Resource Aware Placement of IoT Application Modules in Fog-Cloud Computing Paradigm

Mohit Taneja[*,†], Alan Davy[*,†]
[*]Telecommunications Software and Systems Group, Waterford Institute Of Technology, Waterford, Ireland
[†]CONNECT- Centre for Future Networks and Communications, Ireland
Email: {mtaneja, adavy}@tssg.org

***Abstract*—With the evolving IoT scenario, computing has spread to the most minuscule everyday activities, leading to a momentous shift in the way applications are developed and deployed. With the volume of impact increasing exponentially, a coherent approach of deploying these applications is critical for an efficient utilization of the network infrastructure. A typical IoT application consists of various modules running together with active interdependencies; traditionally running on the Cloud hosted in global data centres. In this paper, we present a Module Mapping Algorithm for efficient utilization of resources in the network infrastructure by efficiently deploying Application Modules in Fog-Cloud Infrastructure for IoT based applications. With Fog computing into picture, computation is dynamically distributed across the Fog and Cloud layer, and the modules of an application can thus be deployed closer to the source on devices in the Fog layer. The result of this work can serve as a Micro-benchmark in studies/research related with IoT and Fog Computing, and can be used for Quality of Service (QoS) and Service Level Objective benchmarking for IoT applications. The approach is generic, and applies to a wide range of standardized IoT applications over varied network topologies irrespective of load.**

***Index Terms*—Fog computing, Cloud computing, Latency sensitive, Application module, Resource aware placement.**

I. Introduction

The Internet of Things (IoT) has reformed the future of connectivity and reachability. It aims to bring every object online, hence generating a huge amount of data that can overwhelm the storage systems and cause a significant surge in the application reaction time. With IoT into play, the near future will involve billions of interconnected IoT devices emitting large volumes of data streams for processing. In its report, McKinsey [1] estimates that the user base will have 1 trillion interconnected IoT devices by 2025, further substantiating the impending scenario. According to this estimate, by 2025 the IoT will have a potential economic impact of USD 11 trillion per year, which nearly represents 11 percent of the world economy. Cloud computing can help here by offering on-demand scalable storage and processing services that can scale the IoT requirements; however, for latency-sensitive applications the delay caused because of communication distance between user base and Cloud is still unpleasant. Cloud computing has its advantages, but with accelerated increase in ubiquitous mobile and sensing devices coupled with upgradation in technology, the upcoming IoT ecosystem challenges the traditional Cloud computing network architecture. To address the above said challenges, meet the dynamic scalability, efficient in-network processing and latency sensitive communication, the need for IoT applications has led to the evolution of Fog Computing paradigm [2], [3].

Fog computing aims to extend the Cloud services and utilities to the network edge, thus catering to the need of latency sensitive applications and providing real-time data processing and dispatching. In this new paradigm, computing is dynamically distributed across the Cloud sites and the network elements based on the Quality of Service (QoS) requirements. Together, these two paradigms can offer a fruitful interplay between the Fog and Cloud, particularly when it comes to catering the needs of latency sensitive applications. However, the devices closer to the network edge (routers, access points, gateways etc.) are traditionally not computationally powerful enough to host all the modules of an application (or heterogeneous modules of various applications, for that matter) in the IoT ecosystem, and hence the strategy needs to be formulated in a way which keeps these constraints in mind, i.e., iterates from Fog layer towards the Cloud and tries to place the modules first on the available resources on Fog layer, thereafter iterating towards the Cloud.

It is beyond doubt that there would arise a need of further research to meet the challenges related to the evolving Fog-Cloud Architecture. The new paradigm requests a change in the way applications are developed and deployed, and to fill this gap, we introduce and formulate a strategy for efficient allocation/placement of application modules in a combined Fog-Cloud paradigm, the main aim of which is to provide an efficient utilization of network resources and minimize application latency. In this paper, we present a different approach to the problem from the application deployment perspective, which aims for the ultimate benefit of various players in the IoT ecosystem. Our approach to the problem addresses the following:

- The latency sensitive need of the upcoming and future IoT applications and their deployment strategy in the Fog-Cloud architecture.
- Efficient utilization of resources in the network infrastructure.

The paper is structured as follows: §II contains related work, §III elaborates the system model and formulates the problem, §IV is the proposed solution, §V evaluates and validates the results, and §VI contains the concluding remarks and future work.

