Latency-Aware Horizontal Computation Offloading for Parallel Processing in Fog-Enabled IoT

Pallav Kumar Deb, *Graduate Student Member, IEEE*, Sudip Misra, *Senior Member, IEEE,* and Anandarup Mukherjee, *Graduate Student Member, IEEE*

***Abstract*—In this paper, we propose a two-step distributed horizontal architecture for computation offloading in a fog-enabled Internet of Things (IoT) environment – HD-Fog – to minimize the overall energy consumption, and latency while executing hard real-time applications. The HD stands for the horizontal distribution of the tasks in the fog layer. Each sensor in the user devices independently captures data of varying formats. Parallel execution on these data is possible based on its Directed Acyclic Task Graph (DATG), and the corresponding results facilitate the ease of decision making. Towards this, in HD-Fog, the sensor nodes in user devices offload their tasks to a nearby fog node based on a greedy selection criterion. This fog node then further offloads the smaller sub-tasks, based on the DATG, among other fog nodes for parallel execution. Through extensive real-life metric-based emulation and comparison against traditional Fog and Cloud computing schemes, we observe that our approach 1) reduces the overall operational delays by 29% and 96%, and 2) offers promising speedup values. The proposed HD-Fog scheme also indicates a reduction in energy consumption by 30% compared to traditional fog computing schemes.**

***Keywords*—Computation Offloading, Fog Computing, Distributed and Parallel Computing, Queueing Theory, IoT**

I. Introduction

Smart user devices depend on multiple sensors like cameras, radars, proximity sensors, and others, which produce a massive amount of data. Data from each of these sensors need complex processing for deriving inferences. Offloading such execution routines to external platforms such as the Fog is beneficial for real-time operations. However, existing schemes only focus on the selection of one Fog Node (FN) and executing the tasks there in its entirety [1]. The tasks from IoT Sensor Nodes (SNs) contain sub-tasks with interdependencies in the form of Directed Acyclic Task Graphs (DATGs) [2]. Simultaneous execution of the ones at the same level in the DATG may reduce the processing delay significantly.

In this work, we propose a two-step task distribution scheme named, Horizontally Distributed Fog Computing (HD-Fog), such that SNs with Radio Access Network (RAN) technology offload their high-level tasks to one of the FNs within their vicinity, which further redistributes the sub-tasks to neighboring FNs considering their current states. Complex optimization techniques for device selection usually overburdens the SNs and FNs as they are typically resource-constrained in terms of *computation capability*, *battery power* and *storage capacity*. Moreover, with increasing participants (nodes), the convergence time increases exponentially. For fast and straightforward execution in robust environments, we propose a horizontal task distribution architecture and formulate a greedy scheme for taking decisions in real-time.

P. K. Deb, S. Misra, and A. Mukherjee are with the Department of Computer Science and Engineering, Indian Institute of Technology Kharagpur, India. e-mail: (pallv.deb,sudipm)@iitkgp.ac.in, anandarupmukherjee@ieee.org

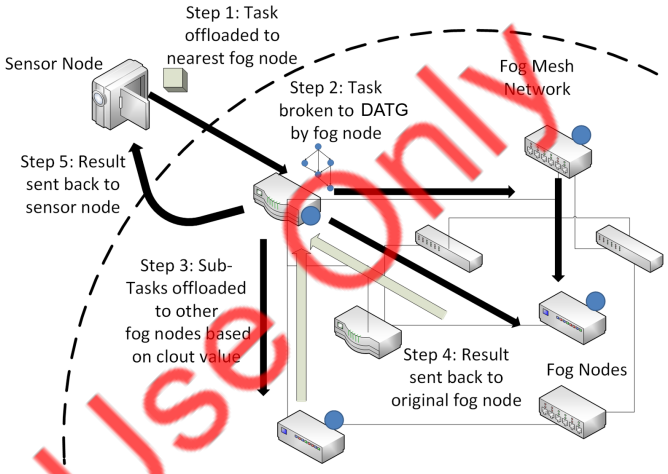


Fig. 1: Overview of the proposed HD-Fog scheme

Example Scenario: An IoT user device consists of a heterogeneous set of sensors like cameras, radars, proximity sensors, lidars, and others (Fig. 1). For instance, data from smart cars need to detect objects of interest and each of these operations is independent of one another, opening the scope for parallel execution. Representation of these operations in the form of DATGs offers simplicity and help in accommodating the possibility of interdependent execution. HD-Fog offloads the sub-tasks on the same level (in DATG) to different FNs for simultaneous execution, which helps in saving significant time. In the case of dependent tasks, the FNs wait for the results.

A. Motivation

The nature of tasks in an IoT environment is hard real-time and cannot afford delays in reacting to the data. While most of the existing solutions rely on offloading tasks to external platforms for execution in their entirety in one single location [1], it is advantageous to break the task into smaller indivisible sub-tasks and form DATGs to introduce parallelism. Such parallel processing saves a lot of time and energy compared to the conventional methods of sequential processing. The possibility

of saving time, energy, and natural resources serves as a motivation for designing our architecture and scheme. Further, as complex optimization computations take significant time to converge with the increasing number of nodes and that the SNs and FNs are devoid of high computational resources, we formulate the selection of the FNs for offloading on a greedy basis, instead of sophisticated optimization techniques.

B. Contributions

In this work, we propose a horizontal computation offloading scheme in the fog layer for enabling parallel processing when possible. The key contributions in this work are as follows:

- ***Horizontal Task Distribution:*** We propose and design a two-step horizontal task distribution architecture for the SNs to offload their tasks to the FNs. These FNs identify the DATG and redistribute the sub-tasks among other FNs (horizontally) for parallel execution.
- ***Parallel Execution:*** We enable parallel execution of the sub-tasks on the same level, saving significant time.
- ***Greedy Formulation:*** We formulate a greedy selection scheme (using principal FN parameters) instead of other complex optimization techniques to avoid delays/overheads in hard real-time applications.
- ***Watchdog:*** Over multiple iterations, the FNs may start misbehaving due to unprecedented issues like power cut, operational overload, and others. To account for their recoveries, we design a watchdog-based alarm system for the FNs based on the neighboring FN's experiences.

