Fog Computing: Towards Minimizing Delay in the Internet of Things

Ashkan Yousefpour, Genya Ishigaki, and Jason P. Jue
Department of Computer Science, The University of Texas at Dallas, Richardson, Texas 75080
Email: {ashkan, gishigaki, jjue}@utdallas.edu

***Abstract*—With the Internet of Things (IoT) becoming a major component of our daily life, understanding how to improve quality of service (QoS) in IoT networks is becoming a challenging problem. Currently most interaction between the IoT devices and the supporting back-end servers is done through large scale cloud data centers. However, with the exponential growth of IoT devices and the amount of data they produce, communication between "things" and cloud will be costly, inefficient, and in some cases infeasible. Fog computing serves as solution for this as it provides computation, storage, and networking resource for IoT, closer to things and users. One of the promising advantages of fog is reducing service delay for end user applications, whereas cloud provides extensive computation and storage capacity with a higher latency. Thus it is necessary to understand the interplay between fog computing and cloud, and to evaluate the effect of fog computing on the IoT service delay and QoS. In this paper we will introduce a general framework for IoT-fog-cloud applications, and propose a delay-minimizing policy for fog-capable devices that aims to reduce the service delay for IoT applications. We then develop an analytical model to evaluate our policy and show how the proposed framework helps to reduce IoT service delay.**

I. Introduction

The Internet of Things (IoT) is likely to be incorporated into our daily life, in areas such as transportation, healthcare, industrial automation, smart home, and emergency response. The IoT enables things to see and sense the environment, to make coordinated decisions, and to perform tasks based on these observations [1]. In order to realize the full benefits of the IoT, it will be necessary to provide sufficient networking and computing infrastructure to support low latency and fast response times for IoT applications. Cloud Computing has been seen as the main enabler for IoT applications with its ample storage and processing capacity. Nonetheless, being far from end-users, cloud-supported IoT systems face several challenges including high response time, heavy load on cloud servers and lack of global mobility.

In the era of Big Data, it may be inefficient to send the extraordinarily large amount of data that swarms of IoT devices generate to the cloud, due to the high cost of communication bandwidth, and due to the high redundancy of data (for instance, constant periodic sensor reading). Instead of moving data to the cloud, it may be more efficient to move the applications and processing capabilities closer to the data produced by the IoT. This concept is referred to as "data gravity," and fog computing is well suited to address this matter.

Fog computing is a newly-introduced concept that aims to put the cloud closer to the end users (things) for better quality of service [2], [3]. Fog computing is an intelligent layer sitting between cloud and IoT, that brings low latency, location awareness, and wide-spread geographical distribution for the IoT. Inheriting main concepts of cloud computing, fog provides computation, storage, and networking services to end-users, but at the edge of the network. Despite the countless benefits of fog, the research in this field is still immature, and many researchers still are working on defining vision, basic notions, and challenges of fog computing [2]–[5].

An open research challenge is to explore the potential benefits of fog computing. In other words, one must study how quality of service will be improved by having a layer of fog nodes between IoT and the cloud. Recent work in [6] addressed the design of a policy for assigning tasks that are generated at mobile subscribers to edge clouds, to achieve a power-delay trade-off. Despite their solid contributions, the proposed approach is limited to the cellular network infrastructures. It assumes an IoT device can be only a User Equipment (UE), and that edge servers must be attached to base stations. Additionally, by not considering the cloud in the approach, scenarios where IoT-cloud or fog-cloud communication takes place are not handled.

In a similar study to the above work, the authors in [7] study base station association, task distribution, and virtual machine placement in fog-computing-supported medical cyber-physical systems. They study the mentioned three issues, while minimizing the overall cost and satisfying QoS requirements. Although the paper has strong contributions, the suggested scheme cannot be generalized to IoT-fog-cloud scenarios, as it is based on the cellular network architecture. The scheme lacks the cloud entity, fog nodes are assumed to co-locate with base-stations, and there is no computation offloading capability.

Another effort is the work in [8] that addresses power-delay trade-off in cloud-fog computing by workload allocation. This scheme is similar to that of [6] in the sense that it tries to achieve a power-delay trade-off in edge clouds. The authors mathematically formulate the workload allocation problem in fog-cloud computing; yet, they use fairly simple models to formulate power consumption and service delay. More recent work in [9] also focuses on theoretical modeling of fog computing architectures, specifically, service delay, power consumption, and cost. Similarly, service delay and power consumption are formulated in basic models, and no policy

978-1-5386-2017-5/17 $31.00 © 2017 IEEE
DOI 10.1109/IEEE.EDGE.2017.12

is introduced for minimizing service delay.

In this work we introduce a common sense general framework to understand, evaluate and model service delay in the IoT-fog-cloud application scenarios. We then propose a delay-minimizing policy for fog nodes whose goal is to minimize service delay for the IoT nodes. In contrast to the existing work in the literature, the proposed policy employs fog-to-fog communication to reduce the service delay by sharing load. For computation offloading, the policy considers not only the queue length, but also different request types that have variant processing times. Additionally, our scheme is not limited to any particular architecture (like cellular network), and IoT, fog, and cloud nodes are not restricted to be of any type or capacity. We also develop an analytical model to evaluate service delay in the IoT-fog-cloud scenarios in detail and perform extensive simulation studies to support the model and the proposed policy.

The rest of this paper is organized as follows. We introduce our IoT-fog-cloud framework in Section II and propose the policy for reducing IoT service delay in Section III. We then formally introduce the analytical model for evaluating IoT service delay and its components in Section IV, and explain the numerical results of our experiment in Section V. Finally, Section VI summarizes the paper and provides future directions of this work.