II. Related Work

As Fog computing is a relatively new paradigm, research on it is still in its nascent stage. The ongoing work ranges from defining the architecture [4] to assessing its suitability in the context of IoT [5]. Other aspects include the programming model approach to be followed for development of Fog based IoT applications [6] to its scalability for large scale geographical distribution [7], [8], as well as a context aware real-time data analytics platform for the Fog [9]. In [10], authors suggest that Fog computing is one of the key ingredients and enablers in the IoT ecosystem. Authors in [11] explore energy consumption optimization and energy management as a service in Fog Computing architecture. The importance, applicability, need and challenges have been addressed by authors in [12], [13] at a preliminary level. The work done by authors in [14] models the problem of workload allocation from a power consumption perspective; and [15] looks into optimizing the service allocation in combined Fog-Cloud scenarios. All these contributions, including ours, work together to enable the deployment of the Fog as a platform to provide better services to the end user, and increase the network efficiency over existing infrastructure to enable and unleash the full potential of the new breed of applications triggered by IoT and smart technology scenarios.

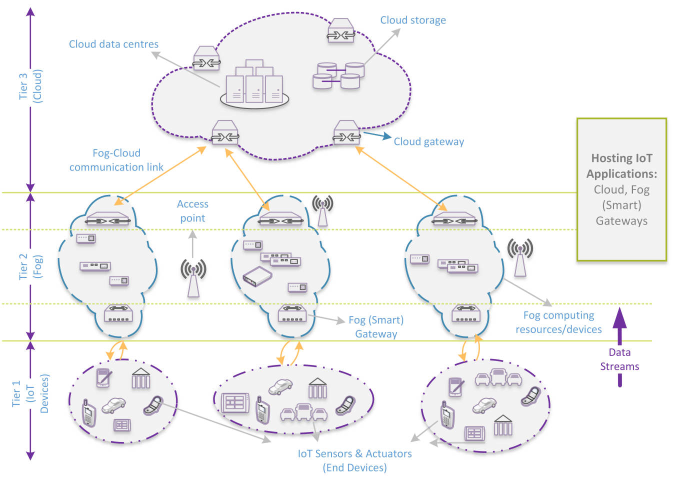
Further, our work can be extended to work in context of Edge computing for Fog only or Fog-Cloud scenarios. For them, another set specifying the resources available in the Edge layer can be included and mapped. The authors in [16] have worked upon distributed scheduling of event analytics across Edge and Cloud.

III. System Model and Problem Formulation

Fog-Cloud approach leverages the strengths of both Edge and Cloud computing by enabling low latency processing over a wide geographical base, while at the same time allowing for a scope of management of compute, networking and storage services between the centralized Cloud and the distributed end devices. This, in turn, thereby supports mobility, as well as resource and interface heterogeneity.

A generic Fog-Cloud architecture illustrating the distributed data processing in Fog-Cloud Environment is shown in Fig. 1. While the end devices come under the first tier, the Fog and the Cloud layer comprise the second and the third tier respectively, thereby together forming the Three Tier Architecture. Each tier can be mapped to support a specific component of the application, which is further elaborated and worked upon further in this section.

In the architecture, any element in the network that is capable of hosting application module(s) is considered as a Fog Device [17]. Depending on case, it may be some other common device, such as small server, or routers, access-points, or gateway etc [18]. Amongst the available networking devices in the network infrastructure, the devices of prime



Tier 3 (Cloud)
Cloud data centres
Cloud storage
Cloud gateway
Fog-Cloud communication link
Hosting IoT Applications: Cloud, Fog (Smart) Gateways
Tier 2 (Fog)
Access point
Fog (Smart) Gateway
Fog computing resources/devices
Data Streams
Tier 1 (IoT Devices)
IoT Sensors & Actuators (End Devices)

Fig. 1. Fog-Cloud Architecture (Distributed Data Processing) in standardized IoT architecture

consideration to us here in Fog-Cloud architecture are the gateways that connect the devices in the bottom most layer (Tier 1/IoT layer) to the Internet. While the Fog layer can be conceptualized as that comprising of all the aforementioned devices, in practicality, the gateways have emerged as the key constituents of the Fog, for the reason that they support/enable protocol conversion across different network segments, and are thus least conservative about additional requirements and constraints. Thus, the gateways that are also additionally working as Fog devices in the network are termed as Smart Gateways, and are present in Tier 2 of the network architecture.

Each network node has a specific computing capability, which in turn forms (and contributes to) the resources available in that layer. While the devices in the Fog layer have a defined and limited computing capability owing to a hardware constraint of the resources each device has, the Cloud, on the other hand has corresponding resources realized in terms of multiple virtual machines (VMs), each configurable to a specific configuration which tells the computing capability of that VM. This is owing to the fact that the overall computational/resource capacity of the Cloud, as a whole, is tremendously more than that of a typical device available in Fog layer, so much that in comparison to the Fog devices, the Cloud is visualized as being limitless. Thus, the complete set of resources comprise of the devices at the Fog, and VMs at the Cloud.

