For example, if an actuator needs to make a decision based on complex event processing on data produced by sensors, it gets his task executed by the upper layers and afterwards gets informed for the decision it should make.

Hub Layer

The *Hub Layer* (*HL*) is the second layer of the fog platform and consists of gateways. In cases where a device uses communication protocols (e.g. ZigBee, Bluetooth, Z-Wave) incapable of exchanging messages over MQTT, the *HL* is responsible for creating fog messages and forwarding them to the *Fog Layer* acting as mediator. However, devices capable of communicating directly with the *Fog Layer* (e.g. MQTT enabled devices) do not use these communication nodes.

Fog Layer

The *Fog Layer* (FL) contains the computing edges that operate in a cooperative manner to execute tasks. In our approach, Stable Edges (e.g. a server near the devices) contain the Fog Broker which is responsible for enriching the messages they receive from the lower layers and for the task management and allocation. Moreover, they preserve a list of all the subscribed entities (Cloud providers, close range/mobile edges).

Workload Balancer

In order to better utilize the connecting Fog Networks, the Fog Broker needs to allocate incoming tasks accordingly. A greedy algorithm would allocate the task into the first valid host. However, this approach does not take into account certain host and network characteristics. On the other hand, advanced allocation mechanisms do not suit the purpose of the fog paradigm, as they take a long time to converge to a near-optimal solution. The Workload Balancer incorporates a simple and fast score based procedure that allows the Fog Broker to quickly sort the available hosts and choose the most suitable one. The algorithm applies a score function with the appropriate weights to three key characteristics of each host: its current utilization, its latency and its battery state (applicable on mobile devices). Furthermore, having the advantage of knowing beforehand the required resources of each task (Section IV), the Workload Balancer can decide whether the task can be directly forwarded to the Cloud provider. In this case, the Workload Balancer does not apply the score based function, but forwards the task directly to the Cloud Layer.

Cooperative Fog Networks

Mobile or close range edges register to the fog platform offering their computational and hardware resources in an altruistic way. There are many models which could be adopted here to offer incentives for the participation of devices. However, it is not in the scope of this article to examine the particular models, but rather to focus on the management of resources in the fog. What is more, their connection time with the platform may be limited. For example, a car passing near by a stable edge will be reachable only for a few seconds. After getting a message from the stable edge, the mobile edge node parses it, executes the contained script (Section IV) and sends the results to the network address of a node that belongs to a lower layer and a verification message to the stable edge. The latter part has to be done to inform the stable edge that the task has been executed.

Cloud Layer

The last layer of the fog platform consists of the *Cloud Layer* (CL). When an IoT task is not time-sensitive or it requires a guaranteed execution environment it is allocated to the Cloud Layer. The Cloud Layer is assumed to be used for batch (offline) analysis and longer-term storage of data instead of online data processing. Nevertheless, in cases where the real-time processing demands are high, the Workload Balancer may forward the Fog Message to the CL. Cloud resources are available to the fog platform in the same way as mobile or close range Edge resources are.

Fog Message

In the proposed platform the exchange of messages is done in an asynchronous way. Specifically, it follows the Pub/Sub messaging pattern.[11] According to this pattern, the data providers are publishers and the data consumers are subscribers. Publishers send their data resources to a mediator component, called broker, which updates the data collection that is related to that specific resource, called topic. Subscribers that want to be informed about these updates, subscribe to this specific topic. We selected MQTT that follows the Pub/Sub pattern to provide the ideal solution as a "lightweight" and low overhead communication protocol, in order to support networks where connectivity isn't guaranteed, or the subscribed devices are low end with limited capabilities.

