Theoretical Modeling of Fog Computing: A Green Computing Paradigm to Support IoT Applications

Subhadeep Sarkar[†], *Student Member*, *IEEE*, Sudip Misra[‡], *Senior Member*, *IEEE*,

[†‡]School of Information Technology, [†]School of Medical Science and Technology

Indian Institute of Technology, Kharagpur, 721302, India

Email: [†]subhadeep@smst.iitkgp.ernet.in, [‡]smisra@sit.iitkgp.ernet.in

Abstract

In this paper, we focus on theoretical modeling of the *fog computing* architecture and compare its performance with the traditional cloud computing model. Existing research works on fog computing have primarily focused on the principles and concepts of fog computing and its significance in the context of *Internet of Things* (*IoT*). This work, one of the first attempts in its domain, proposes a mathematical formulation for this new computational paradigm by defining its individual components and presents a comparative study with cloud computing in terms of service latency and energy consumption. From the performance analysis, the work establishes fog computing, in collaboration with the traditional cloud computing platform, as an efficient green computing platform to support the demands of the next generation IoT applications. Results show that for a scenario where 25% of the IoT applications demand real-time, low-latency services, the mean energy expenditure in fog computing is 40.48% less than the conventional cloud computing model.

Index Terms

Fog computing, Green computing, Theoretical modeling, Internet of Things, Cloud computing

I. Introduction

Cloud computing, in recent years, has added a new dimension to the traditional means of computation, data storage, and service provisioning [1]–[3]. However, the rapid increase in the number of ubiquitous mobile and sensing devices which are connected to the Internet challenges the traditional network architecture of the cloud computing framework. As reported by Cisco, by the year 2020, the next generation Internet-connected devices, often termed as *Internet of Things* (*IoT*) would comprise of around 50 billion Internet-connected devices [4]. A large number of devices are located at the edge of the network require support for mobility and low latency, real-time, and location-aware services [7]. Cisco proposed a new computational paradigm, termed as *fog computing*, that subdues the shortcomings of cloud computing by transferring some of the core functions of cloud towards the network edge.

Fog computing is a distributed computing paradigm, that empowers the network devices at different hierarchical levels with various degrees of computational and storage capability [30]. Bonomi *et al.* [5], [27] discussed and characterized fog computing, and justified its suitability for the services and applications of IoT. Some other works [9]–[13] on fog computing

have also highlighted the advantages of fog computing over the traditional cloud computing. In [7], Hong *et al.* designed a programming model for mobile fog computing to serve large scale IoT applications. Few works [22], [23] have addressed the security and privacy issues associated with fog computing. Resource allocation within the fog computing paradigm [10], [14]–[16] has also been explored at a preliminary level. A practical implementation of fog computing using Raspberry Pi [26] was attempted by Krishnan *et al.*. However, none of these works focuses on modeling of the fog computing architecture which is of pivotal importance from an implementation perspective. Motivated by the works of Misra *et al.* [17] and Dong *et al.* [6], in this paper, we develop a theoretical model to mathematically represent the fog computing architecture by defining its individual components. Also, we provide a comparative study of fog computing with traditional cloud computing, and analyze the impact of this new computational paradigm based on certain parameters in the context of IoT applications.

A. Motivation

The conventional cloud computing architecture is completely centralized in nature, and therefore, fails to provide real-time and low latency services to billions of IoT devices at the edge of the network, simultaneously. The fact that the cloud framework thrives on the idea of *virtualization of services* [8], [19] can be cited as the root behind this major shortcoming of cloud computing. In the process of virtualization, the cloud service providers (CSPs) render their users with services from geo-spatially remote locations, and thereby, invoking high latency in service provisioning. Therefore, the need for a new computational paradigm to support the traditional cloud computing model in handling the requirements of these latency-sensitive IoT applications is imminent.

It should be clearly mentioned, that fog computing is not a substitute of cloud computing, rather these two technologies complement one another. The complementary functions of cloud and fog enable the users to experience a new breed of computing technology that serves the requirements of the real-time, low-latency IoT applications running at the network edge [24], and also supports complex analysis and long-term storage of data at the core of the network.

B. Paper organization

The remaining of the paper is organized as follows. Section II provides a comprehensive outline of the fog computing architecture. In Section III, the mathematical modeling of the system is presented. We design the different performance metrics in Section IV, and analyze the performance of the fog computing paradigm in contrast with that of the traditional cloud computing framework in Section V. In Section VI, we discuss few practical applications of the fog computing paradigm before finally concluding the work in Section VII.

II. Fog Computing Architecture

As stated earlier, fog computing is a non-trivial extension of cloud computing [5], and typically serves as a platform that bridges numerous sensing devices situated at the network edge to the core computing structure of the cloud.

A. Assumptions

Strictly stating, at this early stage of research fog computing is yet to be realized in a large and practical scale. Therefore, in this work, we draw few simplified, yet realistic assumptions.

- The terminal nodes (TNs) in the network, such as mobile phones, smart vehicles, and smart meters are aware of and able to transmit their absolute geo-spatial location through technologies such as GPS, GIS, or GNSS.
- The fog computing tier comprises of 'intelligent' devices which are capable of computing, processing, and storing data in addition to routing and forwarding the data-packets to the upper tier [7].
- The networking devices in the fog computing layer (see Fig. 1) are able to share the network, computational, and storage load among themselves as per requirement.
- The fog computing devices are able to provide optimal support for mobility of the TNs.

B. System Outline

A generic fog architecture can be thought as a three tier network structure [28], as shown in Fig. 1:

(a) *Tier 1*: This is the bottom-most layer encompasses all the TNs (IoT devices), which are responsible for sensing of a multitude of events and transmitting the raw sensed data to its immediate upper layer in the hierarchy.