It may be noted that we refrain from considering channel access and congestion parameters to make this work more focussed and plan to address these issues in our extended work. Approaches such as network slicing [3] may be helpful for optimized RAN access. Further, [2] provides a code-based technique for creating DATGs.

II. Related Work

Significant effort has been made in load balancing in constrained devices in the case of *Mobile Cloud, Edge,* and *Fog Computing*. The devices at the edge can help in complementing the services of the remote cloud. A theoretical analysis of the impact on the performance of IoT devices caused by policies that offload user tasks to edge computing servers can be found in [4]. While most work focus on finding optimal placement of the components of a task, Sardellitti *et al.* [5] considered a multi-cell mobile edge-computing scenario with dense deployment of radio access points and jointly optimized radio and computational resources. The authors in [6] also jointly optimized radio and computational resources for offloading from smartphones to femto-cloud. As these schemes help in avoiding the devices from getting overloaded and enhances reliability, energy-aware decisions further expand lifespan [7]. Wei *et al.* [8] proposed a cache-aware computation offloading scheme in cloud, reducing response time.

Although the term Fog is relatively new, the concept has been in the literature for quite some time for complementing the cloud services using devices at the edge [9]. For instance, The authors in [10] proposed a Fog-based scheme for predicting wildfires in agricultural fields. Wang *et al.* [11] proposed a learning-based approach for offloading tasks in the fog layer within constrained deadlines. However, as the number of users increases, the time complexity to reach equilibrium grows exponentially, which led to the formulation of another near-optimal resource allocation scheme. Jiang *et al.* [12] proposed an energy-efficient task offloading scheme in Fog-enabled IoT environments. Similarly, the authors in [13] presented a Reinforcement Learning-based offloading scheme with a primary focus towards energy harvesting on the edge. Chen *et al.* [14] also proposed an energy-aware offloading scheme to the cloud, with a special focus on industrial scenarios.

Synthesis: Task offloading in Cloud, Edge, and Fog Computing is a matured field of study among researchers. These solutions typically focus on energy, resource, and time-aware offloading of the tasks to external platforms. They also focus on mobility-aware task offloading by predicting the positions of the user devices. However, these solutions only consider executing the tasks on one location in its entirety. Limited work exists that considers tasks as DATGs and offloads them as sub-tasks in a heterogeneous environment in a horizontal fashion. While most of the work relies on centralized agents and vertical task offloading, we propose a decentralized horizontal architecture for dynamic redistribution of the sub-tasks.

III. System Model

Fig. 1 depicts the working of the HD-Fog scheme considering two types of devices: 1) ***Fog Nodes (FNs):*** These are network edge devices that offer computation services in addition to their regular networking services. These devices mostly include *gateways, hubs, switches, bridges, routers,* and others [15], and 2) ***Sensor Nodes (SNs):*** These are resource-constrained devices onboard a user device that monitor (sensors) and act (actuators) when needed. *Cameras, automated machinery, temperature, humidity, light sensors* are some of the examples of SNs, among many others.

A. The Horizontal Architecture

As shown in Fig. 1, the FNs share information among one another via messages using a strong backhaul mesh network. The SNs connect to one of these FNs (within its vicinity) using RAN technology (Step 1). On receiving the task, the corresponding FN identifies sub-tasks and their corresponding DATG, for redistribution among the other available FNs (Step 2 and 3) for parallel execution. We consider our scenario to be pseudo-static so that the SNs get the results back from the FN chosen initially (Step 4 and 5). In the future, we plan to consider scenarios where the SNs will be independent of mobility restrictions. Also, we make the following realistic *assumptions*: 1) We consider that the FNs simultaneously execute communication as well as computation routines. 2) As we refrain from addressing channel access parameters, we consider a lossless network.

B. Solution Approach

In our system, $U = \{u_1, u_2, u_3, ..., u_{\mathcal{M}}\}$ and $V = \{v_1, v_2, v_3, ..., v_{\mathcal{N}}\}$ are the sets of heterogeneous SNs and FNs present in the network. The FNs share their computation power, storage, available power and distance related information through periodic beacon signals. We consider J_i as the i^{th} SN's two tuple high-level task, $< I_i, D_i >$, where I_i is the total size and D_i is the total number of cycles necessary for executing the entire task. We normalize the configuration parameters for a device x, and define them as follows. *Computation Power* is the number of cycles it can perform per unit time, and is the ratio of the device's current computational capability (C_x^{dev}) and the the maximum achievable value (C_x^{max}) as $\mathbb{C}_x^{norm} = (C_x^{dev}/C_x^{max})$. *Residual Storage* is the ratio of the device's current available storage (S_x^{dev}) and the maximum capacity value (S_x^{max}) represented as $\mathbb{S}_x^{res} = (S_x^{dev}/S_x^{max})$. *Waiting Time* is the duration a device has to wait for the task execution, inclusive of transmission ($W_x^{J_i}$), the waiting in the FN's queue (W_x^{queue}) and return (W_x^{result}) times. We compute the waiting time as ratio of the sum of these times and the maximum tolerable delay as: $\mathbb{W}_x^{norm} = (W_x^{total}/W_x^{max})$ where, $W_x^{total} = W_x^{J_i} + W_x^{queue} + W_x^{result}$. We consider M/M/c for the FNs and M/M/1 for the SNs for W^{queue} and calculate it as $1/\rho - \lambda$ where ρ and λ are the execution and arrival rates, respectively. If the value of $\mathbb{W}_x^{norm}$ results to be greater than 1, we do not consider the FN for offloading. *Distance* is the stretch between a device x and the FNs or other SNs within their vicinity and calculate it as the ratio between the distance ($d_x^{dev_i}$) and the transmission range (d_i^{trans}) of the SNs as: $\mathbb{D}_x^{norm} = (d_x^{dev_i}/d_i^{trans})$. *Residual Power* is the power (in units) available for use, we calculate it as the ratio of the device's current available power ($P_{u_i}^{dev}$) and the the maximum capacity value ($P_{u_i}^{max}$) as: $\mathbb{P}_{u_i}^{res} = (P_{u_i}^{dev}/P_{u_i}^{max})$. *Reliability* is the measure of trust based on neighboring devices' past experiences as a ternary variable, that is, it can take values -1 for unknown reliability, 0 for unreliable devices and 1 for reliable devices. We explain the manner in which the devices update reliability of other devices in section III-D. We formulate a utility function for the SNs from which they greedily choose FNs and term it as *Clout* from here onwards. It may be noted that in typical IoT environments, vendors provide abundant supply of power to the FNs and hence we do not consider $\mathbb{P}_{v_j}^{res}$ for FN selection.