Moreover, one of the novel functionalities of the proposed cooperative fog platform is the support for mobile devices. This can pose a problem, due to the uncertainty of the device's availability. MQTT provides the functionality to certain types of devices

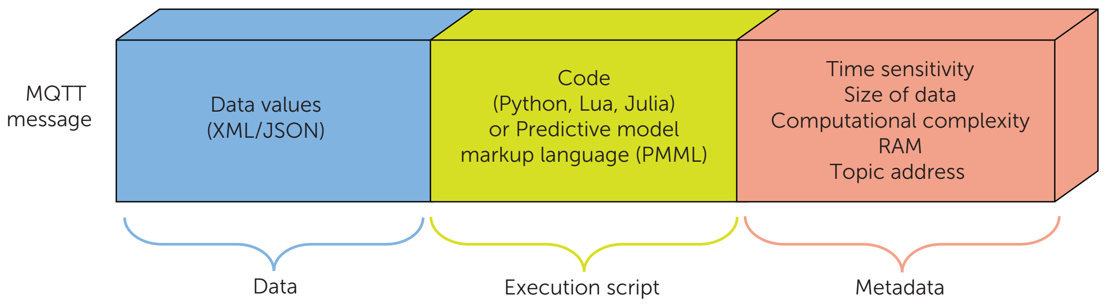

FIGURE 2. The Fog Message

to achieve a specific type of connection with the MQTT broker. Thus, when a client becomes unavailable during the execution of a task the system can be notified quickly and resend the task to another host. In general MQTT provides the following session options:

- *QoS*: QoS 0 is recommended for subscribers that can maintain a stable connection with the MQTT broker, as only the main publish messages are sent. In the case of mobile devices the proposed platform enforces QoS 1, where the MQTT broker keeps sending a publish message to the subscriber until it receives an acknowledgement message.
- *cleanSession*: When set to false this flag indicates that a session between the client and the broker is persistent and topic messages are stored within a queue that is maintained at the MQTT broker. This way when a client reconnects, it can receive messages that were published in the topic during its absence. In our case this is particularly helpful as a reconnected client can be notified to pause or resume an allocated task or whether this task is allocated to another host or not.
- *keepAlive*: This flag indicates the interval during which a client has to send a ping request notifying the MQTT broker that is still active. Otherwise, the MQTT broker closes the connection with the client. By assigning a lower value to this flag, the Fog Broker can be notified quicker whether the device is not active, remove it from the subscribers list and thus ensure that it will not falsely allocate a task to that host.

A typical MQTT message (i.e., message that contains only data values) cannot serve our architecture. As mentioned, the stable edge collects all the messages and allocates the IoT tasks using the Workload Balancer of the Fog Broker. The Workload Balancer needs to have some features as inputs, which are not provided by standard MQTT messages. Furthermore, the mobile/close range edges that are going to receive this data need to be aware of the data analysis algorithm and the network address they will send the results. Consequently, we introduce the Fog Message (FM) to fulfill our platforms needs. The FM (see Figure 2) contains these three types of information

- data
- an execution script
- metadata .

It should be noted that the FM is created in the *DL*, if the devices are MQTT enabled, or in the *HL*. In either case, the stable edge will be the receiver-subscriber of the message.

Data

The data part of the FM contains all the data produced by the entities of the *DL*. These data resources need processing, thus they are sent to the *FL* and consequently (e.g. long-term data storage) to the *CL*.

Execution Script

The execution script contains the code that the selected host will execute and produce the results. These lines of code are provided by the manufacturer of the device and are written in a lightweight scripting programming language (e.g. Lua, Python, Julia). What is more, Predictive Model Markup Language (PMML) could be used in order to define an XML formatted predictive model produced by data mining and machine learning algorithms. Of course, subscribed hosts that will eventually receive the FM need to have already installed software capable of parsing and executing the code.

Metadata

The final part of the FM is the metadata. Metadata describe the needs of the IoT task and are, also, defined by the manufacturer of the device or the system (i.e. native or third party applications defined in the *HL*). They contain the following five values:

- *Time Sensitivity*, a flag that defines whether a task should preferably be allocated to a fog layer host.
- *Size of Data* represents the total size of the provided Data Values.
- *Computational Complexity* expresses how computationally intensive is a task (i.e., the Code). The Big O notation was selected for measuring the task complexity, as it ignores low-level details (e.g., programming language and is capable of predicting how an algorithm/task behaves as the data input grows larger.
- *RAM* defines the amount of system memory needed to import the necessary libraries and maintain the necessary data structures (e.g., arrays, tables etc.) that are defined in the Execution Script.
- *Topic Address* defines the network address for the topic that the results should be published.