(b) *Tier 2*: The middle layer, also known as the fog computing layer, comprises of devices, such as routers, gateways, switches, and access points, which are intelligent enough to process, compute, and temporarily store the received information. These fog computing devices are connected to the cloud framework, and are responsible for sending data to the cloud on a periodic basis.

(c) *Tier 3*: The cloud computing layer is the upper-most layer in this architecture. The layer constitutes of multiple high-end servers and data centers which are capable of processing and storing an enormous amount of data.

C. Architecture Details

The tier closest to the ground builds a network of several sensor-equipped, Internet-connected end-devices, often termed as IoT. The data transmitted by these TNs are received by the edge gateways present at the border of the *fog tier*. Unlike the traditional cloud architecture, in fog computing not every data packet is redirected to core cloud computing module for processing. Instead, all real-time analysis and latency-sensitive applications are run on the fog layer itself. The fog computing devices in this layer have limited semi-permanent storage that allows them to store the received data temporarily for analysis and then send the source devices the needful feedbacks.

The geo-spatially close TNs are grouped together to form a virtual cluster (VC). These VCs are assumed to have an injective (one-to-one) mapping with the fog instances (FIs). An FI is conceptualized specific to a geographic location, and the VC comprising of all the Internet-connected TNs within that location is served by the co-located FI. Based on its location a TN may leave and join any VC, and get disconnected and connected to the corresponding FI. The FIs are assumed to be capable of self-adaptation according to the demand-load.

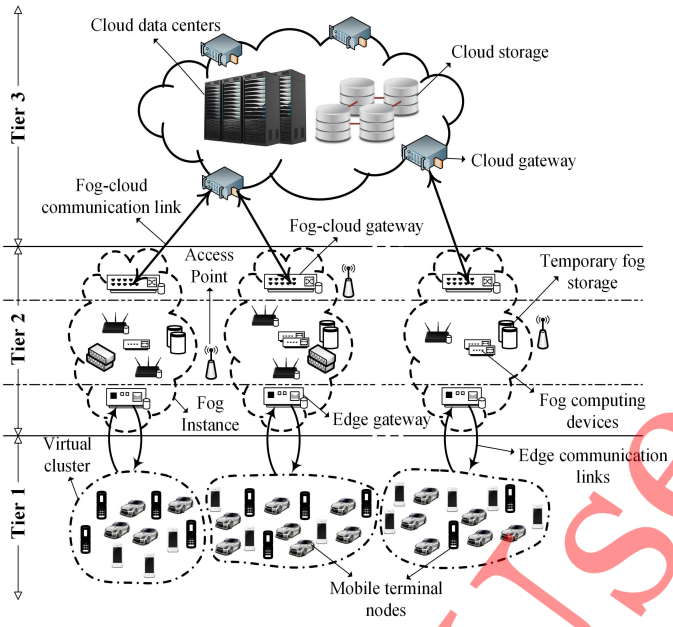


Fig. 1: Fog computing architecture

The cloud computing tier is responsible for permanent and large-scale storage of data and computationally extensive analysis of the global data-set. Unlike the traditional cloud computing architecture, instead of getting blasted for every small query and for storage of every relevant or irrelevant chunk of data, in fog computing, the cloud core module is now accessed in a periodic and controlled manner, leading to efficient and improved utilization of the cloud resources.

III. Mathematical Model of the System

In this Section, we mathematically describe the fog computing architecture by defining its composite entities and the operational functions involved. While formulation of the mathematical model, it is assumed that the total number of TNs spanning across all VCs in tier 1 is constant over time. Also, the VCs provide complete coverage for all the TNs present in the lowest tier. We define the important physical and virtual components of the fog computing architecture below.

Definition 1. *(Terminal node)* A terminal node[1] (TN), denoted by $\mathcal{T}$, is defined as a 6-tuple:

$$\mathcal{T} =< T_{id}, T_{st}, \tau_i, \mathcal{L}, \mathcal{H}, \mathcal{I}[q] >$$

where, T_{id} is an integer representing the unique ID of the TN, and the rest of the tuples are defined as follows.

Definition 2. *(Status of a terminal node)* A status of a TN, T_{st}, defines whether the node is in active state or not, and is represented as a boolean: $T_{st} = \{0, 1\}$, where the values 0 and 1 symbolize the *inactive* and *active* states, respectively.

Definition 3. *(Type of terminal node)* The type of a TN (τ_i) indicates the type of event that a node senses. Mathematically, it is presented an element of the set $\tau = \{\tau_1, \tau_2, \ldots, \tau_p\}$, where τ denotes set of all events being monitored by the TNs, and p

[1]The set of terminal nodes is indicated by $\widetilde{T}$. Henceforth, the set of entities are denoted using this notation, all throughout the paper.

is the total number of distinct types of events.

Definition 4. *(Location of a terminal node)* The geo-spatial location of a TN is defined as a 4-tuple:

$$\mathcal{L} =< l_x, l_y, l_z, t_s >$$

where, l_x, l_y, and l_z represent the longitude, latitude, and altitude of a TN, respectively. t_s denotes the time-stamp at which the node is transmitting its location.

Property 1. *The belonging of a TN in a VC at time t, is independent of its belonging at time instant $t-1$.*

The tuple $\mathcal{H}$ dictates the specifications of a TN which includes its hardware details, along with its mode of operation, operational frequency, and sampling rate.