II. General IoT-Fog-Cloud Model

Fig. 1 illustrates a general framework for an IoT-fog-cloud architecture that is considered in this work. There are three layers in this architecture: things layer, where the "things" and end-users are located, fog layer, where fog nodes are placed, and cloud layer, where distributed cloud servers are located. A *cloud server* can be composed of several processing units, such as a rack of physical servers or a server with multiple processing cores. In each layer, nodes are divided into domains where a single IoT-fog-cloud application is implemented.

For instance, a domain of IoT nodes (in a factory, for instance) is shown in dark green, and they communicate with a domain of fog nodes associated with the application. A domain of IoT nodes could comprise things in a smart home, temperature sensors in a factory, or soil humidity sensors in a farm where all the things in the vicinity are considered to be in a single domain. Normally the fog nodes in one domain are placed in close proximity to each other, for example, in a single zip-code or in levels of a building. Each domain of fog nodes is associated with a set of cloud servers for a single application. We will not pose any restrictions on the topology (any topology is allowed), except for this logical layered architecture, as this will make the model easy to present.

The basic way in which IoT nodes, fog nodes, and cloud nodes operate and interact is as follows. IoT nodes can process requests locally, send it to a fog node, or send it to the cloud; fog nodes can process requests, forward requests to other fog nodes in the same domain, or forward the requests to the cloud; cloud nodes process requests and send the response back to

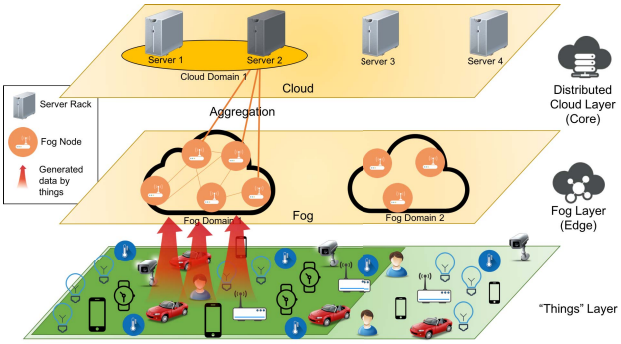

Fig. 1. General framework for IoT-fog-cloud architecture. Each layer is partitioned into domains where a single application is implemented.

the IoT nodes. In this work, the aim is to minimize service delay for IoT devices in the proposed framework based on fog computing. The fog layer lies between IoT devices and cloud, so that it could handle majority of IoT service requests, to reduce the overall service delay.

Definition 1. *service delay* for an IoT node is the time required to serve a request, i.e. the time interval between the moment when an IoT node sends a service request and when it receives the response for that request.

We first introduce the delay-minimizing policy in the following section, then formulate the IoT service delay to evaluate the policy analytically.

III. Fog Node Collaboration Policy

In this section, we introduce the framework in which fog nodes collaborate with each other to fulfill the requests sent from IoT nodes to the fog layer. If a fog node can accept a request based on its current load, it processes the request; however, when the fog node is busy processing many tasks, it may offload the request to some other fog nodes or to the cloud (this is called *Load Sharing*). The concept of Load Sharing is well studied in the literature [10], [11], and we borrow similar concepts for the design of the policy by which fog nodes collaborate. This collaboration is discussed in details the following subsections.

A. When to Offload a Task

In this subsection, we discuss the decision fog nodes make for processing or offloading a task to other fog nodes. The concept of computation offloading for mobile devices is studied in [12], [13], and a number of possible policies are discussed (based on energy consumption, response time, availability, etc.). In our scheme, the decision to offload a task is based on the response time of a fog node, which depends on several factors: the amount of computation needed to be performed on a task, and the queueing status (in terms of current load) and processing capability of a fog node. In particular, we propose a model that takes into account the different processing times of different individual tasks. In other words, in our model there

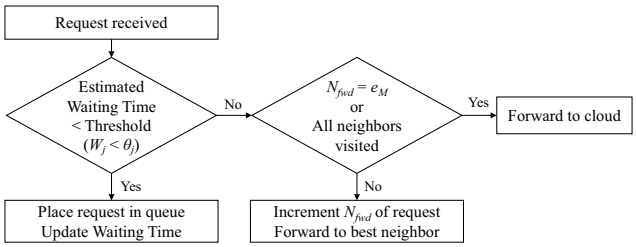

Fig. 2. Policy of Fog node j for handling received requests.

will be distinction between heavy processing tasks and light processing tasks.

We assume requests have two types: type Light (light processing) with an average processing time of z_j at the fog node j and Z_k at the cloud server k, and type Heavy (heavy processing) with an average processing time of z'_j at the fog node j and Z'_k at the cloud node k. For example, requests sent by temperature sensors to fog nodes for calculating the average room temperature can be seen as light processing tasks. Similarly, a license plate reading request in a recorded video of a vehicle, sent by a traffic camera to fog nodes is an example of heavy processing task. Note that, in general, more than two task types could be considered; however, in this paper, we only consider two task types for the simplicity of the presentation.

The procedure of processing or forwarding requests by fog nodes is shown in Fig. 2. When a fog node receives a request, it first checks the estimated waiting time of tasks in the queue.

Definition 2. *Estimated waiting time* (W) is the sum of the estimated processing delays of the requests in the queue (queueing delay), plus the estimated processing delay of the request under service.

A method for estimating processing times and hence waiting time of the tasks in the queue is discussed in Section III-D. If the estimated waiting time is smaller than a threshold θ_j at fog node j, the fog node will accept the task. The task enters the queue and the estimated waiting time is updated. If not, the fog node offloads this request, either to one of its neighbors, or to the cloud. If the number of times the request has been offloaded (N_{fwd}) is less than the *offload limit* $e_{\mathcal{M}}$ for domain $\mathcal{M}$ (domain where fog node j belongs), the request will be forwarded to a neighboring fog node. If not, it will be forwarded to the cloud.

