Chapter 5

A game theory-based passenger assistance system using Fog computing

In the previous chapters, we discussed two challenges encountered in locomotion mode detection, *i.e.,* unseen classes and noisy labels in the dataset. We proposed the deep learning-based approach, which incorporates zero-shot learning for handling unseen class labels in the dataset. Next, we dealt with the challenge of frequently occurring noisy labels in the dataset result in performance diminution. By resolving both the challenges in locomotion mode detection, we have built the classifier that is robust against both unseen classes and noisy labels in the dataset. This chapter presents the mechanism for task processing on smartphones and proposes a Fog computing based transportation system. The system uses multiple Fog devices to assist the passengers. A passenger generates a task and forwards it to the Fog devices for further processing using the Edge device. Selected Fog devices parallelly process the fraction of the task so that the complete task processes within the time constraint.

5.1 Introduction

The public transport in urban society needs to be intelligent and interactive for providing better service to passengers. For example, the transportation system for the tourists should be capable of guiding the passengers about the routes and nearby sights. With the recent advancements in Internet technology and smart devices, the passengers no longer need to rely on the conventional traveling aids (*e.g.*, explanatory brochure or road maps) as they can avail required information using online sources. Such information can be access locally or from the Cloud. Cloud computing suffers from substantial communication delay and requires continuous long-range communication network (such as 4G or 5G). Such networks consume huge power of smart devices. It is convenient if the vehicle locally assists the passengers by providing a faster response to their queries using the preloaded information. However, due to infrequent updates of information and limited local resources, the vehicle can not provide sufficient information.

Fog computing mitigates the shortcomings of Cloud computing and local processing [14]. The layered architecture of Fog computing comprises Edge Devices (EDs), Fog Devices (FDs), and Cloud. The FDs can interact with each other and provide parallel processing of the data. The interaction among the FDs overcomes the limitation of unequal storage and computation power of the FDs. Transportation system using Fog computing can assist passengers in the processing of task and updating the information about their journey. Figure 5.1 illustrates an example scenario for passenger assistance, where a passenger on a sightseeing vehicle (such as hop-on-hop-off) captures some images of a monument using the smartphone. Next, the smartphone wirelessly transfers the images with the query to the vehicle which acts as ED. The vehicle processes the query if they can solve or further forwards to the Road Side Unit (RSU) which acts as FD. The RSU processes the query and transfers the query response to the passenger.

In this chapter, we present a **T**ransportation **S**ystem using **F**og computing (TSF) for passenger assistance. TSF uses multiple FDs (RSUs) that are interconnected with

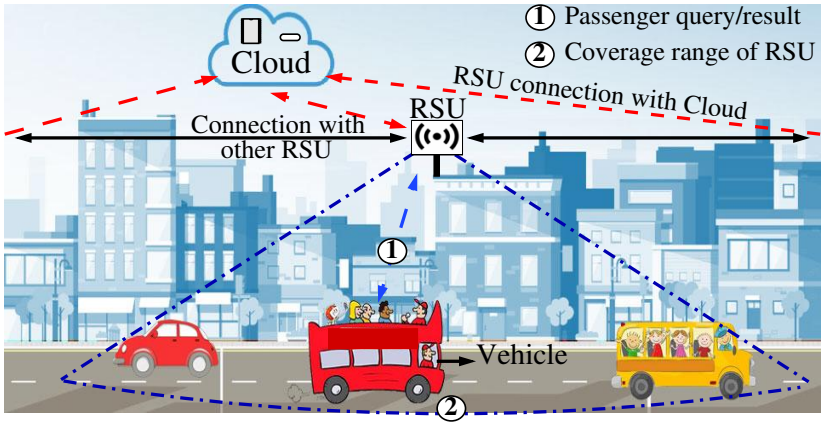


Figure 5.1: An example of transportation system using Fog computing.

each other and the Cloud, as shown in Figure 5.2. EDs (vehicles) collect data from the users and forward it to the connected FD, called primary FD (p-FD). The p-FD is connected with other FDs and the Cloud. Such devices have different processing power; therefore, require unequal processing cost. Based on the availability of the resources, given constraints, and the cost, the p-FD decides to offload the fractions of the task to the secondary FDs (s-FDs). The objective of this work is to address the following problem: *what are the fractions of a task offloaded to the p-FD, s-FDs, and Cloud for processing the task within the given time constraint and requires minimal cost?* To solve the problem, we use the competitive game approach and Knapsack algorithm for partitioning the task among the p-FD, s-FDs, and Cloud. Apart from this, we made the following contributions:

- We first estimate transmission and processing delays of data associated with a task. We next present a reputation model which provides the confidence for successful completion of the given task. The existing work for offloading the task using Fog computing [14–19, 44–46] have not considered the reputation of FDs.
- Next, we present Fog computing based transportation system, which comprises two phases. First phase selects suitable FDs for processing a task using the signal to interference noise ratio. The second phase estimates the fraction of the task

process at p-FD and offloads to s-FDs, such that the task must be processed within given time constraint. The available resources to process a given task using FD depends upon the load at the time of processing. The proposed work considers the actual processing capability or available resources at each FD instead of the average or pre-fixed processing capability, which is assumed by the existing work [14, 42, 43, 47]. We present a Knapsack based task offloading algorithm to fully utilize the resources of the s-FDs.

- Further, we present a competitive game model and near Nash Equilibrium (ϵ-NE) solution to estimate the optimal fraction of the task to be solved at p-FD and offloaded s-FDs. Such fractions of task are the input of Knapsack based task offloading algorithm. The ϵ-NE solution agrees by all FDs; therefore, no FD deviates from task processing. The existing work not considered the cooperation among FDs; thus, they may deviate from task processing, whenever they have higher utility [44–46].
- Finally, we validate the TSF system and experimentally analyze impact of data size, task deadline, devices, and game parameters on accuracy and system cost.

• **Motivation of this work:** This work is motivated by the following shortcomings of the existing work. First, the existing literature on task offloading using Fog computing [14–17, 42–46] presented hierarchical networks for consisting of Edge, Fog, and Cloud layers for distributed tasks execution. However, the existing work do not consider the interaction among the FDs, *i.e.*, offload from one FD to others. Next, a few prior work [14, 42, 43, 47] considered heterogeneity of FDs limited to the transmission of the complete task; hence, utilizing available resources inefficiently. Next, the existing work on task offloading in Fog computing [44–46] incorporated offloading of complete task from one device to another. However, to ensure load balancing among FDs, task partitioning is requisite to assign the most appropriate device to execute the task. Finally, the existing work have not considered the reputation of FDs during task offloading.

The rest of the chapter is organized as follows: In next section, we present the preliminaries and network topology used in this work. Further, Section 5.3 presents the proposed TSF system, followed by competitive game models in Section 5.4. We present prototype setup and performance evaluation in Section 5.5. The chapter ends with the conclusion in Section 5.6.

5.2 Preliminary and network topology

This section discusses the network topology and terms involved in this chapter. We also describe transmission and communication delays, followed by the reputation model.

5.2.1 Network Topology

We use the hierarchical network topology which consists of layered based devices, as illustrated in Figure 5.2. The leaf devices (*e.g.*, smartphone of passengers) collect $\mathcal{D}$ bits data and wirelessly forward to Edge devices (*e.g.*, vehicle). The Edge devices store, route, and forward data $\mathcal{D}$ to Fog devices (*e.g.*, RSUs) denoted as FDs. We assume that initially all FDs have similar processing capacity f. The FDs have higher storage, computation, and communication capabilities compared to Edge devices. The FDs are connected with each other and also with the Cloud. Let $\{F_1, F_2, F_3, \ldots, F_{\mathcal{N}}\}$ and $F_{\mathcal{N}+1}$ denote $\mathcal{N}$ FDs and Cloud, respectively. The task T of size $\mathcal{D}$ is executed at Fog layer with task distribution among FDs. In adverse case when FDs are incapable of processing the task T completely, the remaining portion of the task is offloaded to Cloud to meet out time constraint.