***Proposition* 1.** *The clout for offloading tasks by i^{th} SN to an FN v_j is calculated as a function of the FN's $\mathbb{C}_{v_j}^{norm}$, $\mathbb{S}_{v_j}^{res}$, $\mathbb{W}_{v_j}^{norm}$ and $\mathbb{D}_{v_j}^{norm}$. In case of offloading to the SNs, we use the same factors along with $\mathbb{P}_{u_i}^{res}$. Mathematically,*

$$\mathcal{C}_{v_j}^{dev_i} = (1 + \gamma_{v_j}) \left\lceil \frac{\mathbb{C}_{v_j}^{norm} \mathbb{S}_{v_j}^{res}}{\mathbb{W}_{v_j}^{norm} \mathbb{D}_{v_j}^{norm}} \right\rceil, \text{ in case of FNs} \quad (1)$$

$$\mathcal{C}_{u_k}^{dev_i} = (1 + \gamma_{u_k}) \left\lceil \frac{\mathbb{C}_{u_k}^{norm} \mathbb{S}_{u_k}^{res} \mathbb{P}_{u_k}^{res}}{\mathbb{W}_{u_k}^{norm} \mathbb{D}_{u_k}^{norm}} \right\rceil, \text{ in case of SNs} \quad (2)$$

where $k \in \mathcal{M}$ such that $k \neq i$, γ_x represents reliability and is treated as a constant, x in γ_x is u_i for SNs, and v_j for FNs, respectively.

Justification: As $\mathbb{C}_x^{norm}$ rises, the devices perform faster and execute more number of tasks. Thus, $C_x^{dev_i} \propto \mathbb{C}_x^{norm}$; Devices accept more tasks as well as larger tasks for computation with larger $\mathbb{S}_x^{res}$. Thus, $C_x^{dev_i} \propto \mathbb{S}_x^{res}$; Due to low delay tolerance in hard real-time scenarios, $C_x^{dev_i} \propto 1/\mathbb{W}_x^{norm}$; With lower distance, the SNs disseminate less power and also save $W_x^{J_i}$ which is desirable. Thus, $C_x^{dev_i} \propto 1/\mathbb{D}_x^{norm}$. We conclude,

$$\mathcal{C}_{v_j}^{dev_i} = (1 + \gamma_{v_j}) \left\lceil \frac{\mathbb{C}_{v_j}^{norm} \mathbb{S}_{v_j}^{res}}{\mathbb{W}_{v_j}^{norm} \mathbb{D}_{v_j}^{norm}} \right\rceil, \text{ in case of FNs}$$

where, γ_{v_j} represents reliability of FN v_j and works as a proportionality constant. In case of SNs, we consider an additional parameter, that is ($P_{u_i}^{res}$). SNs with more power have the likelihood of successfully executing the tasks and returning results. Thus, $C_x^{dev_i} \propto \mathbb{P}_x^{res}$, implying:

$$\mathcal{C}_{u_k}^{dev_i} = (1 + \gamma_{u_k}) \left\lceil \frac{\mathbb{C}_{u_k}^{norm} \mathbb{S}_{u_k}^{res} \mathbb{P}_{u_k}^{res}}{\mathbb{W}_{u_k}^{norm} \mathbb{D}_{u_k}^{norm}} \right\rceil, \text{ in case of SNs} \quad (3)$$

C. Horizontally Distributed Fog Computing Scheme

HD-Fog provides a decentralized platform for the SNs to offload their tasks independently. Before choosing an external FN, the SN must decide whether to offload or to perform the required computations locally. We introduce weights (ω_{u_i}) where $0 \leq \omega_{u_i} \leq 1$, such that they can be varied to determine the mode of execution (local/remote). For instance, we set the value of $\omega_{u_i} = 1$ for local computation on-board the SN and activate execution/performance mode (if battery $\mathbb{P}_{u_i}^{res}$ permits). Otherwise, the SNs enter power saving mode by setting ω_{u_i} values close to 0. These weights allow the SNs to make autonomous decisions. In this work, we calculate ω_{u_i} as a function of the residual energy as $\omega_{u_i} = \mathbb{P}_{u_i}^{res}/\mathbb{P}_{u_i}^{max}$ We multiply these weights with the clout values, that is, $\omega_{u_i} \mathcal{C}_{local}^{dev_i}$ for local execution and $(1 - \omega_{u_i}) \mathcal{C}_x^{dev_i}$ for external execution. The SN offloads its task to the external platform only when the clout value of offloading along with the weights is greater than or equal to that of the local value, that is $(1 - \omega_{u_i}) \mathcal{C}_x^{dev_i} \geq \omega_{u_i} \mathcal{C}_{local}^{dev_i}$. On the decision of offloading, the SN needs to select an efficient FN. Each SN maintains a four-column hash table where the first column refers to the ID of the FN; The second column holds the clout values for the respective FNs; The third column holds the reliability values; The fourth column is binary-valued initially set to 0 and changes to 1 for reflecting the device to which the SN decides to offload.