The value of θ_j depends on the incoming traffic pattern from IoT nodes to fog nodes in a domain. In general, if all fog nodes have low load, offloading is unnecessary; and if all fog nodes have heavy load, offloading will not help reducing the delay significantly. Offloading helps when there is a high degree of variance in the load among fog nodes. The value of θ_j could be adjusted in implementation based on the given traffic demands, to reach an optimal value that minimizes the average service delay.

TABLE I
TABLE OF NOTATIONS

d_i	Service delay for an IoT node i
$d_i^{\max}$	Maximum delay allowed by application for IoT node i
p_i^I	Probability that IoT node i processes its own request
p_i^F	Probability that IoT node i sends its request to fog layer
p_i^C	Probability that IoT node i sends its request to the cloud
$X_{st}^{LL'}$	Propagation delay from node s in layer L to node t in layer L', where $s,t \in \{i,j,k\}$ and $L,L' \in \{I,F,C\}$
$Y_{st}^{LL'}$	Sum of all transmission delays on links between node s in layer L to node t in layer L', where $s,t \in \{i,j,k\}$ and $L,L' \in \{I,F,C\}$
A_i	Average processing delay of IoT node i to process its own request
L_{ij}	Delay of processing and handling requests of IoT node i in the fog layer (and possibly the cloud layer), where fog node j is the fog node to which IoT node i initially sends its request ($L_{ij} = L_{ij}(0)$)
$L_{ij}(x)$	Delay of processing and handling requests of IoT node i in fog layer (and possibly cloud layer), by fog node j during the x'th offload in the fog layer
S_Λ^L	Set of nodes in domain Λ at layer L, where $(L,\Lambda) \in \{(I,\mathcal{P}),(F,\mathcal{M}),(C,\mathcal{N})\}$
S^L	$\bigcup_\Lambda S_\Lambda^L$: set of nodes (in all domains) at layer L
$\overline{H}_k$	Average delay for handling the request at cloud server k
ς_i	Average size of request data that IoT node i generates
b_i	Probability that a generated request at IoT node i is Light
$\overline{W}_j$	Estimated waiting time of fog node j
P_j	Probability that an incoming request is accepted by fog node j
θ_j	Offloading threshold at fog node j
$e_{\mathcal{M}}$	Maximum offload limit at the fog layer in domain $\mathcal{M}$

B. Modes of Interaction

We propose two modes of interaction for fog nodes: one mode is centralized, in which a central authority controls the interaction of fog nodes, and the other one is distributed, where fog nodes interact with their neighboring fog nodes using a universal protocol.

In centralized mode of interaction, in each domain of fog nodes there is a Central Node that controls the interaction among the fog nodes in that domain. In particular, based on the current condition of their queue, fog nodes in domain $\mathcal{M}$ report their estimated time to process the requests in queue and in service (i.e. estimated waiting time) to the Central Node of domain $\mathcal{M}$.

The Central Node of domain $\mathcal{M}$ announces the estimated waiting time of the fog nodes in domain $\mathcal{M}$ to their neighboring fog nodes in domain $\mathcal{M}$. Upon reception of estimated waiting times from Central Node, fog nodes record them in a *Reachability table* they maintain. The Reachability table is a table that fog nodes utilize when making decision as to which fog node to offload a request for processing. The Reachability table has three columns as it is shown in Table II. The information in the Reachability table of a fog node in domain $\mathcal{M}$ is updated by both the Central Node of domain $\mathcal{M}$ and the fog node itself; the Central Node announces the estimated waiting times of neighboring fog nodes, and the fog nodes measure the propagation delay between themselves. An efficient way to estimate the waiting time of fog nodes is discussed in Section III-D.

TABLE II
AN EXAMPLE OF REACHABILITY TABLE OF FOG NODE

Propagation Delay (μs)	Est. Waiting Time (μs)	Node ID
450	250	129.110.83.81
300	365	129.110.5.75 *
⋮	⋮	⋮

Using estimated waiting time and propagation delay in the Reachability table, each fog node selects a fog node from among its neighbors as the *best* fog node. It does so by selecting a neighboring fog node in the same domain with the smallest estimated waiting time plus propagation delay. The best fog node's ID is marked with star * in Table II.

In the distributed mode of interaction, no Central Node is present in a domain; instead, fog nodes in domain $\mathcal{M}$ run a protocol to distribute their state information (estimated waiting time) to the neighboring fog nodes in the same domain. Each fog node maintains the Reachability table using the estimated waiting time it receives from neighboring fog nodes, and the propagation delay it measures from itself to them. Similar to central mode of interaction, a fog node always selects the best neighbor with the smallest estimated waiting time plus propagation delay.

Comparison: Central mode of interaction can be seen as central resource orchestration, where the Central Node is knowledgeable of the topology and the state of the fog nodes in a domain. Inherently, central mode of interaction is easier to implement, because there is no need for a distributed communication protocol among fog nodes; all the procedures of the estimated waiting time announcements will be implemented on the Central Node. Moreover, the Central Node could be used as a means to push fog applications and updates to the fog nodes in a domain.

On the other hand, distributed mode is more suitable for scenarios in which fog nodes are not necessarily static, or when the fog network is formed in an ad hoc manner. Furthermore, in distributed mode there is no need to have a dedicated node to act as the Central Node, which is a reduction in the cost of deployment and less vulnerable to a single point of failure. For the purpose of this paper and our simulation studies, we have chosen the distributed mode of interaction, since our policy only needs to announce estimated waiting time updates, which is fairly easy to implement on fog nodes.