5.2.2 Transmission and computation delays

• **Data transmission delay:** Let B_{ij}, h_{ij}, and z_{ij} represent the bandwidth, channel coefficient, and path loss frequency between devices i and j, respectively. The data

transmission rate $\mathcal{R}_{i,j}$ from F_i to F_j can be calculated using Shannon channel capacity $\mathcal{R}_{i,j} = B_{ij} \log_2 \left(1 + \frac{z_{ij} h_{ij}^2}{\sigma^2}\right)$. Time to transmit $\mathcal{D}$ bits data from F_i to F_j is given as:

$$T_{i,j} = \frac{\mathcal{D}}{\mathcal{R}_{i,j}} = \frac{\mathcal{D}}{B_{ij} \log_2 \left(1 + \frac{z_{ij} h_{ij}^2}{\sigma^2}\right)}. \tag{5.1}$$

p-FD initially does not have the available resource information of the selected s-FDs. Therefore, the initial task offloading decision on p-FD is of equally splitting the task having the same datasize. Next, the p-FD informs s-FDs of their sub-tasks to be allocated. Later, the s-FDs estimate the time based on the available resources and notify the p-FD about the portion of the task they can process. Finally, the p-FD updates the offloading decision based on the received responses from s-FDs. During the update of decisions, the task and corresponding datasize vary from one s-FD to other.

• **Data computation delay:** Let f_j denote the computation capability of a FD $F_j \in \mathcal{N}$ in accordance with number of CPU cycles in one time unit. The computation time T_j^c for execution of task T of $\mathcal{D}$ bits at F_j is $T_j^c = \frac{\mathcal{D} \times q}{f_j}$, where q denotes CPU cycles required for processing one bit.

• **Result transmission delay:** Transmitting the result of size d bits from F_j to the source F_i incurs result transmission delay T_{ji}^r as:

$$T_{j,i}^r = \frac{d}{\mathcal{R}_{i,j}} = \frac{d}{B_{ij} \log_2 \left(1 + \frac{z_{ij} h_{ij}^2}{\sigma^2}\right)}. \tag{5.2}$$

The total delay ($\mathbb{T}$) is the sum of data transmission, computation, and result transmission delays, is given as:

$$\mathbb{T} = T_{i,j} + T_j^c + T_{j,i}^r. \tag{5.3}$$

5.2.3 Reputation model

The successful inference of the data portion is a binary event that can be represented using beta distributions. Let r_j^i and w_j^i represent the positive and negative reputation feedback, respectively, of FD F_j by neighboring FD F_i. The reputation function $\mathcal{F}(x|r_j^i, w_j^i)$ of F_j in terms of F_i is given as:

$$\mathcal{F}(x|r_j^i, w_j^i) = \frac{\mathbf{\Gamma}(r_j^i + w_j^i + 2)}{\mathbf{\Gamma}(r_j^i)\mathbf{\Gamma}(w_j^i)} p^{r_j^i}(1-p)^{w_j^i}, \tag{5.4}$$

where, $\mathbf{\Gamma}(\cdot)$ is a gamma function and value of variable x, r_j^i and w_j^i lie between $[0, 1]$ [112]. The expectation value of the reputation function $E[\mathcal{F}(x|r_j^i, s_j^i)] = \frac{r_j^i+1}{r_j^i+w_j^i+2}$.

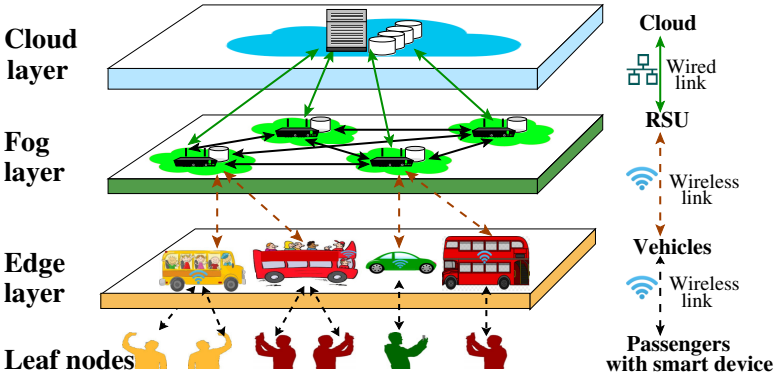


Figure 5.2: Network topology for the TSF system, passengers at bottom layer are sitting in vehicles.

Definition 5.1 ***(Reputation rating)*** *The expected value of reputation function provides a rating to the FD between* $[0, 1]$*. It is convenient to convert expected value to a reputation feedback between* $[-1, 1]$*. Therefore, reputation rating* R_j^i *can be defined as:*

$$R_j^i = \big(E[\mathcal{F}(x|r_j^i, w_j^i)] - 0.5\big) \times 2 = \frac{r_j^i - w_j^i}{r_j^i + w_j^i + 2}. \tag{5.5}$$

Definition 5.2 ***(Combined reputation rating)*** *Let* $E[\mathcal{F}(x|r_j^i, w_j^i]$ *and* $E[\mathcal{F}(x|r_j^k, w_j^k)]$ *be the reputation function of FD* j *by p-FD* i *and s-FDs* $\{k = \{n\}/j\}$*, respectively. The*

combined reputation rating of FD j for p-FD i is as follows:

$$r_j^{i,k} = br_j^i + (1-b)\sum_{k=\{n\}/j} r_j^k,$$
$$w_j^{i,k} = bw_j^i + (1-b)\sum_{k=\{n\}/j} w_j^k, \tag{5.6}$$

where, $0 \leq b \leq 1$ *is a bias factor that indicates the importance of feedback. The reputation rating of s-FD j for p-FD i can be defined as:*

$$\mathbf{r_{ij}} = \frac{r_j^{i,k} - w_j^{i,k}}{r_j^{i,k} + w_j^{i,k} + 2}. \tag{5.7}$$

Definition 5.3 ***(Nash Equilibrium)*** *If each player of the game selects a strategy, none of the players can enhance their payoff by changing its strategy, while other players keep their strategies unchanged; then the game is said to have reached the Nash Equilibrium (NE). It is achieved, when for each player i there exist strategy* $s_i^\star$*, where* $s_i^\star \in S_i$ *(*S_i *is strategies set of* i^{th} *player), such that utility* U_i *of each player i holds:*

$$U_i(s_i^\star, s_{-i}) \geq U_i(s_i, s_{-i}),\ \forall i \in \mathcal{N},$$
$$U_i(s_i^\star, s_{-i}^\star) \geq U_i(s_i, s_{-i}). \tag{5.8}$$

5.3 TSF system

This section proposes Transportation System using Fog computing (TSF) which involves two phases, Phase I: selection of secondary Fog devices and Phase II: task offloading and distributed execution.

5.3.1 Phase 1: Selection of secondary Fog devices

Initially, a FD receives data from the ED which work as p-FD, denoted as F_i, in the system. The p-FD communicates with the nearby FDs ($F_j, \forall j \in \mathcal{N}$) to offload the

task T. The estimation of Signal to Interference plus Noise Ratio (SINR) for each pair (F_i, F_j) increases the reliability of data transmission between F_i and F_j. Let $\mathcal{G}_{i,j}$ and $d_{i,j}$ denote fading power gain and distance between F_i and F_j, respectively. The SINR $\mathcal{S}_{i,j}$ can be obtained as:

$$\mathcal{S}_{i,j} = \frac{\mathcal{G}_{j,i} q_j z_j d_{j,i}^{-\gamma}}{\sigma^2 + \sum_{j \in \mathcal{N}} \mathcal{G}_{r,i} p_r z_r d_{r,i}^{-\gamma}}, \tag{5.9}$$

where, q_j is frequency of path loss and γ denote path-loss. Let $\mathcal{T}$ be the minimum value of SINR required for the successful transmission of data from F_i to F_j. The p-FD F_i considers the s-FDs (F_j) only if the SINR of pair (F_i, F_j) higher than threshold $\mathcal{T}$. Such selected s-FDs are only participating for task offloading. We therefore finally obtain a set of n s-FDs that can provide a significant level of reliability in the successful offloading of the task.

5.3.2 Phase II: Task offloading

This phase involves initialization and offloading decision update steps. In the initialization step, randomly fractions of the task is offloaded on the FDs for the execution. These randomly assigned values are updated using the competitive game model, which is discussed in Section 5.4. In the offloading decision update step, the portion of the task assigned to each s-FD is updated based on the available resources on the s-FD.