Definition 5. *(Specifications of a terminal node)* The specifications of a TN ($\mathcal{H}$) is represented as a 6-tuple:

$$\mathcal{H} =< \mathcal{P}, \mathcal{M}, \mathcal{B}, \mathcal{S}, c, f >$$

where, $\mathcal{P}$ denotes the processor specifications of the node which includes details such as processing core speed, bus specifications, and internal register (cache memory) size. The primary memory (RAM) specifications, such as memory size, memory clock, and data rate are stored in $\mathcal{M}$. The tuple $\mathcal{B}$ describes the battery details, viz. voltage, size (AA or AAA), type (Ni or C electrodes), and the number of the battery-cells required. $\mathcal{S}$ is the symbolic representation of the different types of sensors that are used as sub-modules of the node. The tuple c denotes the hardware used for wireless communication for the node, such as Bluetooth and ZigBee. The frequency range in which the TN operates is denoted by f.

The last tuple in the definition of a TN, $\mathcal{I}[q]$, is an 1-D array (with q elements) that stores the instance IDs of the application instances which are running on the device. It may be noted that although applications have independent existences, it is impossible for an application instance to exist without the presence of a parent device. We define the terms *application* and *application instance* accordingly, as below.

Definition 6. *(Application)* An application $\mathcal{A}$ is defined as a 4-tuple:

$$\mathcal{A} =< A_{id}, A_{type}, A_{sp} >$$

where, A_{id} is the application ID and A_{type} denote the purpose for which the application is used (such as medical, education, finance, entertainment, utility, and gaming). A_{sp} dictates the minimum system specifications that are required to run the application including the processor, primary memory, and secondary storage details and the operating system version.

Definition 7. *(Application instance)* The instance of an application, $\mathcal{I}$, is defined as a 5-tuple:

$$\mathcal{I} =< I_{id}, A_{id}, T_{id}, I_{st}, I_{req} >$$

where, I_{id} is the application instance ID which can be thought of as the process ID generated by the system. A_{id} and T_{id} bear

the same meanings as mentioned earlier. I_{st} is a boolean variable, such that $I_{st} = \{0, 1\}$, where the values 0 and 1 are indicative of the application instance being idle (not inactive) and active, respectively. The final tuple I_{req} is the resource requirement of the application instance. This resource requirement may be in terms of network bandwidth (for streaming applications), computation and analysis ability (for medical applications), or storage and processing power (for gaming applications). Multiple instances of an application may concurrently run on a TN, and are distinguished by their unique instance IDs.

Having defined all the physical components of tier 1, now we define the term *virtual cluster*.

Definition 8. *(Virtual cluster)* A virtual cluster (VC), denoted by $\mathcal{V}$, corresponds to a logical boundary comprising of several localized TNs, and is mathematically defined as a 4-tuple:

$$\mathcal{V} =< V_{id}, \mathcal{T}[u], R, F_{id} >$$

where, V_{id} is the ID of the VC, and $\mathcal{T}[u]$ is a non-empty 1-D array of size u that stores the IDs of all the constituent TNs. It is evident that the array should have a dynamic length, where the array length, at any given time instant, is indicative of the number of TN that are present in the VC, and the length changes as some mobile TNs leave or join the cluster. The region monitored by a VC which encloses all the intermediate TNs is denoted by R. It is mentionworthy that for efficient monitoring of all the TNs we have divided the geographic terrain into multiple non-overlapping regions – each region mapped to a VC. The physical FI to which a VC is mapped is referred to by the fog instance ID, F_{id}.

Property 2. *The mapping from the set of VCs to the set of FIs, represented as: $f(\cdot) : \widetilde{\mathcal{V}} \rightarrow \widetilde{\mathcal{F}}$ is injective.*

We now define the term fog instance and the composite fog computing modules which together constitute the middle tier.

Definition 9. *(Fog instance)* A fog computing instance or simply a fog instance (FI), denoted by ($\mathcal{F}$), is mathematically defined as a 3-tuple:

$$\mathcal{F} =< F_{id}, C_{AP}, \mathcal{D}[v] >$$

where, F_{id} is the unique ID of the fog computing instance, and C_{AP} is the access point through which the FI is connected to the core cloud computing framework. The third tuple, $\mathcal{D}[v]$ is a non-empty 1-D array of size v which stores the device IDs of all the constituent fog computing devices of a FI.

Property 3. *The mapping from the set of TNs to the set of FIs, denoted as: $f'(\cdot) : \widetilde{\mathcal{T}} \rightarrow \widetilde{\mathcal{F}}$ is many-to-one.*

Definition 10. *(Fog computing device)* We define a fog computing device, $\mathcal{D}$, in terms of its type and characteristic features in the form of a 3-tuple:

$$\mathcal{D} =< D_{id}, D_{type}, D_{sp} >$$

where, D_{id} and D_{type} are the ID and the type (such as gateway, router, processing unit, or storage) of the fog computing device. The hardware and related specifications of the device is stored in the D_{sp} tuple.

Now that we have defined all the components of the fog computing architecture, we analyze the demand of a VC before its parent FI. The demand function, $\Omega(\cdot)$, is defined as:

$$\begin{aligned}\Omega(\mathcal{V}) &= \sum_{i} \Omega(\mathcal{T}_i), \forall i = 1(1)u \\ &= \sum_{i}\sum_{q} \Omega(\mathcal{T}_i \succ \mathcal{I}_j), \forall i = 1(1)u, \forall j = 1(1)q \end{aligned} \tag{1}$$

The operator $\succ$ is used to indicate successor relationship between a pair of operands. For instance, $X \succ Y$ indicates Y is a successor of X. Also, from Definition (7), we get that $\Omega(\mathcal{T}_i \succ \mathcal{I}_j) = \mathcal{T}_i \succ \mathcal{I}_j \succ I_{req}, \forall i, j$. Based on this demand function services are granted by the fog tier to the different application instances running within the TNs. Only in special cases, where intervention of the cloud core is mandatory, and for periodic update purpose the cloud computing layer is accessed.