C. Finding Best Neighbor

Propagation delays are given to fog nodes as input parameters. Depending on the mode of interaction, upon receiving an estimated waiting time sample either from the Central Node or another fog node, the fog node updates the corresponding estimated waiting time in the Reachability table. As mentioned above, a fog node selects as its best neighbor the fog node for which the estimated delay for the current request is minimal.

When a request is offloaded to the fog node's best neighbor, it undergoes the corresponding propagation delay to reach the best neighbor, and if it enters the best neighbor's queue, it spends time roughly equal to the estimated waiting time of the best neighbor, before it finishes processing at that neighbor. If the request does not enter the queue of the fog node's best neighbor, then the request needs be offloaded again (multiple offloads are possible when neighboring fog nodes are busy with requests).

We assume that link rates between fog nodes in one domain are the same; hence we did not include transmission delays in the Reachability table. However, if the link rates between fog nodes are different, transmission delay could be included as a column in the Reachability table for better selection of best neighbor. It is worth mentioning that when the fog nodes are mobile, they need to frequently measure the propagation delay and update the corresponding propagation delay in the Reachability table.

D. Checking and Updating Queue Status

When fog nodes receive requests from IoT nodes that participate in an application, they need to distinguish between type Light and type Heavy requests, in order to update the queue parameters. To address this, we assume that requests have in their header a field that identifies the type of the request (for instance *Traffic Class* field in IPv6 header). The application hence sets that field in header of the packets being generated from IoT nodes.

Fog nodes always need to know an estimate of the current total processing time of the tasks in their queue (i.e. estimated waiting time), both for when they need to make offloading decisions, and when reporting their estimated waiting time to neighboring fog nodes. A possible method for estimating the waiting time is for the fog node to store an estimate of the *current* waiting time of the tasks in the queue as a parameter. Upon arrival to or departure from the queue, the fog node simply updates the estimated waiting time of the tasks in the queue.

To calculate the estimated waiting time of the tasks in the queue W, fog node j periodically measures processing times of recently processed tasks (new_z_j) and update the estimated processing time of each task type (z_j). For light processing tasks, we have $z_j = (1-\alpha) \times z_j + \alpha \times new\_z_j$ (a similar equation holds for heavy processing tasks). This equation is a weighted average that maintains a weight of α for new measured processing times and a weight of $1-\alpha$ for the old measurements (current estimate).

To obtain the total processing time of the tasks in the queue on the fly, fog node j stores the current number of type Light and Heavy requests in the queue c_j and c'_j, respectively, in addition to z_j and z'_j. Fog node j then multiplies the estimated processing time of each task type by the number of tasks of that type in the queue. In other words, the estimated waiting time W of the fog node j will be

$$W_j = c_j \times z_j + c'_j \times z'_j. \quad (1)$$

IV. ANALYTICAL MODEL

In this section, we introduce the analytical model to evaluate the service delay in the proposed architecture.

A. Service Delay

Recall that IoT nodes process requests locally, send it to a fog node, or send it to the cloud. Thus, service delay d_i for an IoT node i can be written as:

$$\begin{aligned} d_i = p_i^I \times (A_i) + p_i^F \times (X_{ij}^{IF} + Y_{ij}^{IF} + L_{ij}) \\ + p_i^C \times (X_{ik}^{IC} + Y_{ik}^{IC} + \overline{H}_k + X_{ki}^{CI} + Y_{ki}^{CI}); \\ j = f(i),\ k = g(i), \end{aligned} \tag{2}$$

where p_i^I is the probability that the IoT node i processes its own request at the things layer, p_i^F is the probability of sending the request to the fog layer, and p_i^C is the probability that the IoT node sends the request directly to the cloud; $p_i^I + p_i^F + p_i^C = 1$. A_i is the average processing delay of the IoT node i when it processes its own request. X_{ij}^{IF} is propagation delay from IoT node i to fog node j, Y_{ij}^{IF} is sum of all transmission delays on links from IoT node i to fog node j. Similarly, X_{ik}^{IC} is propagation delay from IoT node i to cloud server k, Y_{ik}^{IC} is sum of all transmission delays on links from IoT node i to cloud server k. X_{ki}^{CI} and Y_{ki}^{CI} are the propagation and transmission delays from cloud server k to IoT node i.

The transmission and propagation delay from the fog layer to IoT node i will be included in L_{ij}, since the request may be further offloaded to a different node in the fog layer (more details in Section IV-D).

L_{ij} is the delay for processing and handling requests of IoT node i in the fog layer (and possibly cloud layer, if fog nodes offload the request to the cloud), where fog node j is the first fog node to which IoT node i initially sends its request. Note that fog node j might offload the request to another fog node or to the cloud, and that all the corresponding incurred delays are included in L_{ij}. $\overline{H}_k$ is average delay for handling the request at the cloud server k, which consists of the queueing time at the cloud server k plus the processing time at the cloud server k. (L_{ij} and $\overline{H}_k$ will be discussed in further detail in Sections IV-D and IV-F respectively).

$f(i)$ and $g(i)$ are mapping functions that indicate the fog node j and cloud server k to which IoT node i sends its requests, respectively. For instance, if in an application, IoT node i always sends its requests to fog node j^* in the fog layer, then $f(i) = j^*$. In another scenario if IoT nodes always send their requests to the closest fog node in the fog layer, then $f(i) = \arg\min_j X_{ij}^{IF}$, which translates to the index of the fog node with smallest propagation delay (distance) from IoT node i.

To formalize the problem further, let us define an IoT-fog-cloud application Ψ. In the rest of this work all the equations are defined on a single application $\Psi(\mathcal{N}, \mathcal{M}, \mathcal{P})$.