5.3.2.1 Initialization

Let p-FD (F_i) receives a task T of size $\mathcal{D}$ that is to be executed within a time constraint τ_{max}. Let time required to execute the task at p-FD be $\tau > \tau_{max}$, with the processing capacity f bits/sec, *i.e.*, $\tau = \mathcal{D}/f$. The p-FD partitions τ in τ_i and $\tau - \tau_i$ to execute T within τ_{max}. Portion τ_i is executed on F_i and $\tau - \tau_i$ is equally divided among n selected F_j such that a s-FD executes $\tau_j = \frac{\tau - \tau_i}{n}$ portion of T, where $F_j \in \mathcal{N}$ and $1 \leq j \leq n$. Initially, offloading decision is made by considering equal capacity of FDs. However,

the processing capability of each FD may not be equal as it depends on the available resources. F_i, F_j assign initial value of the portion of task to be execute *i.e.,* τ_i and τ_j.

5.3.2.2 Estimation of the optimal value of τ_i and τ_j

In this section, competitive game model is used for task offloading among FDs. Initial value of τ_i and τ_j are taken as input and the optimal values of τ_i and τ_j are estimated using the competitive game model, discussed in Section 5.4.

5.3.2.3 Offloading decision update

In this step, the offloading decision is updated based on the current resources available on s-FDs. The current computation capability of F_j can be higher or lower than average processing capability f which depending on available resources. The task portion estimated by initial offloading decision for each F_j is $\mathcal{D}' = f \times \tau_j$, whereas, the actual task portion that F_j can complete is $\mathcal{D}'_j = f_j \times \tau_j$. Using actual processing capability f_j of s-FD, we can define task execution as an indicator function $\mathbb{1}_{\{f>f_j\}}$. The indicator function gives value 1 when $f > f_j$ and 0 otherwise. If $\mathbb{1}_{\{f>f_j\}} = 0$ then F_j will complete the estimated task portion $\mathcal{D}'$ in time $\tau'_j = \mathcal{D}'/f_j$. Thus, F_j is idle for time $\mathcal{I}_j = \tau_j - \tau'_j$. The total task bits W remaining for execution is sum of task bits left from different F_j:

$$W = \sum_{j}^{n} \mathbb{1}_{\{f>f_j\}} \lfloor \mathcal{D}' - \mathcal{D}_j \rfloor. \tag{5.10}$$

W can be offloaded to the Cloud or other s-FDs which are idle for time $\mathcal{I}_j$. As the processing cost at Cloud is higher than F_j; therefore, F_j are preferred for further task offloading. W may or may not be executed completely on a single F_j with limited $\mathcal{I}_j$. This enforces to divide W in such a way that maximum bits of W can be transferred to F_j with highest $\mathcal{I}_j$. The remaining task distribution on idle F_j closely resembles with **Knapsack problem**. Let η_j be the fraction of W assigned on F_j with Knapsack on

time constraints th. The Knapsack problem for W bits task allocated is defined as:

$$\max \sum_{j=1}^{z} \eta_j W \mathcal{I}_j,$$
$$s.t., \sum_{j=1}^{z} \eta_j W \left(\frac{1}{f_j}\right) \leq th, \tag{5.11}$$

where, $\eta_j \in [0, 1)$ and z is the number of F_j remains idle during task execution. The optimization problem of Knapsack is solved using Algorithm 5.1. The stopping condition is reached when either the task is completely assigned or no more available F_j.

5.4 Competitive game model for task offloading

This section presents a competitive game model for task offloading among FDs. First, we derive the utility functions of p-FD and s-FDs. Next, we present the p-FD utility maximization problem for estimating the optimal task partitions. The optimal price taken by the s-FDs is estimated with the utility maximization problem of s-FDs. Finally, the section proposes a near Nash Equilibrium (ϵ-NE) solution among s-FDs for the following competitive game:

- *Players:* s-FDs (total n s-FDs) and p-FD.
- *Actions:* The action of the p-FD is to select an optimal task fraction τ_i that maximizes its utility. On the other hand, each s-FD determines the price p_j for their service to the p-FD to maximize their utilities.
- *Utility function:* The utility functions of p-FD F_i and s-FD F_j are $U_i(\cdot)$ (Equation 5.16) and $U_j(\cdot)$ (Equation 5.17), respectively.

5.4.1 Utility of primary Fog device

The utility function $U_i(\cdot)$ of p-FD F_i have following terms:

- *Price gain from Edge device:* The p-FD F_i receives a price p_i per unit time from

Algorithm 5.1: Knapsack based task offloading.

Input: W, th z, f_j, $\mathcal{I}_j$ $\forall j \in z$.
Output: Suitable f_j to execute W bits data.

$t \leftarrow 0$, $\eta_0 \leftarrow 1$. /* Variables initialization */
Calculate W using Equation 5.10.
$f(\eta_j) = th - \left(t + \eta_j W \frac{1}{f_j}\right)$.
Arrange F_j in descending order as per $\mathcal{I}_j$. /* Select η_j from F_j as Knapsack th */
for $j \leftarrow 1$ *to* z **do**
    while $t < th$ **do**
        if $f(\eta_j) \geq 0$ **then**
            $t \leftarrow t + \eta_j W \left(\frac{1}{f_j}\right)$.
            return
        else
            /* select $\eta_j \in [0, 1)$, using Bisection method */
            $\eta_j \leftarrow$ Bisection_method().
            if $\eta_j = 0$ **then**
                /* Remaining W bits executed on Cloud */
                return
            $t \leftarrow t + \eta_j W \left(\frac{1}{f_j}\right)$.
        $W \leftarrow W - \eta_j W$, $\eta_j \leftarrow \eta_j$.

Function Bisection_method()
$t_l \leftarrow .02$, $a \leftarrow 0$, $b \leftarrow 0.98$, $m_{iter} \leftarrow 20$,$i \leftarrow 1$.
while $i \leq m_{iter}$ **do**
    $n_j \leftarrow \frac{(a+b)}{2}$.
    if $f(n_j) = 0$ *or* $\frac{b-a}{2} < t_l$ **then**
        return n_j
    if $|f(n_j)| == |f(a)|$ **then**
        $a \leftarrow c$
    $b \leftarrow c$, $i \leftarrow i + 1$
return 0

the Edge device, which generates the task. Since the task completes its execution in time τ; therefore, the price gain $L_e = p_i \times \tau$.

- *Cost of task execution:* The p-FD decides to execute τ_i portion; thus, the cost incurred for utilizing its resources for task execution is defined as:

$$L_c = c_h \log\left(1 - \frac{\tau_i}{\tau}\right), \tag{5.12}$$

where, c_h is a constant cost coefficient as given in [113].

- *Service cost to s-FDs:* $(\tau - \tau_i)$ portion of task is offloaded to n s-FDs, which provide service at the price p_j per unit time. The service cost paid by p-FD F_i also depends on the reputation $\mathbf{r_{ij}}$ of the s-FD F_j, given by Equation 5.7. The service cost L_s paid to the s-FDs is computed as:

$$L_s = \sum_{j=1}^{n} \mathbf{r_{ij}} \times p_j \times \tau_j. \tag{5.13}$$

- *Discomfort cost:* The s-FD can either accept or reject the task, based on its available resources. When a s-FD F_j rejects the assigned task by F_i, it creates a discomfort D_{ij} to the p-FD F_i. Let l be the number of s-FDs that were unsuccessful in completing their previous offloaded task. The discomfort cost L_d with given pricing constant $\mathbb{C}$ can be defined as:

$$L_d = \sum_{j=1}^{l} D_{ij} = \sum_{j=1}^{l} (1 - \mathbf{r_{ij}}) \mathbb{C} \left(\log \left(1 - \frac{\tau_j}{\tau} \right) \right). \tag{5.14}$$

Utility function $U_i(\cdot)$ of p-FD F_i is the sum of price it receives for task computation, service cost paid for offloading, cost of its own resources, and discomfort cost.