The computational capacity (or resource capacity) of a network node is bounded by a general set of finite constraints, and can be represented as a set of three narrowed-down basic attributes, namely CPU, RAM and Bandwidth. It is, however, to be noted that the proposed algorithm is scalable even on adding more number of attributes for a node, and more of them (like storage capacity) can be included if required by the application specifically. Thus, if n_i represents a network node i in the infrastructure, the capacity of the said node is represented as:

$$Cap(n_i) = < CPU_i, RAM_i, Bandwidth_i > \quad (1)$$

The set of all computing resources available within the IoT infrastructure/ecosystem is given by N.

$$N = \{n_i\} \quad (2)$$

N can be divided into two mutually exclusive subsets— N_F and N_C.

$$N_F \;=\; Set\ of\ network\ nodes\ in\ Fog\ layer \quad (3)$$

$$N_C \;=\; Set\ of\ network\ nodes\ in\ Cloud\ layer \quad (4)$$

$$N_F \cup N_C \;=\; N \quad (5)$$

$$N_F \cap N_C \;= \phi \quad (6)$$

The applications developed for deployment in this Fog-Cloud architecture as used in our paper are based on the Distributed Data Flow Model (DDF) [6]. Distributed computing environment calls for distributed components, which would give better results with multi-component applications, for which DDF is one of the best approaches available.

As shown in Fig. 2, the application developed and deployed for simulation of the algorithm as proposed in the paper was modeled as a Directed Acyclic Graph (DAG), with various application modules constituting the data processing elements. In context of analytics applications such as stream analytics or event based analytics, these modules are usually termed as application operators. In the DAG so formed, the vertices represent the various modules of the application, and the edges represent the data dependencies between them. These modules perform processing on the incoming data, and the edges connect the output of one module to the input of another, representing the flow of data between the modules. In mathematical notation, the DAG $\mathcal{G}$ of an application consists of vertices (V) and edges (E) and is written as follows:

$$\mathcal{G} = \langle V, E \rangle \quad (7)$$

Each module of the application has a requirement represented as a set of three attributes, namely CPU, RAM and Bandwidth. The proposed algorithm, though, is similarly scalable even on adding more number of attributes as requirements, and thus more of them can be included if need be. Thus, if v_i represents an application module i in the application, the requirement of the said module is represented as:

$$Req(v_i) \;= < CPU_i, RAM_i, Bandwidth_i > \quad (8)$$

The set of all modules of the application is denoted by V.

$$V = \{v_i\} \quad (9)$$

An edge originating from an application module v_i to another application module is denoted by e_i, and indicates data flow in an application. The set of all edges in the DAG of an application is denoted by E.

$$E = \{e_i \mid e_i = \langle v_i, v_j \rangle\} \qquad \forall v_i, v_j \in V \quad (10)$$

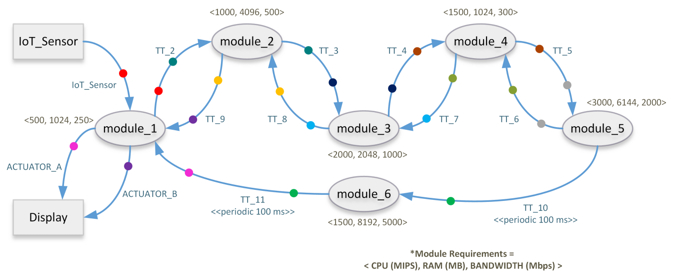


Fig. 2. Directed Acyclic Graph (DAG) of the application deployed, with relevant tuples indicating the values used for simulation of the algorithm. The number of modules as well as sensors/actuators are synchronous to the needs of a typical IoT application, and may vary as per the use case and application size; the modules can be conveniently linked to various logics to convey the various stages of application processing, as are the ones conveyed by the color pair encoding here.

There are two types of edges possible in a DAG— periodic and event based. Tuples on a periodic edge are emitted regularly at the specified interval; whereas in event based, a tuple is sent out if the source module of the edge receives an incoming tuple, and the defined selectivity model (Fractional Selectivity, in our case) allows its emission.

A Module Mapping function $\mathcal{M}$,

$$\mathcal{M} : V \rightarrow N \quad (11)$$

indicates the network node on which the application module is placed during the application deployment, such that it meets the following:

$$\forall (v_i, n_i) \in \mathcal{M} \;\Bigg|\; \substack{\implies Req(v_i) \leq Cap(n_i) \\ \forall v_i \in V \\ \forall n_i \in N} \quad (12)$$

While traditionally all the application modules were placed on the Cloud, this brought in a tremendous network cost, in addition to a high application response. With Fog-Cloud Computing approach, these application modules can be distributed across the Fog and the Cloud resources based on meeting the module requirements and network capacity constraints as shown in the above equation (12).