Definition 3. IoT-fog-cloud *application* Ψ is an application on domain $\mathcal{N}$ of cloud servers, domain $\mathcal{M}$ of fog nodes and domain $\mathcal{P}$ of IoT nodes, and is written as $\Psi(\mathcal{N}, \mathcal{M}, \mathcal{P})$. Examples of Ψ are: video processing, temperature sensor reporting, traffic road analysis, and oil rig pressure monitoring.

We do not assume any particular distribution for p_i^I, p_i^F, and p_i^C, since their values will be defined by individual applications and based on QoS requirements and policies. In other words, their values will be given to this framework as input.

By using the model to evaluate the service delay for the IoT nodes in one application domain, our goal is to minimize delay through the definition of policies for exchanging requests between IoT nodes, fog nodes, and cloud nodes. We formally state the above as the minimization of the average service delay of IoT nodes in domain $\mathcal{P}$, under the condition that every service delay d_i is smaller than a threshold $d_i^{\max}$, or

$$\begin{aligned} \min \quad & \frac{1}{|S_{\mathcal{P}}^I|} \sum_{i \in S_{\mathcal{P}}^I} d_i \\ \text{subject to } & d_i < d_i^{\max} : \forall i \in S_{\mathcal{P}}^I, \end{aligned} \tag{3}$$

where $S_{\mathcal{P}}^I$ denotes the set of IoT nodes that are in domain $\mathcal{P}$. If an application requires one delay threshold $d^{\max}$ for all IoT nodes, then $\forall i, d_i^{\max} = d^{\max}$.

In the following subsection, we will discuss in more details the components of the service delay equation.

B. Propagation and Transmission Delays

In Equation (2) we have the IoT-cloud delay terms X_{ik}^{IC}, Y_{ik}^{IC}, X_{ki}^{CI}, Y_{ki}^{CI} when the IoT node sends its requests directly to the cloud layer. These terms are effective in the equation in cases where the application Ψ is not implemented in the fog layer ($S_{\mathcal{M}}^F = \{\}$), or when the request has to be sent to the cloud for archival purposes, or when there is no fog layer and the IoT node communicates directly to the cloud.

Recall that Y_{ij}^{IF} is the sum of all transmission delays on the links from IoT node i to fog node j. If IoT node i and fog node j are l-hop neighbors, we will have

$$Y_{ij}^{IF} = \sum_{l} \frac{\varsigma_i}{R_l}. \tag{4}$$

where ς_i is the average size of request data that IoT node i generates, and R_l is the transmission rate of the l'th link between IoT node i and fog node j. The expanded equations for transmission delays between other layers (Y_{ik}^{IC}, Y_{jk}^{FC}, etc.) are derived similar to Y_{ij}^{IF}.

C. Processing Delay of IoT node

As explained in Equation (2), for IoT node i, the average processing delay is A_i. If b_i denotes the probability that a generated request at IoT node i is type Light, and $b_i' = 1 - b_i$ is the probability that a generated request at IoT node i is type Heavy, A_i could be written as

$$A_i = b_i \times a_i + b_i' \times a_i', \tag{5}$$

where a_i is the average processing time of requests of type Light at IoT node i, and a_i' is the average processing time of requests of type Heavy at IoT node i. If IoT node i is of type Light (or type Heavy), i.e. it only generates type Light (or type Heavy) requests, $b_i = 1$ (or $b_i = 0$) and $A_i = a_i$ (or $A_i = a_i'$). Should more than two types of tasks be considered, the equation above (and other equations with two task types) could be generalized to support more task types.

D. Delay in Fog Layer

In this section, we define a recursive equation for L_{ij}. Let us define $L_{ij}(x)$ as the delay of processing and handling requests of IoT node i in the fog layer (and possibly the cloud layer), by fog node j during the x'th offload in the fog layer ($x \geq 0$). Also let us label $L_{ij} \equiv L_{ij}(0)$. If P_j denotes the probability that a request is accepted by fog node j (enters the queue of fog node j) $L_{ij}(x)$ can be written as:

$$\begin{aligned} L_{ij}(x) = & P_j.(\overline{W}_j + X_{ji}^{FI} + Y_{ji}^{FI}) \\ & + (1-P_j).\Big[[1-\phi(x)].\big[X_{jj'}^{FF} + Y_{jj'}^{FF} + L_{ij'}(x+1)\big] \\ & + \phi(x).\big[X_{jk}^{FC} + Y_{jk}^{FC} + \overline{H}_k + X_{ki}^{CI} + Y_{ki}^{CI}\big]\Big]; \\ & j' = \textit{best}(j),\ k = h(j). \end{aligned} \tag{6}$$

In the equation above, $\overline{W}_j$ is the average waiting time in fog node j. $\phi(x)$ is the offloading function, which is defined as

$$\phi(x) = \begin{cases} 0 & x < e_{\mathcal{M}} \\ 1 & x = e_{\mathcal{M}} \end{cases}. \tag{7}$$

If $x < e_{\mathcal{M}}$, then $\phi(x) = 0$, which it indicates that the request will be offloaded to another fog node. If $x = e_{\mathcal{M}}$, then $\phi(x) = 1$, which means that the forward limit is reached and that the request will be offloaded to the cloud (recall Fig. 2). x takes on integer values in $[0, e_{\mathcal{M}}]$.

best(j) and $h(j)$ are mapping functions that map a particular fog node j to its best fog neighbor and the cloud server associated with the fog node, respectively. Since the choice of the best neighbor of a fog node depends on the current state of the system, the system is dynamic and $best(j)$ will be a pointer to the current best neighbor of fog node j. $h(j)$ simply holds the index of the associated cloud node for fog node j.