The first four values are given as input to the Workload Balancer at the Fog Broker, while the last one guides the task executioner to which address it should send his results.

Platform Evaluation

To evaluate the proposed architecture we conducted a series of simulations, to study the performance of our proposed cooperative platform against traditional architectural approaches. Simulations were performed with the Cloudsim[12] toolkit and a set of extensions to allow for random arrival rate of tasks.[13]

The main reason for choosing a simulator is that it allowed us to test the performance of our platform on extreme cases and large scales dealing with thousands of tasks and multiple hosts. In addition, it gave us measurable results that have shown how our platform can be compared against widely used scenarios.

As stated in Section III, the Workload Balancer applies a score based function in order to decide which host is more suitable for each task. Weights for the score function were dynamically calculated in each run. In order to determine each weight the algorithm calculated the standard deviation of each category vector (utilization, latency and battery). Effectively the score function is presented as follows:

$$S_i = \sigma_U\, u_i + \sigma_L\, 1/l_i + \sigma_B\, b_i, \;\; l_i > 0 \tag{1}$$

Where u_i, l_i and b_i denote the utilization, latency and battery level of each host, while σ_U, σ_L *and* σ_B denote the standard deviation for each category vector. This approach allows for the system to capture changes during the course of the simulation. For instance, if all hosts have similar resource utilization the score function will apply greater weight for latencies and vice versa.

Simulation Configuration

Time in the Cloudsim simulator is measured in abstract time units. In addition, the simulator measures computational capacity of hosts in MIPS (Million Instructions per Second). Each task's computational complexity is measured in MI (Million Instructions). Although MIPS is not the definitive way to measure CPU performance it allows for Cloudsim to measure task execution time in a simple way, and is an acceptable performance indicator for older or low-end processors. Simulation scenarios aimed at covering real life scenarios, where highly populated IoT networks produce a large amount of tasks and data. In particular, small scale rural areas, such as home automation and smart grid platforms can be greatly optimized with the usage of the proposed fog platform. IoT networks for production environments are another example on the importance of the fog paradigm and how it can aid in optimizing the management of workloads that are time sensitive and affect critical decisions. In that respect, the configuration we applied in the simulation was the following:

- *Tasks*: 5000 tasks were scheduled to arrive at random time points. A small number of tasks (5%) were characterized as time sensitive. This meant that such tasks were not forwarded to the Cloud layer or to mobile devices, in order to a) guarantee the task's successful execution and b) to achieve the best possible execution time. All values for task MI, RAM and task data size were selected randomly from a normal distribution. MI value (or length) ranged from 4500 to 5500 MIs. In addition the required RAM ranged from 150 to 350 MBs. Available system memory does not contribute to the host selection, since it does not affect execution time (as long as there is enough RAM the task will be executed successfully), however the simulator will not include a host that does not possess enough system memory to the eligible host list, which will in turn be provided to the score function. Finally, task data size ranged from 1 to 5 MBs.

Table 1. Simulation Results

		Average Execution Time	Average Delay	Finish Time	Time Sensitive Tasks	Time Sensitive Average Execution Time	Tasks Forward to Cloud	Tasks Forward to Mobile Devices	Failed and Resubmitted Tasks
Scenario 1	1	18.22	2.07	9323	249	14.82			
Scenario 2	1	12.82	0.37	6770	248	10.88	108 (2.16%)		
Scenario 3	1	3.44	0.1	5407	278	2.78	105 (2.1%)		
	2	3.35	0.145	5470	237	2.85	110 (2.19 %)	334 (6.68%)	21

- *Cloud Layer (Scenarios 1, 2, 3)*: We assume that due to conservative allocation in a centralized platform tasks were forwarded to a VM that has a computational capacity of 2660 MIPS (Xeon 3075), 4096 MB of RAM, 10 Mb/s available bandwidth and network latency of 16 ms (assuming a cover area of 300-500 km).
- *Fog Networks (Scenarios 2, 3)*: Each fog network (stable or close range) contains three hosts with 1256 (ARM Cortex v5), 1536 (ARM v7) and 847 (ARM11 family) MIPS respectively in order to model some widely used devices (e.g. Raspberry Pi 1 & 2). Each host had also 1024 MB of RAM. Bandwidth for close range fog networks ranged from 40 to 70 Mb/s and network latency ranged from 2 to 6 ms (assuming a cover area of 10 km).
- *Mobile devices (Scenario 3)*: Two mobile devices were modeled with a computational capacity of approximately 2500 MIPS (Qualcomm Scorpion),[14] 1024 MB of RAM, 20 Mb/s bandwidth and network latency that ranged from 6 to 10 ms.