We propose to identify the DAG of the application as having Static and Dynamic Characteristics as follows:

- *Static Characteristics:* These are those which we expect developers to provide, and remain invariant over time for an application— such as data (Tuple) emission rate of sensors, data processing rate (Selectivity Model) of application modules, etc.
- *Dynamic Characteristics:* These are those which come into play once the application has been deployed on the network infrastructure. These are dynamic run time characteristics of Fog and Cloud resources (which are network nodes in the IoT Ecosystem)— such as their network connectivity.

In the scope of the present study, only the static characteristics of the application DAG have been presented/studied in this paper. The results from these, however, can further be translated to address the dynamic characteristics, as in the scope of future work.

The application was formulated, modeled and deployed by building upon the components of iFogSim : a toolkit developed by Gupta et. al [19] over the CloudSim [20] framework for modeling and simulating IoT, Edge and Fog Computing environments. Built to extend the capabilities of its highly vouched and reliable Cloud counterpart to bring in the Fog layer and address its added properties, iFogSim, though just recently in, is a tool we found having a good scope for further testing the capabilities of applications using the Fog over and along with the Cloud.

The application that has been designed and modeled in the paper is motivated from standardized realistic IoT scenarios like health care [21] and latency-critical gaming [22]. The processing has been redefined to suit such cases by way of the DAG, and the tuple mapping, processing and management been further intensified to cover all cases and ensure a deeper testing of the algorithm. The application works on the Sense-Process-Actuate Model, where the information collected by sensors is emitted as data streams, which is processed and acted upon by application modules running on Fog and Cloud layer, and the resultant commands (or Outputs) are sent to the actuators. The other model, which is Stream Processing Model, is where a network of application modules running on Fog and Cloud layer continuously process data streams emitted from sensors, and information mined (or aggregated) from the incoming streams is stored in Cloud hosted in a data centre for large-scale, long-term and complex analytics. The Stream Processing Model is considered as a subcategory of the Sense-Process-Actuate Model. These models can, however, be extended to cater use-cases other than IoT applications as well.

As shown in Fig. 2, the DAG of the application modeled in the simulation consists of six application modules— module_1 to module_6. IoT_Sensor represents an Internet of Things sensor, which emits tuples of type IoT_Sensor to module_1. Display is an actuator, which is designed to respond to changes in the environment captured by the sensor. Tuples form the fundamental unit of communication between entities in the IoT Ecosystem, and are indicated over the edges in the DAG. The colored dots signify tuple mapping; for example, an incoming tuple of type TT_3 on module_3 will result in an output tuple of type TT_4.

Note that module_1 of the application needs to be placed (and run) on the end devices (Device-X-X in topology, Fig. 3 and IoT_Sensor in Fig. 2) to ensure that the user base, or the source application modules, are co-located with Fog devices in the Fog layer. The actionable (sensor and actuator) are connected to these devices running module_1 on them.

IV. Proposed Solution

To enable resource aware placement of application modules in IoT Fog-Cloud Paradigm, we propose three integrated algorithms.

A. Working of Algorithm

Algorithm 1 is the ModuleMapping Algorithm, which enables Fog-Cloud Placement. It returns the efficient mapping of modules of an application onto a network infrastructure. Taking the set of network nodes N and set of application modules V as input, it first sorts the network nodes and modules in ascending order as per their capacity and requirement respectively. A Key-Value pair corresponding to Network Node as Key and Application Module as Value is then created. The Control Loop of the algorithm (for loop/line5) runs for all the modules of the application that need to be placed, and calls the function LowerBound (Algorithm 2) in each iteration, which searches for the eligible network node meeting the requirement of the module (constraint specified in equation 12). The requirement check is ensured by Compare function (Algorithm 3), and when an eligible network node is found, the corresponding Key-Value pair entry is added into the result (moduleMap). This way the algorithm iterates from Fog nodes to Cloud nodes, first placing the modules on eligible nodes in Fog layer, and once the nodes in Fog layer are exhausted or if there is no eligible node in the Fog layer, only then it places the corresponding module on Cloud.