Explanation: Assume fog node j is the one that is selected first by the IoT node i. When a request from an IoT i node reaches the fog layer, fog node j first tries to process the request. The request enters this node's processing queue with probability P_j, and does not with probability $(1 - P_j)$, which depends on estimated waiting time. If the request enters the queue, it will experience average waiting time $\overline{W}_j$, and propagation and transmission delays of X_{ji}^{FI} and Y_{ji}^{FI} to return back to the IoT node. Note that the processing delay of the current task entering the fog node j's queue is already included in $\overline{W}_j$.

If the request does not enter fog node j, fog node j will offload the request to its best fog neighbor j', which incurs a propagation and transmission delay of $X_{jj'}^{FF}$ and $Y_{jj'}^{FF}$ respectively. The request also undergoes a delay of $L_{ij'}(x+1)$, which is the delay of processing and handling the request in the fog layer (and possibly the cloud layer), by fog node j' during the $(x+1)$'st offload. Finally when a request has been offloaded $e_{\mathcal{M}}$ times ($\phi(x) = 1$), if the last fog node needs to offload the request, it will do so by offloading it to the cloud, which incurs fog-cloud propagation and transmission delay of X_{jk}^{FC} and Y_{jk}^{FC} respectively, cloud processing delay of $\overline{H}_k$, and cloud-IoT propagation and transmission delay of X_{ki}^{CI} and Y_{ki}^{CI} respectively.

The reason that X_{ji}^{FI} and Y_{ji}^{FI} are included in Equation (6) and not Equation (2) is because a request sent from IoT node i to fog node j could be received from fog node j' (offloaded to and processed at fog node j'). In this case the propagation and transmission delay from IoT layer to fog layer are X_{ij}^{IF} and Y_{ij}^{IF} respectively, but the propagation and transmission delays from fog layer to IoT layer are $X_{j'i}^{FI}$ and $Y_{j'i}^{FI}$.

Boundary case: Consider a domain $\mathcal{M}$ of fog nodes where $e_{\mathcal{M}} = 0$, which means that no forwarding is allowed. In this case if a request does not enter a fog node's queue, it will be offloaded to the cloud. In this case, $L_{ij} = P_j.(\overline{W}_j + X_{ji}^{FI} + Y_{ji}^{FI}) + (1 - P_j).[X_{jk}^{FC} + Y_{jk}^{FC} + \overline{H}_k + X_{ki}^{CI} + Y_{ki}^{CI}]$.

Special case: Consider a scenario where a fog node sends some sort of feedback message to the cloud, when processing tasks from IoT nodes (for example aggregated temperature reading for archival purposes). Note that this does not add any new term in the delay equations from the perspective of IoT nodes, because sending feedback messages to the cloud does not affect the service delay experienced by an IoT node. The fog node simply sends the feedback message to the cloud, while the IoT node's request is either processed or being processed by the fog node.

E. Acceptance Probability: P_j

P_j is the probability that a request is accepted by fog node j (enters the queue of fog node j) and is used in Equation (6). P_j depends on the queuing state of a fog node and the decision to accept or offload a request upon the time of receiving the request; in particular, if fog node j's estimated waiting time is greater than a threshold θ_j, it will offload the request to its best neighbor. Thus P_j is extended by the following probability:

$$\begin{aligned} P_j &= P[\text{request enters the queue at fog node } j] \\ &= P[\text{est. waiting time of fog node } j < \theta_j] = P[W_j < \theta_j]. \end{aligned} \tag{8}$$

F. Waiting Time in Fog and Cloud

An efficient method to update the estimated waiting time in a fog node is discussed in Section III-D. As discussed before, when fog node j needs to calculate the current estimated waiting time W_j, it multiplies the estimated processing time of each task type by the number of tasks of that type in the queue.

A similar approach to above is used to calculate $\overline{H}_k$ for the simulation or implementation: the cloud server multiplies the estimated processing time of each task type by the number of tasks of that type in the queue to compute the estimated waiting time. One can substitute in the equations, the estimated delay for handling the request at the cloud server k, $\overline{H}_k$ with the average delay for handling the request at all servers in domain $\mathcal{N}$, $H_{\mathcal{N}}$, when the value of estimated waiting time in the cloud servers are close to one another. $H_{\mathcal{N}}$ then can be obtained by

$$H_{\mathcal{N}} = \sum_{k \in S_{\mathcal{N}}^C} \omega_k \times \overline{H}_k, \tag{9}$$

where ω_k is the probability that a request from an IoT node goes to cloud server k, and $\sum_{k \in S_{\mathcal{N}}^C} \omega_k = 1$.

V. Simulation Results

In this section we evaluate the proposed mechanism through simulation. Each sample point in the graphs is obtained using 1 million requests using an event driven simulation. The network topology is a graph with 500 IoT nodes, 25 fog nodes, and 6 cloud servers. The IoT node either processes its own request, or sends it to its corresponding fog neighbor or to one of the cloud servers. If the request is sent to the fog layer, based on the proposed scheme, the request could be offloaded to other fog nodes or to the cloud. The topology of the fog nodes in the fog layer is generated randomly in each experiment using a random graph generator with average node degree of 3. IoT nodes are associated with the fog node which has the smallest distance (i.e. has the smallest propagation delay).

If an IoT node generates type Light requests (e.g. sensor), the communication between the IoT node and its corresponding fog node is assumed to be through IEEE 802.15.4, or NB-IoT, or ZigBee, in which the transmission rates are 250 Kbps. If the IoT node generates Heavy requests (e.g. traffic camera), the communication between the IoT node and its corresponding fog node is assumed to be through IEEE 802.11 a/g, and the transmission rate is 54 Mbps. The link rates between fog nodes in one domain are 100 Mbps and the link rates on the path from fog nodes to cloud servers are 10 Gbps.