The devices fill their tables based on periodic beacon signals from each device. We consider 4 tuple beacons which contain the *computation power, residual storage, waiting time, and timestamp*. We use the timestamp for calculating the distance of the FN from the SN. For simplicity, we consider the distance is embedded in the FN. Thus, we represent our beacon as: $< \mathbb{C}_{v_j}^{norm}, \mathbb{S}_{v_j}^{res}, \mathbb{W}_{v_j}^{norm}, \mathbb{D}_{v_j}^{norm} >$. It may be noted that the SNs consider only those FNs that may complete execution within tolerable time (deadline). Both FNs and SNs do not consider the ones that do not satisfy the condition. The SNs calculate the clout values and store it in the table mentioned above. The SN selects the device that has the maximum clout

value in the hash table. Suppose the hash table is represented by H and the elements as $H[dev][col]$. The fourth column is subjected to change as follows:

$$H[x][4] = \begin{cases} 1, \text{if } \mathcal{C}_x^{dev} > \mathcal{C}_y^{dev} \\ 0, \text{otherwise} \end{cases} \quad (4)$$

where x is the FN corresponding to the row in H and y represents all other devices, and $\sum_{m=1}^{\mathcal{M}+\mathcal{N}-1} H[m][4] = 1$ where $x, y \in U, V$ and $x \neq y$. We use this for ensuring that each SN does not select multiple FNs for offloading.

In the future, the SNs that wish to provide computational services will generate similar beacons. However, their beacons are 5 tuple as they also have the power constraint. Thus, $< \mathbb{C}_{u_k}^{norm}, \mathbb{S}_{u_k}^{res}, \mathbb{W}_{u_k}^{norm}, \mathbb{D}_{u_k}^{norm}, \mathbb{P}_{u_k}^{res} >$ represents the beacons.

Algorithm 1 Selection of FN by SN u_k

INPUTS:

- $\mathbb{C}_{v_j}^{norm}$: Computation power of FN v_j.
- $\mathbb{S}_{v_j}^{res}$: Residual storage of FN v_j.
- $\mathbb{W}_{v_j}^{norm}$: Waiting time of FN v_j.
- $\mathbb{D}_{v_j}^{norm}$: Distance of FN v_j from SN u_k.
- γ_{v_j}: Reliability of FN v_j.

OUTPUT: $\mathbb{V}_{u_k}$: FN with maximum clout value.

1: **for** i = 1 to $\mathcal{N}$ **do** ▷ $\mathcal{N}$: Total no. of FNs
2: **if** $\mathbb{D}_{v_i}^{norm} \leq$ Communication range of u_k **then**
3: Compute $\mathcal{C}_{v_i}^{dev_{u_k}}$ according to Equation (1)
4: Update H
5: **if** $\mathcal{C}_{v_i}^{dev_{u_k}} \geq \mathcal{C}_{max}^{dev_{u_k}}$ **then**
6: Update device selection in H
7: Set $\mathbb{V}_{u_k} = v_i$
8: **end if**
9: **else**
10: Perform computation locally
11: **end if**
12: **end for**

D. Reliability

HD-Fog scheme has a reliability system where the neighboring devices maintain the reputation of a particular FN. In order to achieve this, the devices in the HD-Fog scheme have four major components: *monitor*, *reputation manager*, a *reputation table* for recording the reliability values, and an *alarm*. The monitor acts like a watchdog that keeps observing the nature of nearby devices that offer services. Since it is not feasible to keep records of all the devices, we limit the observation to only those that are in the range of RAN. The reliability of the devices is a result of self-experience and feedback from others. The data from the monitor passes on to the reputation manager. The reputation manager analyses the table which contains a misbehavior count, and the reliability is lowered or improved in comparison with a predefined threshold. The reliability of the devices may be scaled to a value in the range of 0-10, and categorize as: 0-3: Unreliable, 3-6: Moderately reliable, 6-9: Reliable, and 9-10: Highly reliable. The reliability factor is increased on successfully executing 5-10 tasks (subjective) and decremented by 0.5 in case of each failure. Fuzzy-based methods may also prove beneficial in determining the reliability of the devices. Whenever the reliability value changes, the alarm component broadcasts a message containing the information indicating the increase/decrease of the reliability of services by a particular device. It may be noted that we assume that the fog nodes do not demonstrate failures midway of its execution and we plan to address this issue in our extended work. However, since the FNs transmit periodic beacons, it is relatively straightforward to detect FNs that are not transmitting any. Under such circumstances, the task/subtask may be reassigned to some other FN (considering delays due to reassignment are under the deadlines). The reliability factor may be used to assume the trust on the reviving FN.

Algorithm 2 HD-Fog (on FN selected in Algorithm 1)

INPUTS:

- $\mathbb{J}_i$: Task from i^{th} SN.
- $\mathbb{C}_{v_j}^{norm}$: Computation power of FN v_j.
- $\mathbb{S}_{v_j}^{res}$: Residual storage of FN v_j.
- $\mathbb{W}_{v_j}^{norm}$: Waiting time of FN v_j.
- $\mathbb{D}_{v_j}^{norm}$: Distance of FN v_j from FN v_k.
- γ_{v_j}: Reliability of FN v_j.

OUTPUT: Set of FNs for execution.