Algorithm 1 ModuleMapping Algorithm: Fog-Cloud Placement

Input : Set of Network nodes N and Application modules V
Output : Mapping of modules on to network nodes

```
1:  function ModuleMap(NetworkNode nodes[], AppModule modules[])
2:      Sort(nodes[]),Sort(modules[]);          ▷ in ascending order
3:      Map < $NetworkNode, AppModule[ ]$ > moduleMap;
        ▷ Creates Key-Value Pair with Network Node as Key and AppModule as Value
4:      int low = 0, high = nodes.size-1, start;
5:      for start =0 to modules.size do
6:          int i=LOWERBOUND(nodes[],modules[start],low,high);
7:          if (i != -1) then
8:              moduleMap.insert(nodes[i],modules[start]);
9:              Cap(node[i]) = Cap(node[i]) - Req(modules[start]);
10:             Sort(nodes[]);                  ▷ in ascending order
11:             low = i + 1;
12:         else
13:             moduleMap.insert(nodes[nodes.size-1],modules[start]);
14:         end if
15:     end for
16:     return (moduleMap);
17: end function
```

B. Time Complexity

The sorting of the set of network nodes and application modules takes $O(|N| * log\,|N|)$ and $O(|V| * log\,|V|)$ time respectively. The LowerBound function is called for all the modules of the application, and uses the principle of Binary Search as its basis for searching for the eligible network node for module placement, thus giving us the time complexity of

Algorithm 2 LowerBound Algorithm - Algorithm used for Search

```
1:  function LOWERBOUND(NetworkNode nodes[], AppModule module, int low, int high)
2:    int length = nodes.size, mid = $\frac{(low+high)}{2}$;
3:    while (True) do
4:      NetworkNode $x$ = node[mid];
5:      if COMPARE($x$, module) == 1 then
6:        high = mid-1;
7:        if (high<low) then return mid;
8:        end if
9:      else
10:       low = mid + 1;
11:       if (low>high) then
12:         return((mid<length-1)?mid+1:-1);
13:       end if
14:     end if
15:     mid = $\frac{(low+high)}{2}$;
16:   end while
17: end function
```

Algorithm 3 Compare Network Node and Application Module

```
1: function COMPARE(NetworkNode a, AppModule b)
2:   if (a.CPU≥ b.CPU && a.RAM≥b.RAM && a.Bandwidth≥b.Bandwidth) then return 1;
3:   end if
4:   return -1;
5: end function
```

$O(|V| * log\,|N|)$. An additional sorting is required after updating the network node selected for the placement of module (line 10, Algorithm 1) which gives us another time complexity of $O(|N| * log\,|N|)$. Thus the overall time complexity of the solution is $O((|N|+|V|+|N|*|V|)*log\,|N|+|V|*log\,|V|)$.

If $|N|$ is much greater than $|V|$, then the upper bound becomes $|N|$*$|V|$ and the complexity becomes $O(|N| * |V| * log\,|N|)$.

Usually, the Brute Force solution to such problems tends to be NP-hard, and thus we present the heuristic approach to the problem, which contributes to a logarithmic time complexity.

V. Experimental Evaluation and Validation

To test the proposed algorithm, the application was run on network topologies supplied by us as a JSON (JavaScript Object Notation) file. The scenario has been varied over three network topologies with different workloads respectively, the graphical view of one of which as generated by iFogSim is indicated in Fig. 3. The experiment was iterated on topologies with 2, 4 and 6 Fog gateways, each having two devices per Fog gateway. The experimental network configurations can be found in TABLE I, II and III.

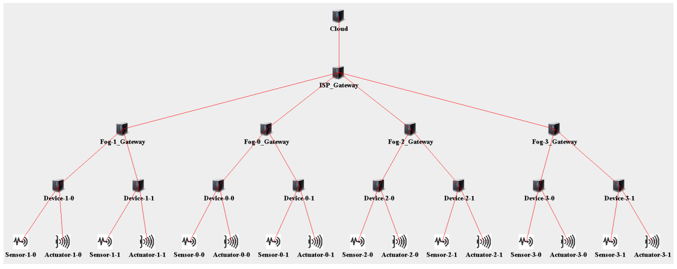


Fig. 3. One of the network topologies used for deployment iteration. The simulation has been varied over three such topologies with varied workloads, but essentially the same standardized network structure.