For each task the balancer algorithm created a list of eligible hosts, and excluded the rest (not enough RAM or battery). Subsequently, it applied the score function to each host and selected the one to allocate the task for execution.

A random value was used to allow for a small chance (<0.05) for a device to become unavailable thus resulting in the task execution failure and consequently enforcing the platform to reallocate the task. Finally, each mobile device had a random battery value assigned, which as simulation time passed dropped at a steady rate (0.001 percent). When battery dropped below a certain level (<20 percent), the balancer module excluded that device from the score function routine. For each scenario we performed ten simulation runs and selected the worst one in terms of Finish Time. Table 1 presents the results of each scenario around eight metrics:

- *Average execution time*: The average time that takes for each task to complete since its execution started
- *Average delay*: The average time that passed from the point that the task left the Fog Broker to the point that its execution started on the host. The delay includes transmission time (depends on available bandwidth and data size), network latency and host utilization (if a host has other tasks running it may postpone the execution of tasks waiting in queue)
- *Finish time*: The overall time needed for the execution of 5000 tasks in total
- *Number of time sensitive tasks*: The number of tasks that were characterized as time sensitive
- *Time sensitive task average execution time*: The same as the first metric for time sensitive tasks
- *Number of tasks forwarded to Cloud*: For Scenarios 2 and 3
- *Number of tasks forwarded to mobile devices*: For Scenario 3
- *Number of failed and resubmitted tasks*: The number of tasks that were allocated to mobile hosts and failed due to the fact that the host became at some time unavailable. This tries to capture the reaction of the platform in cases of unexpected mobility behaviors

Scenario 1

In the first scenario, the traditional Cloud approach is simulated. No fog network is present and all tasks are sent directly to the CL. All metrics are poor, showing how centralized platforms perform when having to satisfy multiple workloads and allocate resources in a conservative manner. Low available bandwidth and higher latency as well as the lack of sufficient resources (for the particular scale of tasks) lead to high average delay and execution times. The overall simulation time for 5000 tasks was very high as well.

Scenario 2

In Scenario 2 a traditional fog approach is simulated. This time tasks are sent to the fog layer (stable fog) for execution, apart from those who require greater amounts of resources, in which case they are forwarded directly to the CL. As shown in Table I, the simulation time is much lower. This happens because more hosts are available for each task and the data transmission time is much lower due to higher bandwidth and lower latencies. In this scenario the Workload balancer score function is utilized, but since network latency is the same for all hosts and because there are no mobile devices, the balancer selects always the most under-utilized host.

Scenario 3

The third scenario simulates the proposed platform architecture where another fog network is subscribed to the stable fog network's topic. An additional simulation sub-scenario is presented, where two mobile devices are present in the fog layer as well. As described below, the simulated architecture performs better in many ways.

Scenario 3.1 In this sub-scenario, an additional close range network is added to the stable edge. At first, the score function tends to select hosts from the stable fog network. This is expected, because all subscribed hosts have more or less the same amount of available resources and thus network latency is the dominant parameter. However, as the stable Fog Hosts become over utilized the score function starts to select hosts from subscribed fog networks to efficiently balance the workload.

Scenario 3.2 In the second sub-scenario mobile devices are utilized as well (in addition to the close range fog network) and while this time some tasks had failed and resubmitted (due to the fact that mobile devices became at some point unavailable), Average Execution and Finish Times are again at a very satisfying level. Mobile devices managed to serve 334 tasks before they ran out of battery or became permanently unavailable.