1: **for** j = 1 to $\mathcal{K}$ **do** ▷ $\mathcal{K}$: Total no. of sub-tasks
2: **for** i = 1 to $\mathcal{N}$ **do** ▷ $\mathcal{N}$: Total no. of FNs
3: Compute $\mathcal{C}_{v_i}^{dev_{u_k}}$ according to Equation (1)
4: Update H
5: **if** $\mathcal{C}_{v_i}^{dev_{u_k}} \geq \mathcal{C}_{max}^{dev_{u_k}}$ **then**
6: Update device selection in H
7: Set $\mathcal{V} = v_j$
8: **end if**
9: **end for**
10: **if** depth (j^{th} sub-task) = depth ($j-1^{th}$ sub-task) **then**
11: Execute $j^{th} sub-task$ immediately at $\mathcal{V}$
12: **else**
13: Execute $j^{th} sub-task$ after dependencies of previous depth at $\mathcal{V}$
14: **end if**
15: **end for**

E. Task Distribution for Parallel Execution

We describe the task distribution and its parallel execution with the help of an example. The SN initially selects an FN for offloading the task (refer Algorithm 1). The SN calculates the $\mathcal{C}_{v_i}^{dev_{u_k}}$ (line 4) for the relevant FNs (that may execute within deadlines) within its communication range (line 3) using the normalized parameters in Section III-B and updates H (line 5). It then selects the one with the maximum value (lines 6-8) for offloading. In case no FN exists within its vicinity, the SN executes the task locally (line 11). Algorithm 2 represents the horizontal distribution of the sub-tasks in the fog layer. For each level of the DATG, the FN calculates the $\mathcal{C}_{v_i}^{dev_{u_k}}$ for all relevant FNs in the network and updates its hash table (lines 2-8). If the sub-tasks are on the $j-1^{th}$ level, the FN starts execution immediately (lines 9-10). Otherwise, it waits

for the execution of the previous dependencies and executions for starting (lines 11-12). We maintain the dependencies of the sub-tasks by simultaneously executing those that are at the same depth/level of the DATG. Each sub-task starts after receiving the results from its parent. Finally, the FN in Algorithm 1 returns the result to the SN.

TABLE I: Simulation Parameters

Parameter	Value
SN computation capability	$0.1, 0.2, 0.3\ GHz$
FN computation capability	$0.8, 1.0, 1.5\ GHz$
Arrival Rate at FN	$188.391, 299.985,$ $427.429\ MIPS$
Service Rate of FN	$1256, 2000, 2850\ MIPS$
Cloud computation capability	$3.9\ GHz$
Task size	$420 - 12600\ KB$
Cycles required	$1000 - 30000$
SN communication range	$30 - 50\ m$
Shannon Capacity	$30\ KBps$
Energy to transfer 1B data	$20\ nJ$
FN Power for processing	$2\ W$

IV. Performance Evaluation

We compare our results based on the following benchmark solutions and setups:

Traditional Cloud Computing [1]: We consider a server at the center of our simulation area that accepts tasks from the IoT SNs and returns the result on process completion. Since we are not considering packet drops and collisions, this central server indefinitely accepts all requests from the SNs. It may be noted that the scheme proposed in [1] does not consider the interdependencies of the sub-tasks and perceive them as independent ones.

Traditional Fog Computing [16]: We consider the selection of FNs based on the same parameters as in HD-Fog using Equation 1. However, we do not redistribute the sub-tasks in this case. The chosen FN executes the entire task. In case the FNs exhaust the number of cores, the FN is not available for offloading, and the SN has to select some other FN.

Simulation Design: We perform an analysis of the HD-Fog scheme based on the metrics designed in Table I, and compare its performance against Traditional Cloud and Fog Computing schemes. We consider that the SNs communicate using WiFi technology. Inspired from Sarkar *et al.* [17], we consider ARM Cortex A5, A7, and A8 as FNs and Intel Core i7 4770k as a cloud data center. We consider that the SNs and FNs are randomly distributed over a geographical region of area $1000 \times 1000m^2$. We study the overall latency endured with varying number of SNs with a constant number of FNs. To establish a detailed insight into the operational latencies, we decompose and present the overall latency into transmission and processing latencies. In this work, we create a set of high-level tasks in the form of DATGs and record the results. The nodes in the DATGs are the sub-tasks, and the edges connecting them are the dependencies among them.

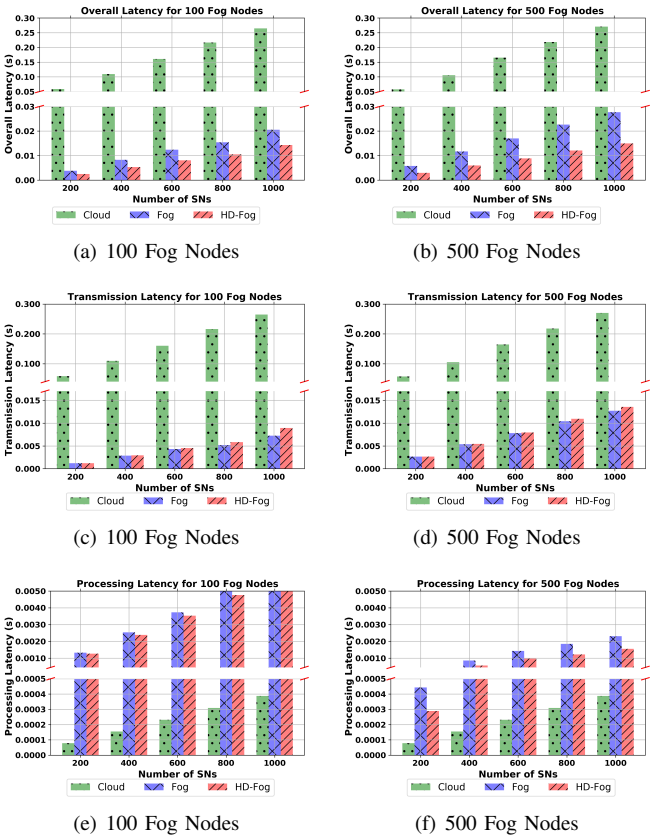
Packet Overhead and Energy Consumption: We capture the number of beacon packets in our simulation using NS3 for a period of 45 seconds. We arbitrarily select two devices and list the values in Table II. Typical WiFi has a beacon interval of 100 ms. We vary our beacon intervals from 1 ms to 5 ms. As expected, the number of packets decreases as we increase the interval. With lower intervals, although the network gets congested (12683 packets), the devices get the latest information about the others, implying the existence of a tradeoff. It may be noted that the difference in the number of packets in the devices is because the devices may enter the network on different time instants. Moreover, the energy consumption is proportional to the number of beacon packets. We consider the packets of size 32 B while calculating the energy consumption. In the worst case, the devices need 8.12 mJ. We infer from these observations that we need to set the beacons such that we consider minimizing the energy and network resource consumption while maintaining the latest information of the devices in the network.