$$\begin{aligned} U_i(\tau_i) &= L_e - L_c - L_s - L_d \\ &= p_i \tau - c_h \log \left(1 - \frac{\tau_i}{\tau} \right) - \sum_{j=1}^{n} \mathbf{r_{ij}} p_j \tau_j - \sum_{j=1}^{l} (1 - \mathbf{r_{ij}}) \mathbb{C} \left(\log \left(1 - \frac{\tau_j}{\tau} \right) \right). \end{aligned} \tag{5.15}$$

Now using Taylor series expansion [114] in Equation 5.15 and removing higher order terms we get utility of p-FD F_i as:

$$U_i(\tau_i) = p_i\tau - c_h\left(\frac{\tau_i}{\tau} + \frac{(\tau_i)^2}{\tau^2}\right) - \sum_{j=1}^{n} \mathbf{r_{ij}} p_j \left(\frac{\tau - \tau_i}{n}\right) - \sum_{j=1}^{l}(1 - \mathbf{r_{ij}})\mathbb{C}\left(\frac{\tau - \tau_i}{n\tau} + \frac{(\tau - \tau_i)^2}{n^2\tau^2}\right). \quad (5.16)$$

5.4.2 Utility of secondary Fog device

Utility function $U_j(\cdot)$ of s-FD F_j holds these terms:

- *Price gain from p-FD:* The s-FD announces a price p_j per unit time that p-FD have to pay for executing task fraction τ_j on s-FD. The price gain is given as: $K_d = \mathbf{r_{ij}} \times p_j \times \tau_j$.
- *Cost of task execution:* The s-FD executes τ_j portion of the task and the cost K_c incurred upon using its resources is: $K_c = c_h \log\left(1 - \frac{\tau_j}{\tau}\right)$.
- *Price gain of multiple offloading:* This price is the part of the profit the p-FD gains by executing the task on n s-FDs in parallel mode. The price p_r is a constant fraction of the gain by the p-FD. The price gain of multiple offloading is: $K_m = n \times p_r \times \tau_j$.

Utility $U_j(\cdot)$ of s-FD F_j is calculated by considering the price received from the p-FD and the cost of executing the offloaded task. $U_j(\cdot)$ is calculated as follows:

$$\begin{aligned} U_j(p_j) &= K_d + K_c + K_m = \mathbf{r_{ij}} p_j\tau_j - c_h \log\left(1 - \frac{\tau_j}{\tau}\right) + np_r\tau_j, \\ &= \left(\mathbf{r_{ij}} p_j - \frac{c_h}{\tau} + np_r\right)\tau_j - \frac{c_h {\tau_j}^2}{\tau^2}. \end{aligned} \quad (5.17)$$

• **Problem 1:** *Utility maximization of primary Fog device:*

$$\begin{aligned} \max_{\tau_i} \quad & U_i(\tau_i) \\ s.t. \quad & \max\left\{\tau_i, \frac{\tau - \tau_i}{n}\right\} < \tau_{max}, \end{aligned} \quad (5.18)$$

where, τ_{max} is maximum allowable execution time of task T.

Theorem 5.1 *The optimal task allocation is:*

$$\tau_i^\star = \frac{\sum_{j=1}^{l} \frac{(1-\mathbf{r_{ij}})\mathbb{C}}{n\tau}(1+2/n) - \frac{c_h}{\tau} + \sum_{j=1}^{n} \frac{\mathbf{r_{ij}} p_j}{n}}{\frac{2}{\tau^2}\left(c_h + \sum_{j=1}^{l}(1-\mathbf{r_{ij}})\frac{\mathbb{C}}{n^2}\right)},$$
$$\tau_j^\star = \frac{\tau - \tau_i^\star}{n}. \tag{5.19}$$

Proof: Consider the utility function of primary Fog device from Equation 5.16. Let λ_1 and λ_2 denote Lagrangian multipliers. The Lagrangian form of problem given in Equation 5.18 can be represented as:

$$\begin{aligned}\mathcal{L}(\tau_i) =& p_i\tau - c_h\left(\frac{\tau_i}{\tau} + \frac{(\tau_i)^2}{\tau^2}\right) - \sum_{j=1}^{n}\mathbf{r_{ij}}p_j\left(\frac{\tau-\tau_i}{n}\right) - \sum_{j=1}^{l}(1-\mathbf{r_{ij}})\mathbb{C}\left(\frac{\tau-\tau_i}{n\tau} + \frac{(\tau-\tau_i)^2}{n^2\tau^2}\right)\\ &+ \lambda_1(\tau_i - \tau_{max}) + \lambda_2\left(\frac{\tau-\tau_i}{n} - \tau_{max}\right),\\ s.t.\quad & \lambda_1(\tau_i-\tau_{max}),\ \lambda_2\left(\frac{\tau-\tau_i}{n} - \tau_{max}\right) = 0,\end{aligned} \tag{5.20}$$

The first order derivative of $\mathcal{L}(\cdot)$ *w.r.t.* τ_i is obtained as:

$$\frac{d\mathcal{L}(\tau_i)}{d\tau_i} = -c_h\left(\frac{1}{\tau} + \frac{2\tau_i}{\tau^2}\right) + \sum_{j=1}^{n}\mathbf{r_{ij}}p_j\left(\frac{1}{n}\right) + \lambda_1 - \frac{\lambda_2}{n} + \sum_{j=1}^{l}(1-\mathbf{r_{ij}})\mathbb{C}\left(\frac{1}{n\tau} + \frac{2(\tau-\tau_i)}{n^2\tau^2}\right).$$

The second order derivative of $\mathcal{L}(\cdot)$ *w.r.t.* τ_i, *i.e.*, $\frac{d^2\mathcal{L}(\tau_i,\lambda)}{d^2\tau_i} = -\frac{2c_h}{\tau^2} - \sum_{j=1}^{l}(1-\mathbf{r_{ij}})\mathbb{C}\left(\frac{2}{n^2\tau^2}\right)$ is negative, which indicates the utility function given in Equation 5.16 is concave and continuous. Therefore, we conclude that the problem given in Equation 5.18 have at least one unique solution. Using Karush Kuhn Tucker condition and

solving for different λ_1 and λ_2, the maximal utility is obtained at $\lambda_1, \lambda_2 = 0$ and $\tau_i^\star$ as:

$$\tau_i^\star = \frac{\sum_{j=1}^{l} \frac{(1-\mathbf{r_{ij}})\mathbb{C}}{n\tau}(1+2/n) - \frac{c_h}{\tau} + \sum_{j=1}^{n} \frac{\mathbf{r_{ij}} p_j}{n}}{\frac{2}{\tau^2}\left(c_h + \sum_{j=1}^{l}(1-\mathbf{r_{ij}})\frac{\mathbb{C}}{n^2}\right)}. \tag{5.21}$$

This implies $\tau_j^\star = \frac{\tau - \tau_i^\star}{n}$ and hence proved. □

Proposition 5.1 *τ_i lies in the range $0 \leq \tau_i \leq \tau$, if following condition holds:*

$$\frac{\sum_{j=1, j\neq i}^{l}(1-\mathbf{r_{ij}})\mathbb{C}}{\tau} + \sum_{j=1, i\neq j}^{n} \mathbf{r_{ij}} p_j \leq \frac{3nc_h}{\tau}. \tag{5.22}$$

Proof: Using τ_i from Equation 5.21, we get $\tau_i = \frac{\eta}{\beta}$, where, $\eta = \sum_{j=1}^{l} \frac{(1-\mathbf{r_{ij}})\mathbb{C}}{n\tau}(1+2/n) - \frac{c_h}{\tau} + \sum_{j=1}^{n} \frac{\mathbf{r_{ij}} p_j}{n}$ and $\beta = \frac{2}{\tau^2}\left(c_h + \sum_{j=1}^{l}(1-\mathbf{r_{ij}})\frac{\mathbb{C}}{n^2}\right)$. Since τ_i lies in the range $0 \leq \tau_i \leq \tau$, it implies $0 \leq \frac{\eta}{\beta} \leq \tau \implies \frac{\eta}{\beta} \leq \tau$; thus, $\eta \leq \beta\tau$. Substituting η and β in the inequality $\eta \leq \beta\tau$, we can obtain:

$$\frac{\sum_{j=1\neq i}^{l}(1-\mathbf{r_{ij}})k}{n\tau} - \frac{c_h}{\tau} + \frac{\sum_{j=1\neq i}^{n} \mathbf{r_{ij}} p_j}{n} + 2\frac{\sum_{j=1\neq i}^{l}(1-\mathbf{r_{ij}})k}{n^2\tau} \leq \frac{2}{\tau^2}\Big(c_h + \sum_{j=1\neq i}^{l}(1-\mathbf{r_{ij}})\frac{k}{n^2}\Big)\tau.$$