TABLE I
EXPERIMENTAL NETWORK CONFIGURATIONS

Between		Latency (ms)
Cloud	ISP_Gateway	200
ISP_Gateway	Fog-X-Gateway	25
Fog-X-Gateway	Device-X-X	5
Device-X-X	Sensor	2
Device-X-X	Actuator	3

TABLE II
EXPERIMENTAL NETWORK CONFIGURATIONS

Devices in Network Infrastructure	Upstream Capacity (Mbps)	Downstream Capacity (Mbps)	RAM (MB)	CPU (MIPS)
Cloud	1000	10000	40960	40000
ISP_Gateway	10000	10000	8192	10000
Fog-X-Gateway	10000	10000	6144	8000
Device-X-X	100	250	2048	4000

TABLE III
EXPERIMENTAL NETWORK CONFIGURATIONS

Tuple Type	Tuple CPU Length (MIPS)	Network Length
IoT_ Sensor	3000	500
TT_2, TT_3, TT_4, TT_5	6000	500
TT_6, TT_7, TT_8, TT_9	1000	500
TT_10, TT_11	1500	1000
ACTUATOR (A/B)	2000	500
#The average tuple emission rate of a sensor is 10 milliseconds, specified by a deterministic distribution.		

The proposed Fog-Cloud placement approach (ModuleMapping Algorithm) was compared with the traditional Cloud-based placement approach in terms of Application Latency (Response Time), Network Usage and Energy Consumption; various metrics reported by iFogSim for the modeled application using both the placement approaches were collected. The results of the simulation (Fig. 4, 5, 6) demonstrate an immensely favorable impact on Network Usage, Application Latency (Response Time) and Energy Consumption in the proposed placement approach on all 3 network topologies used.

As shown in Fig. 4, there was noticed a staggering decrease in the network usage via the approach as proposed in the paper.

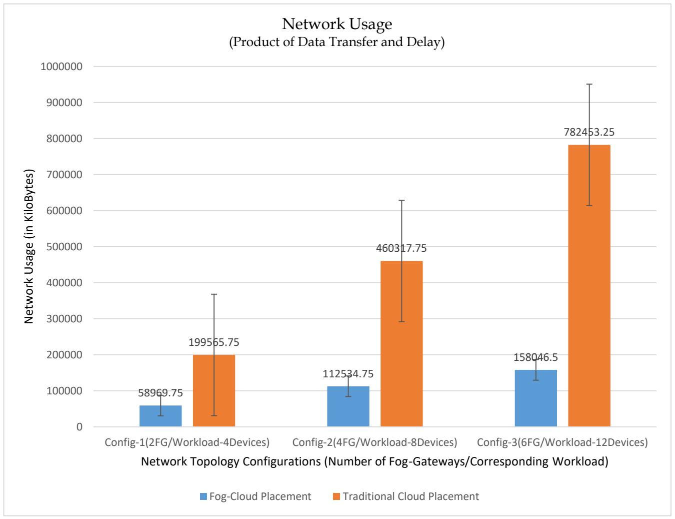


Fig. 4. Staggering decrease in network usage via proposed approach

There was also a huge effect of efficient module mapping on end-to-end latency, with highly favourable results towards Fog-Cloud placement as per the designated approach as shown in Fig. 5. Fig. 6 shows the variation of energy consumption in both the placement approaches, where we look towards creating a balance between energy consumption at high (Cloud data centres) and low (Fog layer) cost sites by spreading computing across the network via the proposed approach.

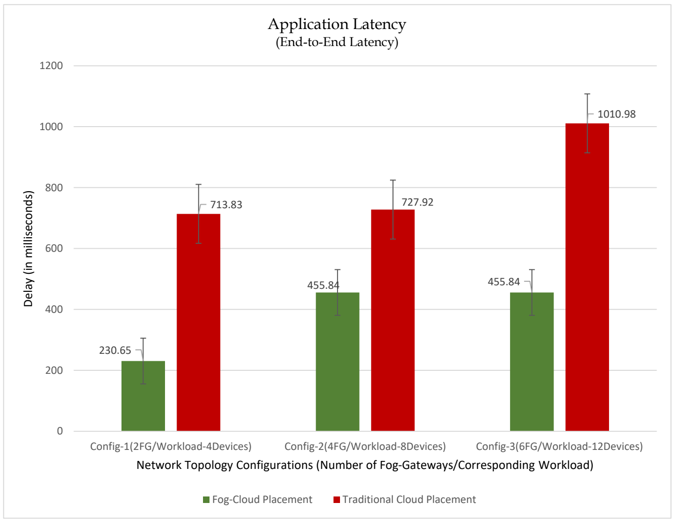


Fig. 5. Huge effect of efficient module mapping on end-to-end latency

VI. Conclusion and Future Work

Carrying forward our work [23], [24] here we present the result of the efficient utilization of resources in the network infrastructure by efficiently deploying Application Modules in Fog-Cloud Infrastructure for IoT based applications. We present the impact of an evolving paradigm that is Fog Computing towards solving the problem of latency in time critical IoT applications, while also accounting for the pressure