TABLE II: Number of Packets (NoP) overhead and Energy Consumption (EC) due to periodic beacons

Beacon interval	Device 1		Device 2	
	NoP	EC	NoP	EC
1.02 ms	10743	6.88 mJ	12683	8.12 mJ
2.04 ms	5369	3.44 mJ	6342	4.06 mJ
3.07 ms	3581	2.29 mJ	4229	2.71 mJ
4.09 ms	2686	1.72 mJ	3171	2.03 mJ
5.12 ms	2149	1.38 mJ	2537	1.62 mJ

Overall Latency: We define this as the delay endured by an SN to get back the result after offloading its task to an FN. It consists of the sum of the time needed for transferring the entire task to an FN or the cloud, the processing time, and the time to get back the results. Figs. 2(a) – 2(b) shows the overall latencies endured in cases when there are 100 and 500 FNs with SNs ranging from 200 – 1000 and compares HD-Fog, Traditional Fog, and Cloud Computing schemes. Although Fog Computing executes faster than Cloud Computing, on average, we observe a further 30% reduction in latency in the case of HD-Fog. We attribute this reduction of latency in the case of HD-Fog to the parallel execution of sub-tasks. On the other hand, traditional Fog Computing schemes sequentially perform all of the subtasks on the same FN. We further notice a proportionate increase in latency in all the 3 cases as the number of SNs increases. To get a better insight, we break the overall latency into Transmission and Processing Latencies, respectively. As the result is a single packet, we discard its transmission time and only focus on the transfer of the tasks.

Transmission Latency: We define this as the time needed for sending the data from an SN to an FN. For our simulation, we consider the Shannon Capacity value to determine the transmission data rate. As shown in Figs. 2(c) – 2(d), as the Cloud is situated far away from the SNs, it takes a higher time to reach the destination and get the processing done, which dominates the overall latency. In the case of HD-Fog, an SN can offload the data to any FN within its RAN range. The same may not hold in the case of Fog Computing schemes



(a) 100 Fog Nodes (b) 500 Fog Nodes

(c) 100 Fog Nodes (d) 500 Fog Nodes

(e) 100 Fog Nodes (f) 500 Fog Nodes

Fig. 2: Latencies endured by SNs in case of Traditional Fog Computing, Cloud Computing, and HD-Fog schemes

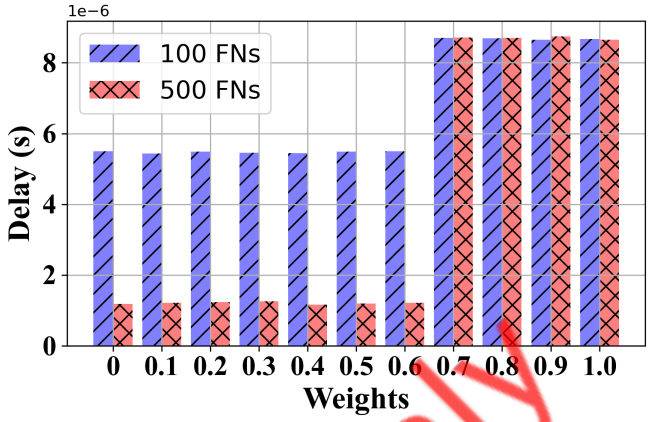
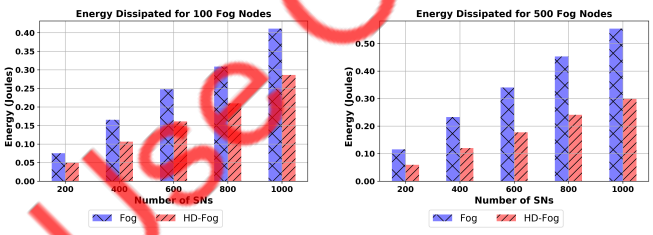


Fig. 3: Processing delay with changing weights



(a) 100 Fog Nodes (b) 500 Fog Nodes

Fig. 4: Comparison of Energy dissipated by SNs in case of Traditional Fog Computing and HD-Fog schemes

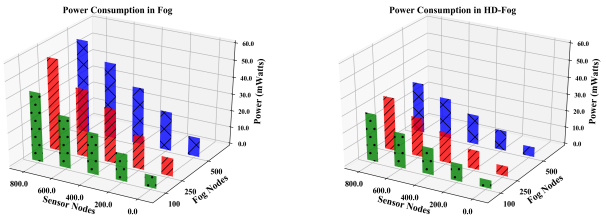
as the SNs generally lock the nearby FNs for their tasks, reducing the available number of processing cores. In HD-Fog, on transferring a task to an FN, it further redistributes the subtasks among other FNs, which increases the transmission time, compared to the Traditional Fog Computing scheme.