Proposition 1. *The pairwise intersection of the VCs is null.*

Proof: We prove this by the method of contradiction. We assume, $\exists \mathcal{V}_i, \mathcal{V}_j$, such that $\mathcal{V}_i \bigcap \mathcal{V}_j \neq \Phi$. Thus,

$$\exists \mathcal{T}_k \text{ such that, } \mathcal{T}_k \in \mathcal{V}_i, \mathcal{V}_j \Rightarrow \mathcal{T}_k \in \mathcal{V}_i \bigcap \mathcal{V}_j \tag{2}$$

As per Property 2,

$$f'(\mathcal{T}_k) = \mathcal{F}_p, \mathcal{F}_p \in \widetilde{\mathcal{F}} \Rightarrow f^{-1}(\mathcal{F}_p) = \{\mathcal{V}_i, \mathcal{V}_j\} \tag{3}$$

which is absurd as per injectivity of Property 1. Thus, our assumption is invalid. This concludes the proof. ■

Proposition 2. *At any given time instant, $|\widetilde{\mathcal{T}}| = |\mathcal{V}_1| + |\mathcal{V}_2| + \ldots + |\mathcal{V}_m|$, where $|\widetilde{\mathcal{T}}|$ denotes the total number of TNs present at tier 1, $|\mathcal{V}_i|$ denotes the number of TNs mapped to the i^{th} VC, and m is the total number of VCs present in the system.*

Proof: We prove the above by the method of contradiction. We assume,

$$|\widetilde{\mathcal{T}}| \neq |\mathcal{V}_1| + |\mathcal{V}_2| + \ldots + |\mathcal{V}_m| \tag{4}$$

This implies, there exists atleast a pair $(\mathcal{V}_i, \mathcal{V}_j)$, such that

$$\mathcal{V}_i, \mathcal{V}_j \in \widetilde{\mathcal{V}}, \mathcal{V}_i \bigcap \mathcal{V}_j \neq \Phi \tag{5}$$

which contradicts with Proposition 1, and disapproves our assumption. This concludes the proof. ■

Proposition 3. *The mapping $g(\cdot)$ of the set of all TNs to the set of VCs is surjective.*

Proof: As per Properties 1, and 2, it is intuitive that $\forall \mathcal{T}_i \in \widetilde{\mathcal{T}}$, $\exists \mathcal{V}_j \in \widetilde{\mathcal{V}}$, such that $g(\mathcal{T}_i) = \mathcal{V}_j$. Therefore, $\forall \mathcal{V}_j \in \widetilde{\mathcal{V}}$, $g^{-1}(\mathcal{V}_j) = K \subseteq \widetilde{\mathcal{T}}$. Now, if $g^{-1}(\mathcal{V}_j) = \Phi$, length of $\mathcal{V}_j \succ \mathcal{T}[u]$ is 0. However, as per Definition 8, $\mathcal{T}[u]$ is non-empty. Thus, for every $\mathcal{V}_j$, there exists atleast a single pre-image $\mathcal{T}_i$. This concludes the proof of surjectivity for $g(\cdot)$. ■

IV. Performance Metrics

In this Section, we define the performance metrics for fog computing based on which the analysis and comparison of this model against the traditional cloud computing model is made.

A. Service Latency

Service latency for a request sent by an application instance running within an IoT device is basically its response time, and is computed as the sum of the transmission latency and the processing latency for the request. We assume that communication among multiple FIs at tier 2, and that among different cloud data-centers at tier 3 take place over high-bandwidth channel – leading to negligible delay. Also, the communication delay among the TNs is taken to be insignificant [20]. Let δ_{tf} and δ_{fc} be the delays in transmission of a data packet from a TN to the corresponding FI, and from an FI to the cloud data-center, respectively. Thus, the mean transmission latency, δ_{fog}, for the data packets of N_i application instances running within $\mathcal{V}_i$ is given by:

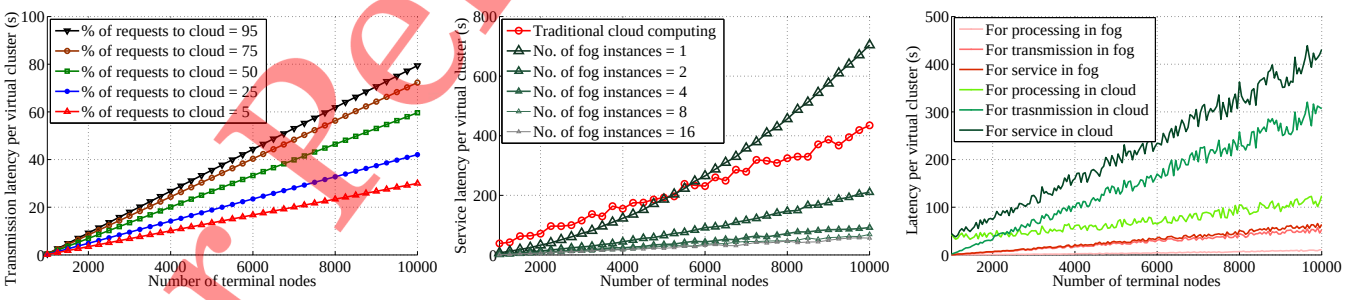
$$\delta_{fog} = \left[\delta_{tf} B_i + \delta_{fc} b_i + \delta_{fc} b_i^r + \delta_{tf} b_i^r + \delta_{tf} (B_i - b_i)^r\right] / B_i \tag{6}$$

where, B_i and b_i ($B_i > b_i$) are the total number of packets sent, cumulatively, by N_i application instances to $\mathcal{F}_i$, and from $\mathcal{F}_i$ to the cloud data centers, respectively. X^r indicates the total number of data packets that are sent as a response to X request data packets. The mean transmission latency for an application instance ($\mathbb{D}_{fog}^{tr}$) request is given by:

$$\mathbb{D}_{fog}^{tr} = \frac{\delta_{tf} \sum_{i=1}^{w} \left[B_i + b_i^r + (B_i - b_i)^r\right] + \delta_{fc} \sum_{i=1}^{w} \left[b_i + b_i^r\right]}{\sum_{i=1}^{w} B_i} \tag{7}$$