On solving above, we can obtain $\frac{\sum_{j=1, j\neq i}^{l}(1-\mathbf{r_{ij}})k}{n\tau} + \frac{\sum_{j=1, j\neq i}^{n} \mathbf{r_{ij}} p_j}{n} \leq \frac{3c_h}{\tau^2}$ and hence proved. □

Proposition 5.2 *The portion of task allocated to the primary FD is unique and optimal solution for **Problem 1** given in Equation 5.18.*

Proof: (**Uniqueness**) Task portion allocated to the p-FD, as given in Equation 5.21, is unique, when it holds *positivity*, *monotonicity*, and *scalability*. Using Proposition 5.1, we confirm that $\tau_i \geq 0$, which is sufficient to prove positivity of task portion τ_i. Let τ_i and $\tilde{\tau_i}$ are the portions of task allocated to the p-FD. Equation 5.19 indicates that if $\tau_i \geq \tilde{\tau_i}$, then $U_i(\tau_i) \leq U_i(\tilde{\tau_i})$, therefore, we conclude τ_i holds monotonicity. Using Equation 5.19, we get $\gamma U_i(\tau_i) - U_i(\gamma\tau_i) \geq 0$; hence, we prove scalability.

(**Optimality**) The utility function $U_i(\tau_i)$ of the problem given in Equation 5.18 is a quadratic function for given τ_i and the constraint is an affine function. It indicates that the problem in Equation 5.18 is a convex optimization problem. From Equation 5.21, $\tau_i^\star = \min(\tau_i, \tau^{max})$. Let us consider $\tau_i \geq \tau^{max}$ and hence $\tau_i^\star = \tau^{max}$. The objective function $U_j(p_j)$, defined in Equation 5.17 is an increasing function *w.r.t.* p_j and direcly depends upon the value of τ_j (*i.e.* $\tau - \tau_i$). The function would be maximum for $\tau_i < \tau^{max}$, which contradicts our assumption. Therefore, $\tau_i^\star = \tau_i$ and ends the proof. □

For simplifying the expressions further, let us consider $\eta = D_1 + \frac{1}{n}(\mathbf{r_{ij}} p_j)$, where $D_1 = \sum_{j=1}^{l} \frac{(1-\mathbf{r_{ij}})\mathbb{C}}{n\tau}(1 + 2/n) - \frac{c_h}{\tau} + \frac{\sum_{k\neq j} \mathbf{r_{ik}} p_k}{n}$. The value of τ_j can be computed using the value of η as $\tau_j = \frac{\tau - \tau_i}{n} = \frac{1}{n}\left(\tau - \frac{D_1}{\beta}\right) - \frac{\mathbf{r_{ij}} p_j}{n^2\beta} = D_2 - \frac{\mathbf{r_{ij}} p_j}{n^2\beta}$, where, $D_2 = \frac{1}{n}\left(\tau - \frac{D_1}{\beta}\right)$. Substituting value of τ_j in Equation 5.17, we obtain the utility of s-FD F_j as:

$$U_j(p_j) = (\mathbf{r_{ij}} p_j - \frac{c_h}{\tau} + np_r)(D_2 - \frac{\mathbf{r_{ij}} p_j}{n^2\beta}) - \frac{c_h}{\tau^2}\left(D_2 - \frac{\mathbf{r_{ij}} p_j}{n^2\beta}\right)^2. \tag{5.23}$$

• **Problem 2:** *Utility maximization of secondary Fog device:*

$$\max_{p_j} U_j(p_j)$$

Theorem 5.2 *The optimal price set by secondary FD for executing task portion τ_j is:*

$$p_j^\star = \frac{\lambda - \theta \sum_{k\neq j} \mathbf{r_{ik}} p_k}{\kappa(\mathbf{r_{ij}})}, \tag{5.24}$$

where, $\lambda = \frac{c_h}{n^2\beta\tau} - \frac{p_r}{n\beta} + \frac{\tau}{n}\left(1 + \frac{2ch}{n^2\beta\tau}\right) - \frac{D_1}{n\beta}\left(1 + \frac{2ch}{n^2\beta\tau}\right)$, $\kappa = \frac{2}{n^2\beta} + \left(1 + \frac{ch}{n^2\beta\tau^2}\right)$, *and* $\theta = \frac{1}{n^2\beta}\left(\frac{2c_h\mathbf{r_{ij}}}{(n^2\beta)^2\tau^2} + 1\right)$

Proof: The utility of s-FD can be obtained from Equation 5.23. The first order derivative of $U_j(p_j)$ *w.r.t* p_j can be written as:

$$\frac{dU_j(p_j)}{dp_j} = \mathbf{r_{ij}}\left(D_2 - \frac{\mathbf{r_{ij}}p_j}{n^2\beta}\right) - \left(\mathbf{r_{ij}}p_j - \frac{c_h}{T} + np_r\right)\frac{\mathbf{r_{ij}}}{n^2\beta} - \frac{2c_h r_{ij}}{n^2\beta\tau^2}\left(D_2 - \frac{\mathbf{r_{ij}}p_j}{n^2\beta}\right). \quad (5.25)$$

The second-order derivative of Equation 5.25 is given as follows:

$$\frac{d^2U_j(p_j)}{d^2p_j} = -\frac{r_{ij}^2}{n^2\beta}\left(2 + 2\frac{c_h}{n^2\beta\tau}\right). \quad (5.26)$$

As the second order derivative is negative, the utility function of the s-FD poses maxima at p_j for which $\frac{dU_j}{dp_j} = 0$. The matrix for equation for p_j can be written as:

$$p_j\mathbf{r_{ij}}\kappa + \sum_{k\neq j, k\neq i}\mathbf{r_{ik}}p_k\theta = \lambda, \quad (5.27)$$

where, $\lambda = \frac{c_h}{n^2\beta\tau} - \frac{p_r}{n\beta} + \frac{\tau}{n}\left(1 + \frac{2ch}{n^2\beta\tau}\right) - \frac{D_1}{n\beta}\left(1 + \frac{2ch}{n^2\beta\tau}\right)$, $\kappa = \frac{2}{n^2\beta} + \left(1 + \frac{ch}{n^2\beta\tau^2}\right)$, and $\theta = \frac{1}{n^2\beta}\left(\frac{2c_h\mathbf{r_{ij}}}{(n^2\beta)^2\tau^2} + 1\right)$. Let $\mathbf{A}$ and $\mathbf{\Lambda}$ be the coefficient matrix. The result can be expressed in matrix form as:

$$\overbrace{\begin{bmatrix} \kappa\mathbf{r_{i1}} & \mathbf{r_{i2}}\theta & \dots & \mathbf{r_{in}}\theta \\ \mathbf{r_{i1}}\theta & \kappa\mathbf{r_{i2}} & \dots & \mathbf{r_{in}}\theta \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{r_{i1}}\theta & \mathbf{r_{i2}}\theta & \dots & \kappa\mathbf{r_{in}}\theta \end{bmatrix}}^{\mathbf{A}} \overbrace{\begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ p_n \end{bmatrix}}^{\mathbf{p}} = \lambda \overbrace{\begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}}^{\mathbf{\Lambda}}. \quad (5.28)$$

By using strictly diagonal dominant theorem [115], $\mathbf{A}$ is non-singular matrix, iff $\kappa\mathbf{r_{ij}} \geq 0, \forall j \in n$. As $\mathbf{r_{ij}}$ is the reputation which is always greater than zero and $\kappa > 0$. It proves $\mathbf{A}$ is non-singular matrix. As, $\mathbf{A}$ is non-singular; thus, inverse of $\mathbf{A}$ is feasible. The optimal price set by the s-FDs for executing task portion τ_j are $\mathbf{p} = [p_1, p_2, \cdots, p_n]$.