Processing Latency: We define this as the time needed for executing the tasks offloaded to the FNs and Cloud by the SNs. The cloud has much higher computation capability than the FNs due to which we see minuscule processing time while simulating the Cloud Computing scheme in Figs. 2(e) – 2(f). However, we observe that HD-Fog demonstrates a reduction of over 20% in processing latency than the Traditional Fog Computing scheme due to the introduction of parallelism. We present the impact of the weights on processing delays in Fig. 3. We strictly offload the computation to the fog layer when $\omega_{fn} \geq 0.6$. Interestingly, we observe a stable pattern (averaged over 30 iterations each) and similar delays to that in Figs. 2(e) and 2(f). We attribute the stability to the SNs choice of offloading the tasks rather than executing locally. We also observe that the processing time improves as the number of FNs increases. Intuitively, we attribute this behavior to the availability of a larger number of free processing cores.

We observe that there exists a tradeoff between the transmission and processing delays. HD-Fog involves transferring relevant data for sub-task offload, increasing the cumulative transmission time. However, the parallel execution in HD-Fog significantly reduces the processing time in comparison to the traditional Fog Computing scheme, resulting in lower overall delays. This compensates for the increase in the transmission time. However, HD-Fog necessitates reliable communication links. We plan to extend this work by making HD-Fog compliant under constrained conditions.

Energy Dissipated: This is the energy required by the SNs to offload their tasks to the FNs. Since we assume that the FNs have an unlimited supply of power, we focus only on the energy spent by the SNs. Figs. 4(a) –4(b) compares the energy dissipated in case of HD-Fog and Traditional Fog Computing scheme. We observe that the SNs in the case of HD-Fog need over 32% less energy to get the jobs done than the traditional Fog Computing schemes. This reduction is because, in HD-Fog, the SNs can offload to any nearby FN. This FN then redistributes the subtasks to the appropriate FNs. Traditional FNs need the SNs to select the destination FNs for the entire task. An increase in distance, as well as computation for the selection of FNs increases the energy consumption.

Power Consumption: We observe the power consumption by an FN while executing a task. The power consumption mostly depends on activities, such as *Data Forwarding*, *Computation*,



(a) Traditional Fog (b) HD-Fog

Fig. 5: Comparison of Power Consumption by FNs in case of Traditional Fog Computing and HD-Fog schemes

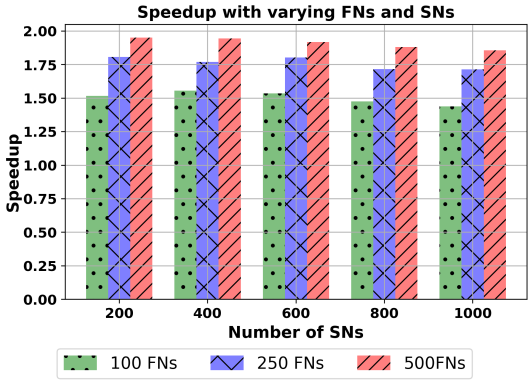


Fig. 6: Speedup with varying number of FNs and SNs

and *Data Storage*. Since HD-Fog only forwards and stores sub-tasks, we mainly focus on computation requirements. We observe in Fig. 5(b) that in the case of HD-Fog, the maximum power consumed is less than $30.0mW$, and in the case of Fog Computing in Fig. 5(a) is more than $50.0mW$. Roughly, the power consumption in HD-Fog is less than Traditional Fog Computing schemes by over 44%. We attribute this reduction to the distribution and parallel processing of the subtasks. As subtasks need fewer CPU cycles, it allows the FNs to operate only for small durations leading to less consumption in power.

Speedup: The speedup is the ratio of the time needed to perform the operations sequentially to that on operating the same in parallel. Fig. 6 illustrates the speedups in case of using 100, 250, and 500 FNs on 200 – 1000 SNs. As the overall latency in Figs. 2(a) – 2(b) of HD-Fog demonstrates significant savings in time, we observe the same reflection in the speedups. It is interesting to see that the speedups are proportionate as the number of SNs increases. However, the absolute value reduces, which dictates the need for determining the appropriate number of FNs with respect to the number of SNs, which we plan to address this in our extended work.

V. Conclusion

In this work, we proposed a horizontal task distribution architecture. We demonstrated HD-Fog as a distributed scheme for the parallel processing of tasks from IoT sensors. Sophisticated optimization techniques usually add computational overheads on the resource-constrained FNs and IoT SNs and take high time to converge. Thus, we formulated a greedy solution based on Queueing Theory and reliability. We also presented how HD-Fog outperforms Traditional Cloud and Fog Computing schemes with respect to latencies, and power consumptions. Although due to redistribution of sub-tasks, we observed an increase in the transmission latency in HD-Fog, the overall delay is less due to the reduction in processing latency as a result of the induced parallelism leading to promising speedup values. We conclude that HD-Fog is an undeniable solution for hard real-time operations. In the future, we plan to extend this work by developing methods for overcoming issues due to FN failures midway during executions.

References

[1] F. Hao, D. Park, J. Kang, and G. Min, "2L-MC3: A Two-Layer Multi-Community-Cloud/Cloudlet Social Collaborative Paradigm for Mobile Edge Computing," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4764–4773, Jun. 2019.

[2] Y. Ding, C. Liu, X. Zhou, Z. Liu, and Z. Tang, "A Code-Oriented Partitioning Computation Offloading Strategy for Multiple Users and Multiple Mobile Edge Computing Servers," *IEEE Transactions on Industrial Informatics*, pp. 1–1, Nov. 2019.

[3] Y. Sun, M. Peng, S. Mao, and S. Yan, "Hierarchical Radio Resource Allocation for Network Slicing in Fog Radio Access Networks," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 4, pp. 3866–3881, Apr. 2019.