In a traditional cloud computing environment, the expression for the same would be:

$$\mathbb{D}_{cloud}^{tr} = \frac{\delta_{fc} \sum_{i=1}^{w} \left[B_i + B_i^r\right]}{\sum_{i=1}^{w} B_i} \tag{8}$$



(a) Transmission latency vs number of TNs (b) Service latency vs number of TNs (c) Service latency: Fog vs Cloud

Fig. 2: Analysis of service latency

Processing latency for an application instance request is computed in terms of the number of requests that are processed at the server end prior to its service. We assume that at time t, N_i number of application instances are running within $\mathcal{V}_i$. Thus,

for a total of w VCs the total number of application instances served at the same time is $N = \sum_{i=1}^{w} N_i$. $\Delta(\mathcal{V}_i, \mathcal{I}_i)$ is the service delay of an application instance, $\mathcal{I}_i$ running within $\mathcal{V}_i$, served by $\mathcal{F}_i$. Out of N_i application instances we assume n_i ($N_i > n_i$) instances are redirected the core cloud computing module for processing. Intuitively, the total number of application instances processing at the cloud-end at time t is $n = \sum_{i=1}^{w} n_i$, and the processing latency at the cloud end for each of these n application instances is denoted as $\nabla(n)$. We obtain the mean processing latency of an application instance running within $\mathcal{V}_i$ as:

$$\bar{\Delta}_{fog}(\mathcal{V}_i, \mathcal{I}_i) = \big[N_i \Delta(\mathcal{V}_i, \mathcal{I}_i) + n_i \nabla(n)\big]/N_i \tag{9}$$

Now, as we take into consideration all the VCs $(\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_w)$ present in tier 1, the expression for mean service latency $(\mathbb{D}^{sr}_{fog})$ boils down to:

$$\mathbb{D}^{sr}_{fog} = \frac{\sum_{i=1}^{w} \bar{\Delta}_{fog}(\mathcal{V}_i, \mathcal{I}_i)}{w} \tag{10}$$

In contrast, in a general cloud computing model, all N application instances running at the user-end, directly communicate with the core computing module, and require its constant involvement. The mean processing latency of an application instance request $(\mathbb{D}^{sr}_{cloud})$ in this case is given by:

$$\mathbb{D}^{sr}_{cloud} = \frac{\sum_{i=1}^{w} N_i}{N} = \nabla(N) \tag{11}$$

B. Energy Consumption

The energy expended due to transmission of unit byte of data from the bottom tier to the middle tier, and from the middle tier to the cloud tier are denoted by γ_{tf} and γ_{fc}, respectively. The energy required to process unit byte of data within the fog and cloud computing tiers are denoted by α_{fog} and α_{cloud}, respectively. In a fog computing environment, the rate of energy dissipation due to transmission and processing of data packets is, therefore, expressed as:

$$\begin{aligned}\Gamma_{fog}(t) = \Big[\gamma_{tf} \sum_{i=1}^{w}\sum_{j=1}^{t} \Lambda_i(j) + \gamma_{fc} \sum_{i=1}^{w}\sum_{j=1}^{t} \lambda_i(j)\Big]/t + \\ \alpha_{fog} \sum_{i=1}^{w}\sum_{j=1}^{t} \Lambda_i(j) + \alpha_{cloud} \sum_{i=1}^{w}\sum_{j=1}^{t} \lambda_i(j)\end{aligned} \tag{12}$$

where $\Lambda_i(j)$ and $\lambda_i(j)$ (where $\lambda_i(j) > \lambda_i(j)$) be the total number of bytes being transmitted from $\mathcal{V}_i$ to $\mathcal{F}_i$ and from $\mathcal{F}_i$ to the cloud core at $t = j$, respectively.

On the other hand, in traditional cloud computing framework, every byte of data are transmitted to the computing core for analysis and storage. The corresponding energy dissipation rate due to transmission and processing of the data is, thereby,

given by the following equation.

$$\Gamma_{cloud}(t) = \left[\gamma_{tc} \sum_{i=1}^{w} \sum_{j=1}^{t} \Lambda_i(j)\right]/t + \alpha_{cloud} \sum_{i=1}^{w} \sum_{j=1}^{t} \Lambda_i(j) \tag{13}$$

where γ_{tc} demotes the energy required to transfer unit byte of data from the bottom tier to the cloud data centers.

V. Performance Evaluation

We perform an analysis of the fog computing paradigm based on the metrics designed in Section IV, and compare its performance against traditional cloud computing. We consider a system with 10 FIs connected to a single CSP. The TNs are assumed to be uniformly distributed among the VCs, the data generation rate from each TN being 1 packet/s. Length of each data packet is taken as 65536 bytes and the machine instruction size is assumed to be 64 bits. Processing speed of the devices at the fog computing tier and the cloud data centers are taken as 1256 MIPS (ARM Cortex A5) and 124850 MIPS (Intel Core i7 4770k), respectively. Also, the energy required to transmit a single data byte is taken to be 20 nJ, whereas, the processing energy is assumed to be 10 J/GB data.