The propagation delay between the IoT nodes and the fog nodes, among fog nodes, and between fog nodes and the cloud servers are uniformly distributed between U[1,2], U[0.5,1.2], and U[15,35] respectively (in ms). Request lengths are exponentially distributed with an average length of 100 bytes for light processing tasks, and 80 KB for heavy processing tasks. We assume that the length of the response is the same as the length of its corresponding request, on average.

To obtain realistic values for the processing ratio of IoT nodes to fog nodes, we looked at the processing capabilities of the Arduino Uno R3 microcontroller (example of IoT node generating Light requests) and an Intel dual-core i7 CPU (example of fog node). In the worst case, a fog node's processing capability is found to be around 3000 times faster than that of an IoT node that generates type Light requests ("Fog-to-IoT-Light ratio"), and 200 times faster than that of an IoT node that generates type Heavy requests ('Fog-to-IoT-Heavy ratio"). We also assume that a cloud server is 100 times faster than a fog node ("Cloud-to-Fog ratio"), on average, and that the average processing time of IoT node for Light and Heavy requests is 30 ms and 400 ms, respectively. Other simulation parameters are summarized in Table III. To account for the variation of values of the above parameters in real IoT-fog-cloud applications, we altered the parameters uniformly as: Fog-to-IoT-Light ratio, U[500,4000]; Fog-to-IoT-Heavy ratio, U[100,400]; and Cloud-to-Fog ratio, U[50,200]; we found that the result (average delay) fluctuates only by -0.77% to +5.51%.

TABLE III
Simulation Parameters

Fig.	p_i^I	p_i^F	b_i	θ_j	$e_{\mathcal{M}}$	γ_i	γ_i'	q
(a)	0	1	0.8	0.2	1	0.1	0.25	-
(b)	0	0.85	0.5	0.0002	-	0.5	0.6	0.5
(c) (d)	0.1	0.75	-	0.0002	1	0.05	0.005	0.5
(e) (f)	(0) (0.2)	-	0.9	0.0002	1	0.01	0.001	0.5

We compare the average service delay in three different modes of the IoT-fog-cloud operation. In No Fog Processing (NFP) mode, the IoT node either processes its own requests, or sends them to the cloud. In All Fog Processing (AFP) and Light Fog Processing (LFP) modes, the IoT node either processes its own requests or sends them to the fog or cloud. In AFP, both request types Light and Heavy can be sent to the fog layer, whereas in LFP, only type Light requests are sent to the fog layer. For even more detailed analysis, the delay for type Light and Heavy requests is plotted separately for the three modes in the color figures (Namely, AFP$_H$ and AFP$_L$). All time units are in ms.

We assume IoT node i generates type Light (or Heavy) requests according to a Poisson process, with rate γ_i (or γ_i'), depending on the type of IoT node i. Fig. 3a shows the average delay as a function of fairness factor q. $q \in (0, 1)$ is the fog fairness factor and its value depends on how the fog node selects jobs from its processing queue. In the *fair* case $q = 0.5$, a fog node simply selects the head of the queue. The closer the value of q is to 0 (or 1), the higher priority is given to type Heavy (or type Light) requests in the queue for processing. It can be seen that when q is closer to 1, more priority is given to light processing tasks, thus the delay of light processing tasks is decreased and the delay of heavy processing tasks is increased. Note that this change is only seen in AFP, as the fairness factor q is not defined in NFP (there is no fog) and LFP (all Light requests).

Fig. 3b shows the average service delay as a function of $e_{\mathcal{M}}$. For AFP, the optimal value of $e_{\mathcal{M}}$ where the service delay is minimum is achieved for $e_{\mathcal{M}} = 1$ using the mentioned parameters, and when $e_{\mathcal{M}} > 5$, AFP performs worse than NFP. It is interesting to see that changes in $e_{\mathcal{M}}$ do not change the average service delay in LFP, since the incurred transmission and propagation delay to offload a request among fog nodes is negligible for Light requests with small length.

Fig. 3c and 3d show the average service delay as a function of b_i (probability that a generated request at IoT node i is type Light). Fig. 3c shows that the average service delay for both Heavy and Light requests do not change notably when the percentage of Light and Heavy requests change. This is because we are looking at each of the task types separately, and this is oblivious to the effect of b_i. By comparing the delay of Light and Heavy processing tasks in Fig. 3c for the three modes, it is clear that the AFP is the best mode. Fig. 3d illustrates the interesting relationship between average service delay (of combined Light and Heavy requests) in three modes when the b_i changes. It can be seen that AFP in general outperforms LFP and NFP in terms of average service delay;