The efficient management of resources and data in the rapidly evolving fog computing landscape is crucial for allowing all kinds of IoT networks to operate in way that best possible QoS is achieved. In this article, we presented a platform that tries to cover most such aspects by utilizing the advantages of Edge Computing in novel and smart way. The introduction of workload balancing logic into the stable Edge, the communication of devices and networks through the MQTT messaging protocol, as well as the incorporation of close range or mobile edges in a cooperative way for specific types of tasks, are features that can be integrated into Edge Computing architectures to allow better efficiency, coverage and QoS. Simulation results showed that the proposed architectural approach enables fog networks to be more efficient in handling time critical tasks.

The use of incentives for the participation of edge devices in the creation of the fog computing framework is essential. Though in certain use cases the benefits in the use of personal devices for participating in the fog are evident, there are scenarios where the voluntary participation of edge devices cannot be guaranteed. A characteristic case is that of remote sensing (i.e. for early detection of risks related to forest fires,[15] monitoring of environmental conditions in archaeological sites[16]), where mobile devices of citizens can be used. In such cases, reciprocal models, where end users could voluntarily offer the use of resources from their devices in order to get discounts or credits could be used. Moreover, future steps will include the use of opportunistic mobile services,[10] and the use of reinforcement learning techniques (e.g., Q-learning) for adjusting the weights of the proposed cost function. In terms of validation, the constant evolution of the simulator software is planned in order to produce the most accurate results possible (e.g. include latency in the Hub Layer as well). Finally, due to security and privacy issues, lightweight, but yet effective, encryption and access control techniques should be examined in the future.

References

1. F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things", In Proceedings of the first edition of the MCC workshop on Mobile cloud computing (MCC '12). *ACM*, New York, NY, USA, pp. 13–16, 2012.
2. MQTT Connectivity Protocol, http://mqtt.org/, accessed on February 2017
3. O. Salman, I. Elhajj, A. Kayssi and A. Chehab, "Edge computing enabling the Internet of Things", Internet of Things (WF-IoT), 2015 *IEEE 2nd World Forum* on, Milan, pp. 603-608, 2015.
4. Y. Xu, V. Mahendran, Sr. Radhakrishnan, "Towards SDN-based fog computing: MQTT broker virtualization for effective and reliable delivery", in the proceedings of *COMNETS* 2016, pp. 1–6, 2016.

5. M. Aazam and E. N. Huh, "Dynamic resource provisioning through Fog micro datacenter,"Pervasive Computing and Communication Workshops (PerCom Workshops), 2015 *IEEE International Conference* on, St. Louis, MO, pp. 105–110, 2015.
6. C. Prazeres, M. Serrano, "SOFT-IoT: Self-Organizing FOG of Things", in the proceedings of *30th International Conference on Advanced Information Networking and Applications Workshops*, pp. 803–808, 2016.
7. F. Li, M. Voegler, M. Claessens, Sch. Dustdar, "Efficient and Scalable IoT Service Delivery on Cloud", in the proceedings of *IEEE Sixth International Conference on Cloud Computing*, pp. 740–747, 2013.
8. M.Aazam, Eui-Nam Huh, "Fog Computing and Smart Gateway Based Communication for Cloud of Things", in the proceedings of *Future Internet of Things and Cloud* (FiCloud), pp. 464–470, 2014.
9. F. Bonomi, R. Milito, P. Natarajan and J. Zhu, "Fog computing: A platform for internet of things and analytics.", *Big Data and Internet of Things: A Roadmap for Smart Environments*, Springer International Publishing, pp. 169–186, 2014.
10. M. Chen, Y. Hao, Y. Li, C. F. Lai and D. Wu, "On the computation offloading at ad hoc cloudlet: architecture and service modes", *IEEE Communications Magazine*, vol. 53, no. 6, pp. 18–24, June 2015.
11. R. Baldoni, M. Contenti, and A. Virgillito. "The Evolution of Publish/Subscribe Communication Systems", *Future Directions of Distributed Computing*. Springer Verlag LNCS Vol. 2584, pp. 137–141, 2003.
12. R. N. Calheiros et al., "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms", *Software: Practice and Experience*, vol. 41, pp.23–50, 2011.
13. CloudSimEX Project, https://github.com/Cloudslab/CloudSimEx, accessed on September 2016
14. STORM project, H2020, DRS-11-2015 - Disaster Resilience & Climate Change topic 3: Mitigating the impacts of climate change and natural hazards on cultural heritage sites, structures and artefact, http://www.storm-project.eu/, accessed on 29/09/2016.
15. L. Yu, N. Wang, X. Meng, "Real-time forest fire detection with wireless sensor networks", *International Conference on Wireless Communications, Networking and Mobile Computing*, pp. 1214–1217, 2005.
16. Dhrystone 2 MIPS Benchmarks, http://www.roylongbottom.org.uk/Raspberry%20Pi%20Benchmarks.htm#anchor5a, accessed on January 2017.