Service latency

The transmission latency of a data packet is based on the round trip time between two terminals, and is computed as $rtt(ms) = 0.03 \times distance(km) + 5$ [18]. We vary the percentage of applications which require to access the cloud computing core, and plot the cumulative transmission latency for all the nodes within a VC against variable number of TNs. As shown in Fig. 2(a), with the increase of the number of TNs present at the lowest tier, the cumulative transmission latency increases with a linear slope. Also, as the percentage of applications routed towards the CSP increases, the transmission latency is observed to increase.

In Fig. 2(b), we observe that as the number of FIs decreases there is a significant rise in service latency, and except for one case (where number of FIs = 1), in all other cases the service latency is found to be less than that in a traditional cloud computing environment. However, when there is only a single FI present in the system, it boils down to a bi-layered cloud computing architecture, which yields a higher service latency as compared to that in conventional cloud computing due to the additional layer overhead.

To provide a thorough contrast of the service latency for processes running in fog computing (number of FIs = 8) and cloud computing environments, we plot the transmission and processing latencies along with the total service latency for both these environments in Fig. 2(c).

Energy Consumption

In Fig. 3, we study that in a fog computing platform, as the percentage of requests that are required to be redirected to the cloud computing core is increased, the overall upload cost also increases. It is observed that, the energy expended due to transmission for the fog computing architecture is observed to be lower than that for cloud computing. However, it can be

fairly concluded that if the number of low-latency IoT applications are significantly less, the energy consumption in case of fog computing will be higher than its counterpart.

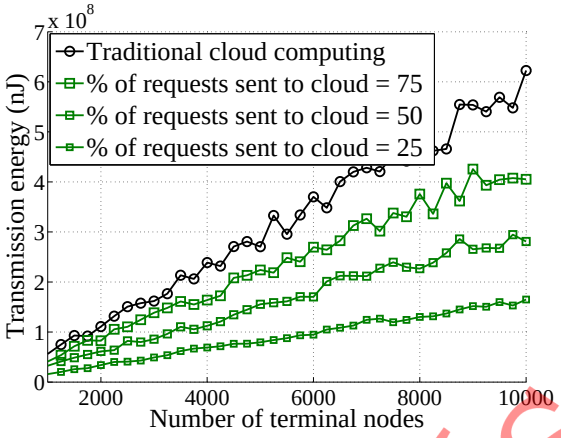


Fig. 3: Upload cost vs number of TNs

The energy expended due to processing at the computing tier is measured as the energy expended per hour to process the requests sent to the computing core. In Fig. 4, it is observed that as the number of requests referred to the cloud increases, the processing energy increases almost linearly. In the context of IoT applications, with approximately 25% of requests requiring real-time services, the fog computing architecture is observed to improve the mean energy consumption by 40.48%.

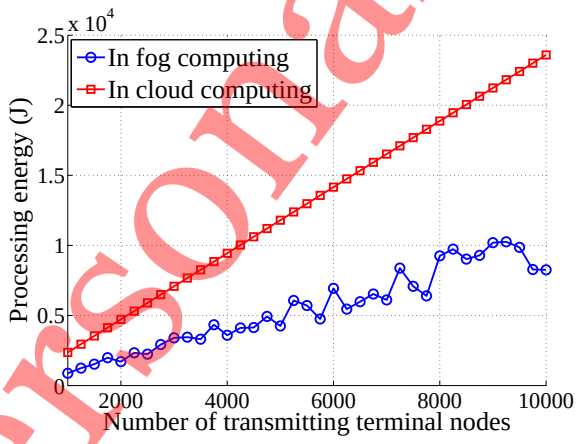


Fig. 4: Server cost vs number of transmitting TNs

VI. Application Specific Case Studies

In this Section, we discuss the real-life applicability of the fog computing paradigm coupled with the traditional cloud computing framework, and discuss few of its practical implementations. We present a generic flow diagram in Figure 5 which illustrates the service flow for a real-life application being served by the fog computing framework.

A. Smart Vehicle Management

Smart vehicle management is basically a technological fusion of connected vehicle and smart traffic management. The connected vehicular system thrives on real-time communication between the on-road vehicles and the road-side access points.