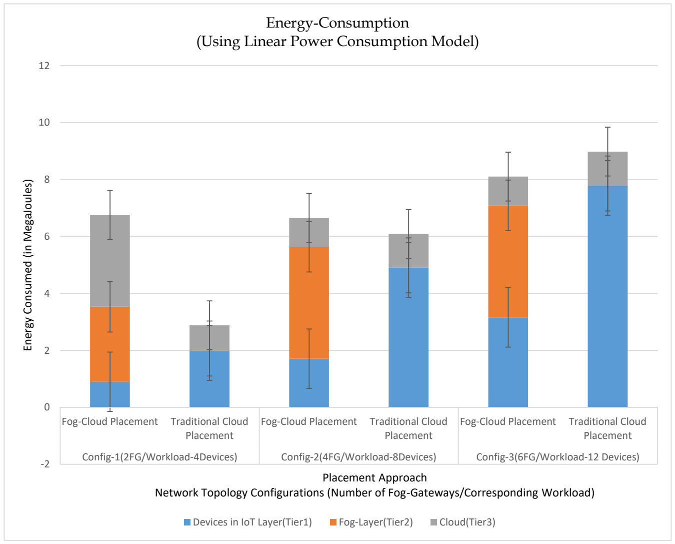


Fig. 6. Variation of energy consumption in both the placement approaches—Creating a balance between energy consumption at high (Cloud data centres) and low (Fog layer) cost sites by spreading computing across the network

on the existing network resources owing to the exponentially increasing workload due to heavy IoT usage in daily life across myriad sectors.

We outlined the key characteristics that impact the performance of such IoT applications, and have classified and kept into account the static part while increasing the network efficiency and broadening the scope of such applications. The logarithmic complexity of the Module Mapping Algorithm as proposed in the paper trumps the usual Brute Force solution to such problems, which tends to be NP-hard.

We believe that the result of this work can serve as a Micro-benchmark in studies/research related with IoT and Fog Computing, as the algorithmic approach is generic and the case study of the application has been developed keeping in mind several inline use cases applying to a wide range of IoT and Fog/Cloud applications over varied network topologies. The result obtained can thus be used for QoS and Service Level Objective benchmarking for IoT applications.

In future work, we plan to include further dynamic characteristics of the DAG once the application has been deployed. This includes the likes of network connectivity, failure of nodes, etc., all of which are dynamic run time characteristic of Fog and Cloud components. We also plan to look into the scheduling policies of resources on Fog Devices after the application deployment.

Acknowledgment

This work has emanated from research conducted with the financial support of Science Foundation Ireland (SFI) and is co-funded under the European Regional Development Fund under Grant Number 13/RC/2077.

References

[1] J. Manyika et al., “Unlocking the potential of the Internet of Things,” *McKinsey & Company*, June 2015- [Available Online,Last Accessed October 2016].

[2] 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*. ACM, 2012, pp. 13–16.

[3] F. Bonomi, R. Milito, P. Natarajan, and J. Zhu, “Fog computing: A platform for Internet of Things and analytics,” in *Big Data and Internet of Things: A Roadmap for Smart Environments*. Springer, 2014, pp. 169–186.

[4] I. Stojmenovic, “Fog computing: A Cloud to the ground support for smart things and machine-to-machine networks,” *2014 Australasian Telecommunication Networks and Applications Conference, ATNAC 2014*, pp. 117–122, 2015.

[5] S. Sarkar, S. Chatterjee, and S. Misra, “Assessment of the Suitability of Fog Computing in the Context of Internet of Things,” *IEEE Transactions on Cloud Computing*, vol. pp, no. 99, pp. 1–1, 2015.

[6] N. K. Giang, M. Blackstock, R. Lea, and V. C. M. Leung, “Developing IoT applications in the Fog: A Distributed Dataflow approach,” *Proceedings - 2015 5th International Conference on the Internet of Things, IoT 2015*, pp. 155–162, 2015.

[7] C. T. Do, N. H. Tran, C. Pham, M. G. R. Alam, J. H. Son, and C. S. Hong, “A proximal algorithm for joint resource allocation and minimizing carbon footprint in geo-distributed Fog Computing,” *International Conference on Information Networking*, pp. 324–329, 2015.

[8] E. Saurez, K. Hong, D. Lillethun, U. Ramachandran, and B. Ottenwälder, “Incremental deployment and migration of Geo-distributed situation awareness applications in the Fog,” in *Proceedings of the 10th ACM International Conference on Distributed and Event-based Systems*. ACM, 2016, pp. 258–269.