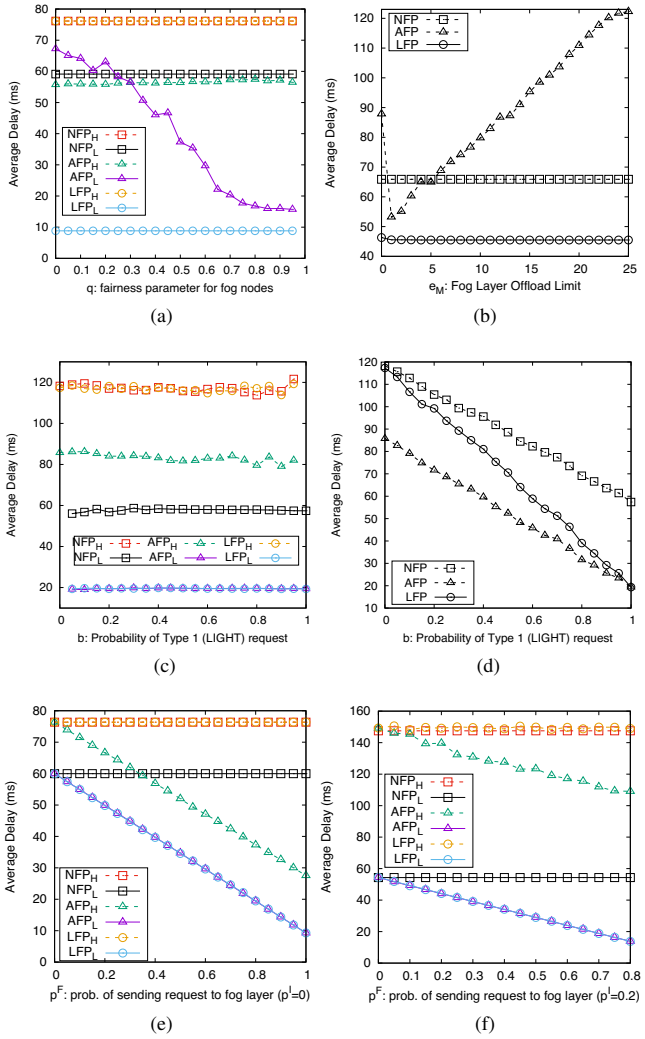

Fig. 3. Simulation results

however, when the percentage of Light requests in the network increases, LFP's performance gets closer to that of AFP. This effect is due to having fewer heavy processing tasks in the network, which make the average service delay larger.

Fig. 3e and 3f show the effects of p_i^I, p_i^F, and p_i^C on the average service delay. Both figures show how the average service delay is reduced under each policy when the probability of sending requests to fog nodes increases. In Fig. 3e, $\forall i : p_i^I = 0$ and it is clear that the performance of both LFP and AFP is better than that of NFP, as the delays in all the cases is lower. For Fig. 3f, $\forall i : p_i^I = 0.2$ and it can be seen that the overall delay has been increased, because of weak processing capabilities of IoT nodes. Yet, the overall delay is decreased to from 60 ms to 18 ms for light processing tasks, and from 150 ms to 117 ms, for heavy processing tasks. In this figure, it is also evident that the performance of LFP and AFP is better than that of NFP.

VI. Conclusion

The vision of fog computing is studied in this paper as a complement to cloud computing and an essential ingredient of the IoT. We introduced a framework for handling IoT request in the fog layer and an analytical model to formulate service delay in the IoT-fog-cloud scenarios. We showed how our delay-minimizing policy can be beneficial for IoT applications. Various numerical results are included to support our claims by showing how changes in parameters could affect the average service delay.

Our analytical model can support other fog computing policies. Suppose when decision to offload a task is not based on queueing status, one can replace P_j and L_{ij}, for instance, with the desired equations based on their policy. As future work, we plan to model the fog and the cloud using Markovian queueing systems and obtain closed-form equations for all components of our model. Moreover, it is interesting to see how cost can affect the minimizing service delay problem. In other words, studying the delay-cost tradeoff in the fog computing could be a potential research direction.

References

[1] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of things: A survey on enabling technologies, protocols, and applications," *IEEE Communications Surveys Tutorials*, vol. 17, no. 4, pp. 2347–2376, 2015.

[2] L. M. Vaquero and L. Rodero-Merino, "Finding your way in the fog: Towards a comprehensive definition of fog computing," *ACM SIGCOMM Computer Commun. Review*, vol. 44, no. 5, pp. 27–32, 2014.

[3] 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*, pp. 13–16, 2012.

[4] S. Yi, C. Li, and Q. Li, "A survey of fog computing: Concepts, applications and issues," in *Proceedings of the 2015 Workshop on Mobile Big Data*, pp. 37–42, ACM, 2015.

[5] P. Garcia Lopez, A. Montresor, D. Epema, A. Datta, T. Higashino, A. Iamnitchi, M. Barcellos, P. Felber, and E. Riviere, "Edge-centric computing: Vision and challenges," *SIGCOMM Computer Communication Review*, vol. 45, no. 5, pp. 37–42, October 2015.

[6] X. Guo, R. Singh, T. Zhao, and Z. Niu, "An index based task assignment policy for achieving optimal power-delay tradeoff in edge cloud systems," in *2016 IEEE International Conference on Communications (ICC)*, pp. 1–7, May 2016.

[7] L. Gu, D. Zeng, S. Guo, A. Barnawi, and Y. Xiang, "Cost-efficient resource management in fog computing supported medical cps," *IEEE Transactions on Emerging Topics in Computing*, no. 99, 2016.

[8] R. Deng, R. Lu, C. Lai, T. H. Luan, and H. Liang, "Optimal workload allocation in fog-cloud computing toward balanced delay and power consumption," *IEEE Internet of Things Journal*, vol. 3, no. 6, pp. 1171–1181, Dec 2016.

[9] 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*, no. 99, pp. 1–1, 2015.

[10] K. G. Shin and Y. C. Chang, "Load sharing in distributed real-time systems with state-change broadcasts," *IEEE Transactions on Computers*, vol. 38, no. 8, pp. 1124–1142, Aug 1989.

[11] K. G. Shin and C.-J. Hou, "Design and evaluation of effective load sharing in distributed real-time systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 5, no. 7, pp. 704–719, Jul 1994.

[12] M. Sharifi, S. Kafaie, and O. Kashefi, "A survey and taxonomy of cyber foraging of mobile devices," *IEEE Communications Surveys Tutorials*, vol. 14, pp. 1232–1243, Fourth Quarter 2012.

[13] B. Li, Y. Pei, H. Wu, and B. Shen, "Heuristics to allocate high-performance cloudlets for computation offloading in mobile ad hoc clouds," *Journal of Supercomputing*, vol. 71, no. 8, pp. 3009–3036, 2015.