Upon solving $\mathbf{p} = \mathbf{A}^{-1}\mathbf{\Lambda}$, we can obtain optimal price as:

$$p_j^{\star} = \frac{\lambda - \theta \sum_{k \neq j} \mathbf{r_{ik}} p_k}{\kappa(\mathbf{r_{ij}})}. \tag{5.29}$$

This ends the proof for the optimal price of secondary FD. □

5.4.3 Near Nash Equilibrium (NE) between s-FDs:

Let us assume a game having n s-FDs. The number of strategies or price taken by the s-FDs $(p_1, p_2, \cdots, p_j, \cdots, p_n)$ achieving NE if $\forall j \in n$:

$$U(p_j^*, p_{-j}^*) \geq U(p_j, p_{-j}). \tag{5.30}$$

This chapter propose a near-optimal price estimation algorithm (Algorithm 5.2) for determining the best response of s-FD (p_j^*, $\forall j \in n$) in polynomial time. Algorithm 5.2 iteratively estimates the value of p_j by maximizing the utility of the s-FD and halts in polynomial time with near optimal solution.

Definition 5.4 *(ϵ-NE): A set of strategies or price taken by the s-FDs $(p_1, p_2, \cdots, p_j, \cdots, p_n)$ achieving ϵ-NE if $\forall j \in n$*

$$U(p_j^*, p_{-j}^*) \geq U(p_j, p_{-j}) - \epsilon. \tag{5.31}$$

In case of ϵ-NE, the s-FDs have marginal incentive to deviate from NE. However, s-FDs do not increase their utility by more than ϵ. Algorithm 5.2 facilitates the s-FDs to plays its best response strategy (best price) by iterating until ϵ-NE is achieved. The iterative steps in Algorithm 5.2 always results in an improvement or consistent utility as earlier. The utility function of the s-FDs eventually converges to a fixed point, when the increment in its value is less than ϵ. Theorem 5.3 proves the existence of ϵ-NE in Algorithm 5.2, which converges in polynomial time. The complexity of message passing,

i.e., the required messages need to be communicated among the participants to reach the near-Nash Equilibrium (or ϵ-Nash Equilibrium) is $O(n^{\log n/\epsilon^2})$ [116].

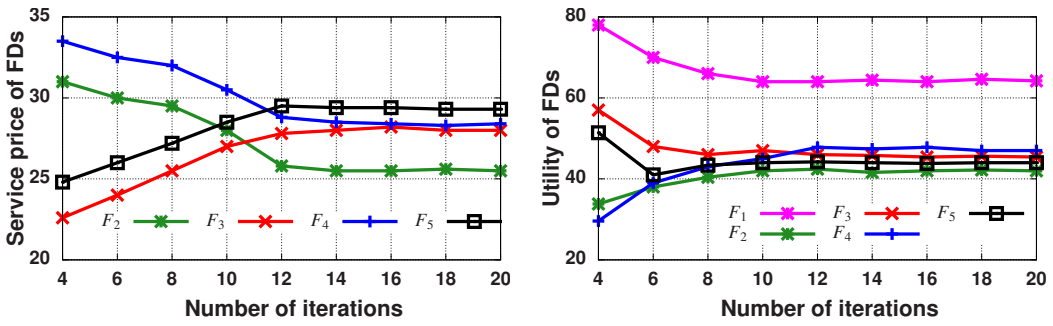
Algorithm 5.2: Iterative algorithm for ϵ-NE solution.

Input: $k, r_{ij}, c_h, n, \tau, p_j$, $\mathbb{C}$, precision threshold ϵ;
Output: Pricing strategies of s-FDs p_j;
1 $t \leftarrow 0$ /*Initialization of variable*/
2 Calculate τ_i from Equation 5.21.
/*Iterative loop to obtain near optimal solution*/
3 **do**
4 $t \leftarrow t + 1$.
5 Update $\tau_i \leftarrow {\tau_i}^{t-1}$.
6 /*Calculate p_j from Equation 5.29*/
7 $p_j[t+1] = \frac{\lambda - \theta \sum_{k \neq j} \mathbf{r_{ik}} p_k[t]}{\kappa(\mathbf{r_{ij}})}$.
8 **while** $(|U(p_j^{t+1}) - U(p_j^t)| > \epsilon)$;
9 **return** p_j.

Theorem 5.3 *Algorithm 5.2 achieves near NE (ϵ-NE) using $O(N/\epsilon)$ iterations, $\epsilon > 0$.*

Proof: According to the termination condition of Algorithm 5.2 in step 8, *i.e.,* $(|U(p_j^{t+1}) - U(p_j^t)| > \epsilon)$, the s-FDs change their strategies (p_j) if the change in utility is more than ϵ. This indicates that FDs increase their utility by atleast ϵ in each iteration. Therefore, the upper bound of the utility can be obtained in absence of competition. This implies that the utility of FDs would never reaches upper bound $U^\star(\cdot)$. Hence, Algorithm 5.2 must run at most $(U^\star(\cdot)/\epsilon)$ times. The maximum run of Algorithm 5.2 in worst case is $\max U^\star(\cdot)/\epsilon$. Thus, it proves time-complexity of Algorithm 5.2 as $O(N/\epsilon)$ and hence we reach to the end of proof. □

Example 1 *We consider* 5 *connected FDs out of which one is the p-FD and other are s-FDs. Initially, all s-FDs set their price randomly and afterwards in each iteration, according to the strategies of other FDs, they re-calculate their price to maximize the utility of all FDs. Figure 5.3(a) and Figure 5.3(b) illustrates that the s-FDs ($F_2 \cdots F_5$) iteratively update their service price and utilities. Algorithm 5.2 converges to a stable value after* 12 *iterations for* 4 *s-FDs that confirms its convergence and stability.*



(a) Service price (b) Utility

Figure 5.3: Iterations for convergence of Algorithm 5.2 with 5 FDs, p-FD (F_1) and s-FDs ($F_2 \cdots F_5$).

5.5 Prototype setup and performance evaluation

This section covers the description of the hardware and software used in the prototype setup for experimental evaluations. We also describe different parameters' impacts on the accuracy, delay, and system cost.

• **Prototype specification:** We used the resources of the lab for creating a prototype setup. A smartphone (Honor 7A smartphone: 13 megapixels camera with Wi-Fi 802.11 b/g/n), router (Cisco Linksys router: WRT54G supports IEEE 8203, 802.3 u/g/b), Personal Computer (Intel $i7$ processor and 8 GB memory), and Dell server (Intel Xenon processor and 192 GB memory) work as the leaf, Edge, Fog, and Cloud devices, respectively. Initially, this chapter considered 10 FDs in the system. Most of the other networking parameters used during the experiment are similar as given in [117, 118]. The smartphone is used for capturing the image of different buildings in the institute. The captured image is forwarded wirelessly to the PC in the lab using a router installed near the image acquisition point. Further, we use Dell server of the institute as Cloud. Figure 5.4 illustrates the prototype setup of TSF.

We have created a script file running on p-FD and calculates τ_i portion of the task. The p-FD also determines the portion τ_j to be executed on s-FDs using the same script. We preserve log files for estimating delay on every FD. The start log is triggered when a device starts computing received data and stops as the computation completes. We use

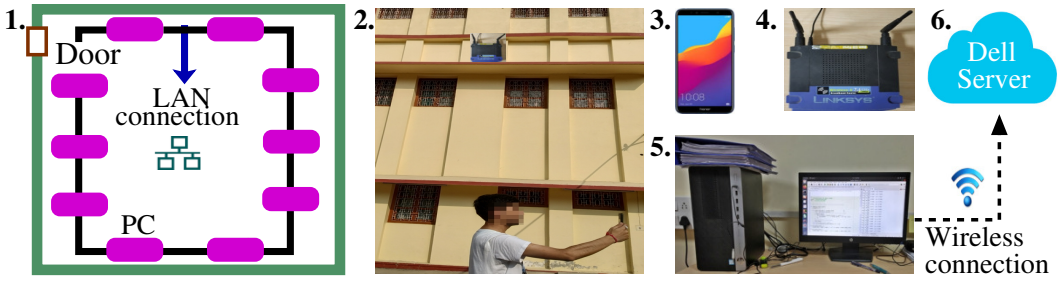


Figure 5.4: Prototype components: (1) PCs organization in the lab, (2) prototype deployment area, (3) smartphone, (4) router, (5) PC in lab, and (6) DELL server as Cloud.

a deep learning model (AlexNet [119]) trained on the images of different monuments and architectures in our locality. The training of the deep learning model is performed on the Cloud (server). The pretrained model on the Cloud is deployed on all the FDs interconnected with each other. We repeat each experiment 500 times to reach the confidence level of 95%; thus, error bars are not presented in results.