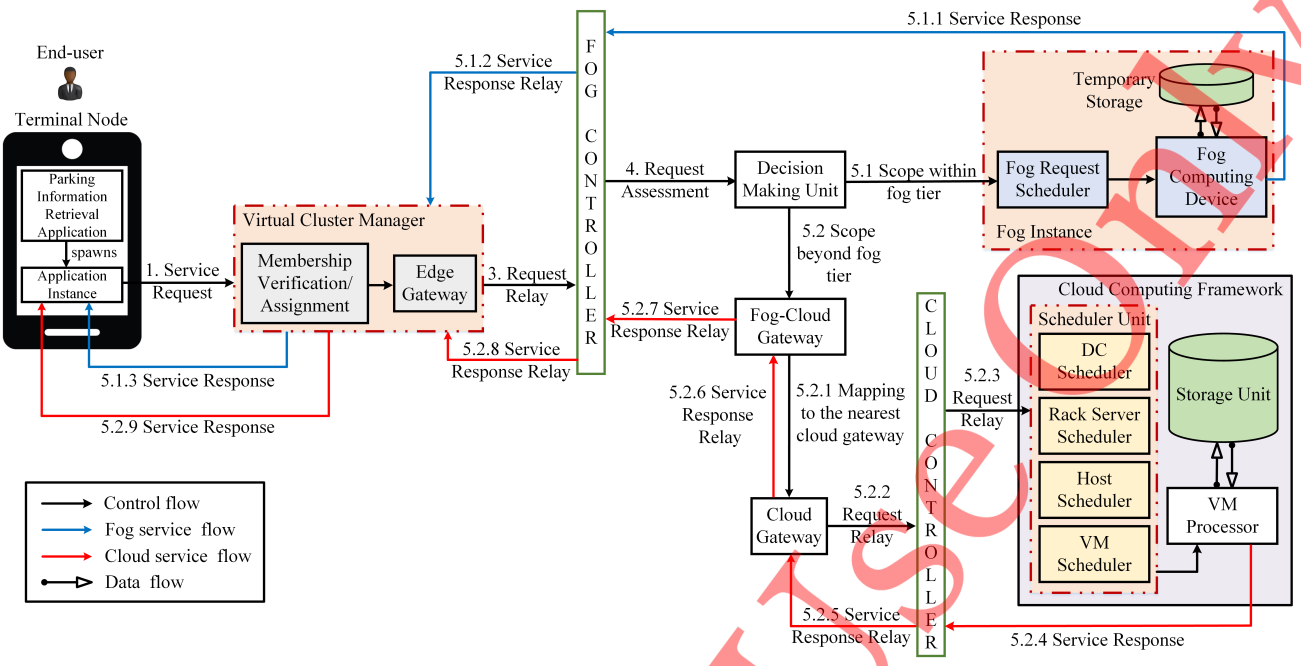


Fig. 5: Flow diagram of real-life application service in a fog computing framework

These communications essentially take place over WiFi, 3G, 4G, or LTE [25], [29] infrastructure. The vehicles are equipped which multiple sensors which assess the road conditions, traffic congestion and send the data to a local server. These data-feeds are analyzed in real-time and crucial location-aware information are provided to the vehicles. Sensor-equipped parking lots which provides *a priori* information about the availability and location of the parking slots are also a part of this connected vehicular system. In a fog computing framework, the location-aware data analysis would take place within the co-located FI and low-latency service provisioning is done from the fog tier without the intervention of the cloud computing tier.

Smart traffic management system, the smart traffic lights interact with the sensors which detect and estimate the traffic flow on the roads and manages its light-cycle adaptively [5]. The FIs are responsible for managing the traffic lights within a particular region. Also, interaction between the neighboring FIs are required to maintain smooth traffic flow across regions. While all the real-time latency-sensitive computations and analyses are managed by the fog computing tier, the consolidate information are later sent to the cloud computing core for long-term analysis of the global traffic conditions.

B. Smart Grids

Smart grids serve as another example which requires real-time processing with very low latency. The lowest tier comprises of heterogeneous electronic devices, such as electronic home appliances, industrial machinery, computer servers, and plug-in hybrid electric vehicles (PHEVs), which draw energy from the grid. The smart meters govern the electricity consumption for each of these devices and needs to send high volumes of data continuously over time. The fog computing tier, in this scenario, could act as a local data sink maintaining the demand-supply and consumption details, and take care of all transactions within the region. These data are later send to the cloud computing tier for permanent storage, necessary aggregation, and global analysis [21]. By introducing the intermediate fog computing tier, the load on the core cloud computing module is greatly

diminished and services are provided from the fog computing tier without imposing any additional latency.

VII. Conclusion

In this work, we theoretically modeled the fog computing architecture, and analyzed its performance in the context of IoT applications. It was observed that for a system with large number of real-time, low latency IoT applications running, the service latency associated with fog computing was significantly lower than that with cloud computing. Also, the rates of energy dissipation due to transmission of data bytes to the computing cores and subsequent analysis were noticed to be considerably low. It should be reiterated that fog computing is not a substitute of cloud computing; rather anticipating the next generation IoT applications and their huge demand of real-time services, fog computing, in collaboration with the traditional cloud computing model, will serve as a greener computing platform. Finally, our future works involve characterization of fog computing from the perspective of resource management and virtualization and extension in the context of big data analysis involving Internet of Everything (IoE).

References

[1] Armbrust M., Fox A., Griffith R., *et al.*: 'A view of cloud computing', ACM Communications Magazine, 2010, 53, (4), pp. 50–58

[2] Rimal B. P., Choi E., and Lumb I.: 'A taxonomy and survey of cloud computing systems', 5^{th} International Joint Conference on INC, IMS and IDC, Seoul, South Korea, August 2009, pp. 44–51

[3] Lai C.-F., Wang H., Chao H.-C., and Nan G.: 'A Network and Device Aware QoS Approach for Cloud-Based Mobile Streaming', IEEE Transactions on Multimedia, 2013, 15, (4), pp. 747–757

[4] MarketWatch, 'Cisco delivers vision of fog computing to accelerate value from billions of connected devices', http://www.theiet.org/resources/journals/research/index.cfm, accessed August 2014

[5] Bonomi F., Milito R., Zhu J., and Addepalli S.: 'Fog computing and its role in the internet of things', Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing (ACM), Helsinki, Finland, August 2012, pp. 13–16

[6] Dong H., Hao Q., Zhang T., and Zhang B.: 'Formal discussion on relationship between virtualization and cloud computing', International Conference on Parallel and Distributed Computing, Applications and Technologies, Wuhan, China, December 2010, pp. 448–453.