[4] Y. Zhang, B. Feng, W. Quan, G. Li, H. Zhou, and H. Zhang, "Theoretical Analysis on Edge Computation Offloading Policies for IoT Devices," *IEEE Internet of Things Journal*, pp. 1–1, Oct. 2018.

[5] S. Sardellitti, G. Scutari, and S. Barbarossa, "Joint Optimization of Radio and Computational Resources for Multicell Mobile-Edge Computing," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 1, no. 2, pp. 89–103, Jun. 2015.

[6] O. Muñoz, A. Pascual-Iserte, and J. Vidal, "Optimization of Radio and Computational Resources for Energy Efficiency in Latency-Constrained Application Offloading," *IEEE Transactions on Vehicular Technology*, vol. 64, no. 10, pp. 4738–4755, Oct. 2015.

[7] A. Boukerche, S. Guan, and R. E. De Grande, "A task-centric mobile cloud-based system to enable energy-aware efficient offloading," *IEEE Transactions on Sustainable Computing*, vol. 3, no. 4, pp. 248–261, Oct 2018.

[8] H. Wei, H. Luo, Y. Sun, and M. S. Obaidat, "Cache-Aware Computation Offloading in IoT Systems," *IEEE Systems Journal*, vol. 14, no. 1, pp. 61–72, Mar. 2020.

[9] C. Shi, V. Lakafosis, M. H. Ammar, and E. W. Zegura, "Serendipity: Enabling Remote Computing Among Intermittently Connected Mobile Devices," in *Proceedings of the* 13^{th} *ACM International Symposium on Mobile Ad Hoc Networking and Computing*. ACM, Jun. 2012, pp. 145–154.

[10] H. Kaur and S. K. Sood, "Energy-Efficient IoT-Fog-Cloud Architectural Paradigm for Real-Time Wildfire Prediction and Forecasting," *IEEE Systems Journal*, vol. 14, no. 2, pp. 2003–2011, Jun. 2020.

[11] K. Wang, Y. Tan, Z. Shao, S. Ci, and Y. Yang, "Learning-Based Task Offloading for Delay-Sensitive Applications in Dynamic Fog Networks," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 11, pp. 11 399–11 403, Nov. 2019.

[12] Y. Jiang, Y. Chen, S. Yang, and C. Wu, "Energy-Efficient Task Offloading for Time-Sensitive Applications in Fog Computing," *IEEE Systems Journal*, vol. 13, no. 3, pp. 2930–2941, Sep. 2019.

[13] M. Min, L. Xiao, Y. Chen, P. Cheng, D. Wu, and W. Zhuang, "Learning-Based Computation Offloading for IoT Devices With Energy Harvesting," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 2, pp. 1930–1941, Feb. 2019.

[14] S. Chen, Y. Zheng, W. Lu, V. Varadarajan, and K. Wang, "Energy-Optimal Dynamic Computation Offloading for Industrial IoT in Fog Computing," *IEEE Transactions on Green Communications and Networking*, vol. 4, no. 2, pp. 566–576, Jun. 2020.

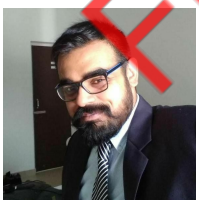
[15] Cisco, "Fog Computing and the Internet of Things: Extend the Cloud to Where the Things Are," *Cisco*, no. C11-734435-00, 2015.

[16] H. Shah-Mansouri and V. W. S. Wong, "Hierarchical Fog-Cloud Computing for IoT Systems: A Computation Offloading Game," *IEEE Internet of Things Journal*, vol. 5, no. 4, pp. 3246–3257, Aug. 2018.

[17] S. Sarkar and S. Misra, "Theoretical Modelling of Fog Computing: A Green Computing Paradigm To Support IoT Applications," *IET Networks*, vol. 5, no. 2, pp. 23–29, Mar. 2016.

Pallav K. Deb He is a Ph.D. Research Scholar in the Department of Computer Science and Engineering, Indian Institute of Technology Kharagpur, India. He received his M.Tech degree in Information Technology from Tezpur University, India in 2017. Prior to that, he has completed the B. Tech degree in Computer Science from the Gauhati University, India in 2014. The current research interests of Mr. Deb include UAV swarms, THz Communications, Internet of Things, Cloud Computing, Fog Computing, and Wireless Body Area Networks. Further details are available in https://pallvdeb.github.io/

Sudip Misra (M'09–SM'11) He is a Professor with the Department of Computer Science and Engineering, Indian Institute of Technology, Kharagpur. He received his Ph.D. degree in Computer Science from Carleton University, in Ottawa, Canada, and the masters and bachelor's degrees, respectively, from the University of New Brunswick, Fredericton, Canada, and the Indian Institute of Technology, Kharagpur, India. Dr. Misra is the Associate Editor of the IEEE Transactions Mobile Computing and IEEE Systems Journal, IEEE Transactions on Sustainable Computing, IEEE Network, and Editor of the IEEE Transactions on Vehicular Computing. His current research interests include algorithm design for emerging communication networks and Internet of Things. Further details about him are available at http://cse.iitkgp.ac.in/ smisra/.



Anandarup Mukherjee He is currently a Senior Research Fellow and Ph.D. Scholar in Engineering at the Department of Computer Science and Engineering at Indian Institute of Technology, Kharagpur. He finished his M.Tech and B.Tech from West Bengal University of Technology in the years 2012 and 2010, respectively. His research interests include, but are not limited to, networked robots, unmanned aerial vehicle swarms, Internet of Things, Industry 4.0, 6G and THz Networks, and enabling deep learning for these platforms for controls and communications. His detailed profile can be accessed at http://www.anandarup.in