[9] P. P. Jayaraman, J. B. Gomes, H. L. Nguyen, Z. S. Abdallah, S. Krishnaswamy, and A. Zaslavsky, “CARDAP: A scalable energy-efficient context aware distributed mobile data analytics platform for the Fog,” in *East European Conference on Advances in Databases and Information Systems*. Springer, 2014, pp. 192–206.

[10] M. Yannuzzi, R. Milito, R. Serral-Gracià, D. Montero, and M. Nemirovsky, “Key ingredients in an IoT recipe: Fog Computing, Cloud computing, and more Fog Computing,” in *2014 IEEE 19th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*. IEEE, 2014, pp. 325–329.

[11] M. A. Al Faruque and K. Vatanparvar, “Energy Management-as-a-Service Over Fog Computing Platform,” *IEEE Internet of Things Journal*, vol. 3, no. 2, pp. 161–169, 2016.

[12] J. Preden, J. Kaugerand, E. Suurjaak, S. Astapov, L. Motus, and R. Pahtma, “Data to decision: pushing situational information needs to the edge of the network,” in *2015 IEEE International Multi-Disciplinary Conference on Cognitive Methods in Situation Awareness and Decision*. IEEE, 2015, pp. 158–164.

[13] T. H. Luan, L. Gao, Z. Li, Y. Xiang, and L. Sun, “Fog computing: Focusing on mobile users at the Edge,” *arXiv preprint arXiv:1502.01815*, 2015.

[14] R. Deng, R. Lu, C. Lai, and T. H. Luan, “Towards power consumption-delay tradeoff by workload allocation in Cloud-Fog Computing,” in *2015 IEEE International Conference on Communications (ICC)*, June 2015, pp. 3909–3914.

[15] V. B. C. Souza, W. Ramrez, X. Masip-Bruin, E. Marn-Tordera, G. Ren, and G. Tashakor, “Handling service allocation in combined Fog-Cloud scenarios,” in *2016 IEEE International Conference on Communications (ICC)*, May 2016, pp. 1–5.

[16] R. Ghosh and Y. Simmhan, “Distributed Scheduling of Event Analytics across Edge and Cloud,” *arXiv preprint arXiv:1608.01537*, 2016.

[17] E. Kavvadia, S. Sagiadinos, K. Oikonomou, G. Tsioutsiouliklis, and S. Aïssa, “Elastic virtual machine placement in Cloud Computing network environments,” *Computer Networks*, vol. 93, pp. 435–447, 2015.

[18] L. M. Vaquero and L. Rodero-Merino, “Finding Your Way in the Fog: Towards a Comprehensive Definition of Fog Computing,” *SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 5, pp. 27–32, Oct. 2014.

[19] H. Gupta, A. V. Dastjerdi, S. K. Ghosh, and R. Buyya, “iFogSim: A Toolkit for Modeling and Simulation of Resource Management Techniques in Internet of Things, Edge and Fog Computing Environments,” pp. 1–22, 2016. [Online]. Available: http://arxiv.org/abs/1606.02007

[20] R. Buyya, R. Ranjan, and R. N. Calheiros, “Modeling and simulation of scalable Cloud computing environments and the CloudSim toolkit: Challenges and opportunities,” in *High Performance Computing Simulation, 2009. HPCS ’09. International Conference on*, June 2009, pp. 1–11.

[21] T. N. Gia, M. Jiang, A. M. Rahmani, T. Westerlund, P. Liljeberg, and H. Tenhunen, “Fog computing in healthcare Internet of Things: A case study on ECG feature extraction,” *Proceedings - 15th IEEE International Conference on Computer and Information Technology, CIT 2015, 14th IEEE International Conference on Ubiquitous Computing and Communications, IUCC 2015, 13th IEEE International Conference on Dependable, Autonomic and Se*, pp. 356–363, 2015.

[22] J. K. Zao, T. T. Gan, C. K. You, S. J. R. Mndez, C. E. Chung, Y. T. Wang, T. Mullen, and T. P. Jung, “Augmented Brain Computer Interaction Based on Fog Computing and Linked Data,” in *Intelligent Environments (IE), 2014 International Conference on*, June 2014, pp. 374–377.

[23] M. Taneja and A. Davy, “Resource aware placement of data analytics platform in Fog Computing,” *Procedia Computer Science*, vol. 97C-PROCS9714, pp. 153–156, 2016.

[24] M. Taneja and A. Davy, “Poster Abstract: Resource Aware Placement of Data Stream Analytics Operators on Fog Infrastructure for Internet of Things Applications,” in *2016 IEEE/ACM Symposium on Edge Computing (SEC)*, Oct 2016, pp. 113–114.