[7] Hong K., Lillethun D., Ramachandran U., Ottenwälder B., and Koldehofe B.: 'Mobile fog: A programming model for large-scale applications on the internet of things', Proceedings of the Second ACM SIGCOMM Workshop on Mobile Cloud Computing, Hong Kong, China, August 2013, pp. 15–20

[8] Liu N., Li X., and Wang Q.: 'A resource & capability virtualization method for cloud manufacturing systems', IEEE International Conference on Systems, Man, and Cybernetics, Anchorage, USA, October 2011, pp. 1003–1008

[9] Madsen H., Albeanu G., Burtschy B., and Popentiu-Vladicescu F. L.: 'Reliability in the utility computing era: Towards reliable fog computing', 20^{th} International Conference on Systems, Signals and Image Processing, Bucharest, Romania, July 2013, pp. 43–46

[10] Nishio T., Shinkuma R., Takahashi T., and Mandayam N. B.: 'Service-oriented heterogeneous resource sharing for optimizing service latency in mobile cloud', Proceedings of the First International Workshop on Mobile Cloud Computing & Networking, MobileCloud, Bangalore India, July 2013, pp. 19–26

[11] Stolfo S. F., Salem M. B., and Keromytis A. D.: 'Fog computing: Mitigating insider data theft attacks in the cloud', IEEE Symposium on Security and Privacy Workshops, San Francisco, USA, May 2012, pp. 125–128

[12] Preden, J. S., Tammemae, K., Jantsch, A., Leier, M., Riid, A., and Calis, E.: 'The Benefits of Self-Awareness and Attention in Fog and Mist Computing', Computer Magazine, 2015, 48, (7), pp. 37–45

[13] Yi, S., Li, C., and Li, Q.: 'A Survey of Fog Computing: Concepts, Applications and Issues', ACM Proceedings of the 2015 Workshop on Mobile Big Data, Hangzhou, China, June 2015, pp. 37–42

[14] Do, C. T., Tran, N. H., Pham C., Alam, M. G. R., Son J. H., and Hong C. S.: 'A proximal algorithm for joint resource allocation and minimizing carbon footprint in geo-distributed fog computing', International Conference on Information Networking, Cambodia, January 2015, pp. 324–329

[15] Aazam, M. and Eui-Nam H.: 'Dynamic resource provisioning through Fog micro datacenter', IEEE International Conference on Pervasive Computing and Communication Workshops, St. Louis, USA, March 2015, pp. 105–110

[16] Aazam, M. and Eui-Nam H.: 'Fog Computing Micro Datacenter Based Dynamic Resource Estimation and Pricing Model for IoT', IEEE 29^{th} International Conference on Advanced Information Networking and Applications, Gwangiu, South Korea, March 2015, pp. 687–694

[17] Misra S., Chatterjee S., and Obaidat M. S.: 'On theoretical modeling of sensor-cloud: A paradigm shift from wireless sensor network', IEEE Systems Journal, 2014, DOI: 10.1109/JSYST.2014.2362617, pp. 1–10

[18] Qureshi A.: 'Power-Demand Routing in Massive Geo-Distributed Systems', PhD thesis, MIT, 2010

[19] Sotomayor B., Montero R. S., Llorente I. M., and Foster I.: 'Virtual infrastructure management in private and hybrid clouds', IEEE Internet Computing, 2009, 13, (5), pp. 14–22

[20] Zhang L., Wu C., Li Z., Guo C., Chen M., and Lau F. C. M.: 'Moving big data to the cloud: An online cost-minimizing approach', IEEE Journal on Selected Areas in Communications, 2013, 31, (12), pp. 2710–2721

[21] Stojmenovic, I. and Sheng W.: 'The Fog computing paradigm: Scenarios and security issues', Federated Conference on Computer Science and Information Systems, Warsaw, Poland, September 2014, pp. 1–8

[22] Dsouza, C., Ahn, G.-J., and Taguinod, M.: 'Policy-driven security management for fog computing: Preliminary framework and a case study', IEEE 15^{th} International Conference on Information Reuse and Integration, Redwood City, USA, August 2014, pp. 16–23

[23] Kulkarni, S., Saha, S., and Hockenbury, R.: 'Preserving privacy in sensor-fog networks', 9^{th} International Conference for Internet Technology and Secured Transactions, London, UK, December 2014, pp. 96–99

[24] Yannuzzi, M., Milito, R., Serral-Gracia, R., Montero, D., and Nemirovsky, M.: 'Key ingredients in an IoT recipe: Fog Computing, Cloud computing, and more Fog Computing', Athens, Greece, December 2014, pp. 325–329

[25] Zhu J., Chan, D.S., Prabhu, M.S., Natarajan, P., Hu H., and Bonomi, F.: 'Improving Web Sites Performance Using Edge Servers in Fog Computing Architecture', IEEE 7^{th} International Symposium on Service Oriented System Engineering, Redwood City, USA, March 2013, pp. 320–323

[26] Krishnan, Y. N., Bhagwat, C. N., and Utpat, A. P.: 'Fog computing — Network based cloud computing', 2^{nd} International Conference on Electronics and Communication Systems, Coimbatore, India, Febriary 2015, pp. 250–251

[27] Bonomi F., Milito R., Natarajan P., and Zhu J.: 'Fog Computing: A Platform for Internet of Things and Analytics', in Bessis N. and Dobre C. (Ed.): 'Big Data and Internet of Things: A Roadmap for Smart Environments - Part I' (Springer International Publishing, Switzerland, 2014, Volume 546), pp. 169–186

[28] Stojmenovic, I.: 'Fog computing: A cloud to the ground support for smart things and machine-to-machine networks', Australasian Telecommunication Networks and Applications Conference, Southbank, Australia, November 2014, pp. 117–122

[29] Vaquero, L. M. and Rodero-Merino, L.: 'Finding Your Way in the Fog: Towards a Comprehensive Definition of Fog Computing', ACM SIGCOMM Computer Communication Review, 2014, 44, (5), pp. 27–32

[30] Preden, J., Kaugerand, J., Suurjaak, E., Astapov, S., Motus, L., and Pahtma, R.: 'Data to decision: pushing situational information needs to the edge of the network', IEEE International Inter-Disciplinary Conference on Cognitive Methods in Situation Awareness and Decision Support, Orlando, USA, March 2015, pp. 158–164