• **Schemes for experimental analysis:** We considered three schemes, namely only p-FD, only Cloud (CL), and proposed TSF as shown in Figure 5.5. In p-FD and CL schemes, the entire processings run on the primary FD and Cloud, respectively. In TSF scheme, the fraction of the process running at p-FD, s-FDs, and Cloud, is determined by Algorithm 5.1. The τ_i and τ_c denote the fraction of the task processes at p-FD and Cloud, respectively. We calculate the fraction of the task process at each s-FD and take the average, denoted by τ_j.

• **Overview of results:** Algorithm 5.1 shows that fraction of task depends upon different parameters, including data size, task deadline, number of FDs, and game parameters. In following section, we describe different parameters impacts on accuracy, delay, and system cost. The cost of the system is sum of the resources consume during processing of the task. Let a device i consume e_i power per unit time and run a fraction of the task for t_i time duration, where $1 \leq i \leq k$ and k is the total devices running for task processing. The system cost is therefore $\sum_{i=1}^{k} e_i t_i$ [117, 118]. The considered performance metrics are Precision(P), Recall(R), and F_1-score, where P =

$\frac{\text{True\_Positive}}{\text{True\_Positive + False\_Positive}}$, $\text{R} = \frac{\text{True\_Positive}}{\text{True\_Positive + False\_Negative}}$, and $\text{F}_1 = \frac{(2 \times \text{P} \times \text{R})}{(\text{P+R})}$.

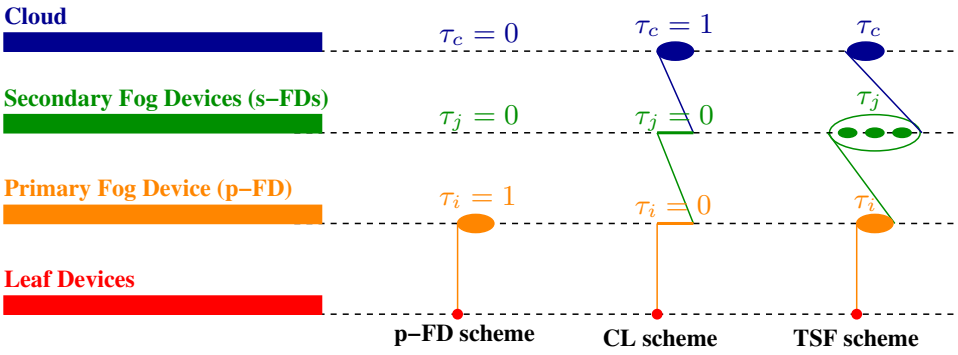


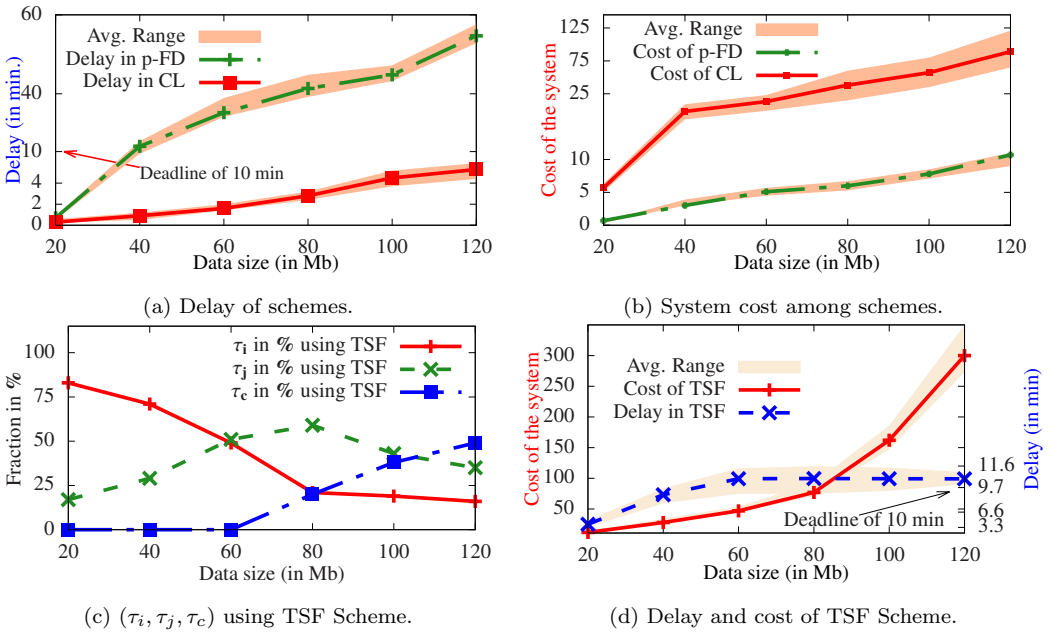
Figure 5.5: Illustration of p-FD, CL, and TSF schemes.

5.5.1 Impact of data size

We first study the impact of data size on system cost and execution delay using considered schemes. The data size range varies from $20 - 120$ Mb, with the task execution deadline of 10 min. Figure 5.6(a) and Figure 5.6(b) demonstrated the CL scheme incurs less delay and more cost in comparison with the p-FD scheme. The CL scheme required 42% higher energy and 55% less delay in contrast with p-FD when averaged over the data size range ($20 - 120$Mb). It is due to the colossal processing speed and significant energy requirement at the Cloud to that of p-FD. Since the Cloud is assumed to have enormous processing capability; therefore, the increase in data size does not contribute to the delay in task execution.

Figure 5.6(c) depicted the most suitable value of (τ_i, τ_j, τ_c) computed from Algorithm 5.1. We can observe that the low data size results in shifting of processing towards the p-FD ($\tau_i = 0.76, \tau_j = 0.24, \tau_c = 0$). However, the increase in data size tends the processing towards Cloud ($\tau_i = 0.20, \tau_j = 0.29, \tau_c = 0.51$). Figure 5.6(d) demonstrated the results obtained while using TSF. We obtained $\tau_c < \tau_j < \tau_i$. It indicates the task fraction of Cloud is low, indicating lower system cost. Further, beyond the data size of 60Mb, the proposed TSF scheme performs task offloading to the Cloud for meeting the deadline requirement. We observed that the p-FD scheme tends to miss

the task deadline for large size data. On the other hand, the CL scheme always satisfies the task deadline, requiring significant system cost in contrast with TSF. Moreover, TSF always satisfies the deadline requirement and incurs low execution cost.



(a) Delay of schemes.

(b) System cost among schemes.

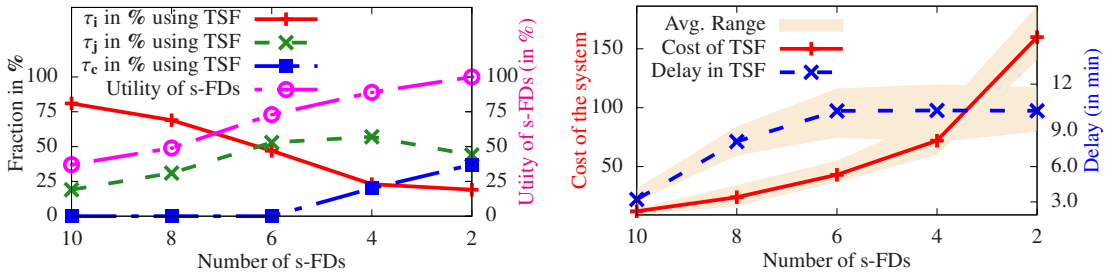
(c) (τ_i, τ_j, τ_c) using TSF Scheme.