ANDREAS KAPSALIS *is a PhD candidate in the Department of Electrical and Computer Engineering at the National Technical University of Athens and a researcher in the Institute of Communications and Computer Systems. He received his degree in Telecommunications from the University of Peloponnese and his MSc diploma in Network-centric Systems from the department of Digital Systems at the University of Piraeus. His active research interests include Cloud Computing, Big Data, and the Internet of Things. He has participated in several National and European research projects. Contact him at akapsalis@icbnet.ece.ntua.gr.*

PANAGIOTIS KASNESIS *is a PhD candidate in the Department of Electrical and Computer Engineering at the National Technical University of Athens and a researcher in the Institute of Communications and Computer Systems. His research interests include Semantic Web technologies machine learning, game theory in multi-agent systems, and the Internet of Things. Kasnesis is a semantic application developer and a machine learning engineer at the Piraeus University of Applied Sciences and has participated in several European research and development projects. He received an MSc in techno-economic systems from the University of Piraeus and the National Technical University of Athens. Contact him at pkasnesis@icbnet.ece.ntua.gr.*

CHARALAMPOS Z. PATRIKAKIS *is an Associate Professor at the Dept. of Electronics Engineering of Piraeus University of Applied Sciences. He has participated in more than 32 National, European and International programs, in 16 of which he has been involved as technical coordinator or principal researcher. He has more than 100 publications in chapters of books, international journals and conferences, and has 2 contributions in national legislation. He is a member of the editorial committee of more than 50 international journals and conferences, and has acted as editor in the publication of special issues of international journals, conference proceedings volumes and coedited three books. He is a senior member of IEEE, a member of the Technical Chamber of Greece, the European Association for Theoretical Computer Science, ACM, and counselor of the IEEE student department of Piraeus University of Applied Sciences. Contact him at bpatr@puas.gr.*

IAKOVOS S. VENIERIS *is a Professor in the School of Electrical and Computer Engineering of the National Technical University of Athens and director of the ICBNet Laboratory. His research interests are in large scale distributed systems, security and privacy, software and service engineering, agent technology, multimedia, mobile communications, intelligent networks, internetworking, signaling, resource scheduling and allocation for network management, modeling, performance evaluation, and queuing theory. Prof. Venieris has more than 350 publications in these areas and has participated in and successfully led several national and international R&D projects. He has served as associate and guest editor of international journals, as well as member of the TPC of several conferences. He is a coeditor of "Intelligent Broadband Networks" (Wiley, 1998), "Object Oriented Software Technologies in Telecommunications" (Wiley, 2000) and "Enhancing the Internet with the CONVERGENCE System" (Springer, 2014). Contact him at venieris@cs.ntua.gr.*

DIMITRA I. KAKLAMANI *received the PhD degree from the National Technical University of Athens (NTUA) in electrical and computer engineering (ECE), in 1992. In April 1995, April 2000, October 2004, and February 2009, she was elected Lecturer, Assistant Professor, Associate Professor, and Professor, respectively, with the School of ECE, NTUA. She has over 300 publications in the fields of software development for information transmission systems modeling, microwave networks, mobile and satellite communications, and has coordinated the NTUA activities in the framework of several EU and National projects in the same areas. She is Editor of one international book by Springer-Verlag (2000) in applied CEM and reviewer for several IEEE journals. Contact her at dkaklam@mail.ntua.gr.*