(d) Delay and cost of TSF Scheme.

Figure 5.6: Impact of data size on delay and system cost using different schemes.

5.5.2 Impact of secondary Fog devices

Next, we discuss the impact of s-FDs count on delay, system cost, and utility of the FDs. As the number of s-FDs increases, a substantial portion of the task can be offloaded and executed in parallel. More number of s-FDs also reduced the execution at the Cloud and load on each FD, as shown in Figure 5.7(a). It also shown that more s-FDs reduces the utility of the FDs because each FD needs to solve a small fraction of task. The offloading and the higher parallel processing have reduced the delay of the task, as shown in Figure 5.7(b).



(a) (τ_i, τ_j, τ_c) using TSF Scheme.

(b) Delay and cost of TSF Scheme.

Figure 5.7: Impact of s-FDs count on delay, system cost, and FDs utilities.

5.5.3 Impact of deadline

This experiment demonstrates the impact of the deadline on the performance of the TSF system for a specific task. The task processing at p-FD within the given deadline needs high-order data compression. To facilitate data compression, we have incorporated the data aggregation technique. However, the compression results in accuracy compromise. Table 5.1 illustrated the Mean and Standard Deviation (SD) of different schemes using various performance metrics over 150 disjoint runs. Here, the accuracy achieved in both CL and TSF schemes is around 100%. Thus, the representation of SD for CL and TSF schemes is obsoleted from Table 5.1. Further, the Compression Ratio (CR) is defined as the ratio of compressed data size for a task to the original data size. It is interesting to observe that the Scheme p-FD can meet the deadline constraints but lag in achieving accuracy as CL and TSF. In addition, the TSF scheme achieved similar accuracy as CL and simultaneously reduces the execution cost and time.

5.5.4 Impact of game parameters

Further, this section discusses the impact of various game parameters on utility and Rate of Convergence (RoC). First, we illustrated the impact of the task portion on the game performance, computed at p-FD (τ_i). We considered three scenarios where the number of s-FDs are three, five, and eight. Figure 5.8 shown the impact of τ_i, p_j, c_h, and $\mathbb{C}$ on the average utility of p-FD and s-FDs. The results illustrated average utility of p-

Table 5.1: Accuracy metrics for p-FD, CL, and proposed schemes.

		p-FD scheme		**CL** scheme	**TSF** scheme
		Mean	**SD**	**Mean**	**Mean**
20Mb	**P**	0.92	0.001	0.98	0.98
	R	0.85	0.002	0.99	0.97
	F_1	0.88	0.002	0.99	0.98
	CR	19.5%	0.002	0%	0%
60Mb	**P**	0.83	0.004	0.98	0.98
	R	0.88	0.003	0.99	0.99
	F_1	0.85	0.002	0.98	0.97
	CR	27.3%	0.002	0%	0%
120Mb	**P**	0.74	0.004	0.98	0.98
	R	0.77	0.003	0.98	0.97
	F_1	0.75	0.002	0.99	0.98
	CR	38.7%	0.003	0%	0%

FD and s-FDs when we consider scenarios of three, five, and eight s-FDs. Figure 5.8(a) shown that the average utility of p-FD linearly increases with the increase in τ_i because least cost is paid to the s-FDs and the p-FD preserves maximum profit. The average utility of the p-FD decreases as the price to be paid for fractional task execution is high, as shown in Figure 5.8(b). Average utility of p-FD and s-FDs decrease with the increase in the price constants c_h and $\mathbb{C}$. Rate of convergence of Algorithm 5.2 with game parameters τ_i, p_j, c_h, and $\mathbb{C}$ is also illustrated for different number of s-FDs. Figure 5.8(a) demonstrated the rate of convergence increases with the increase in τ_i. Other three parameters have a reverse impact on the convergence rate, as demonstrated in Figure 5.8(b), Figure 5.8(c), and Figure 5.8(d). The price constant c_h incurs on the p-FD and s-FDs while executing the fraction of the task on these devices. Therefore, if the price constant c_h increases, the utilities of p-FD and s-FDs decrease. c_h can be inferred as the number of CPU cycles, processing energy, and execution time.

5.5.5 Comparison with existing work

This section compares TSF with existing approaches including [18], [43], and [120]. To make the comparison fair, we use our prototype setup to evaluate and compare the pro-

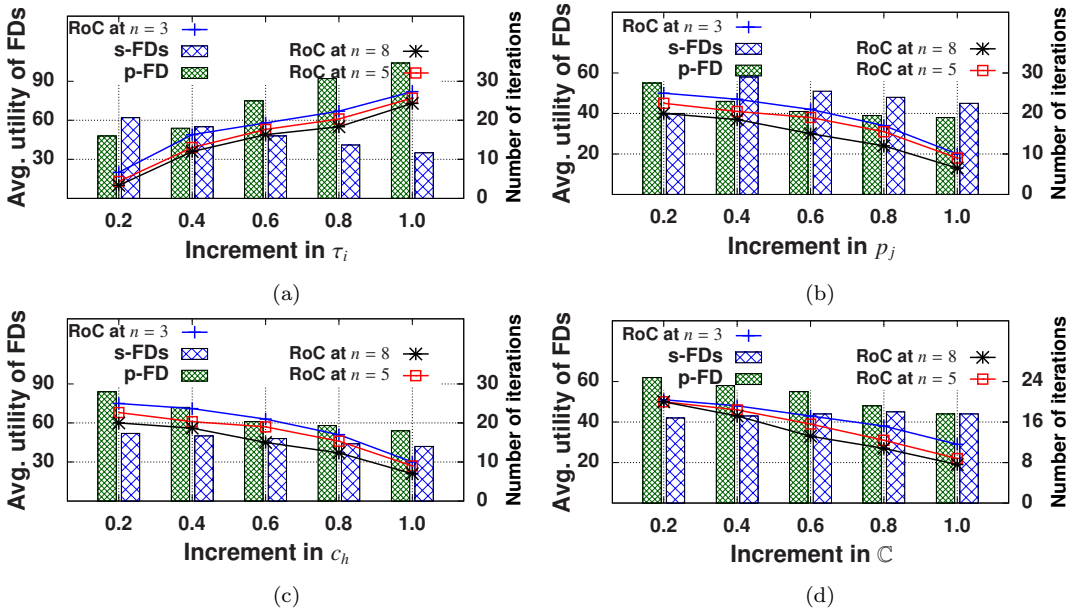
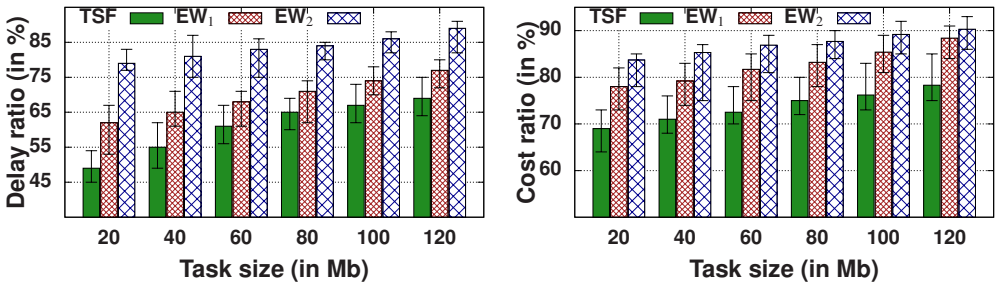


Figure 5.8: Impact of change in τ_i, p_j, c_h, and $\mathbb{C}$ on the utilities of the p-FD and s-FDs .

posed work with existing offloading schemes. Authors in [18] proposed a communication scheme that enabled data processing in a distributed manner on the connected vehicle. Liu *et al.* in [43] investigated a cost minimization model for scheduling multi-level task using Fog computing architecture. In [120], authors aim at maximizing the users quality of experience by determining optimal task offloading decision among different FDs or Cloud. We perform the comparison using delay and cost ratio with respect to [120]. We considered the scheme [120] because it incurs maximum delay and system cost. The delay ratio is estimated as delay ratio $= \frac{\text{delay in considered approach}}{\text{delay in [119]}}$ and system cost ratio is formulated as system cost ratio $= \frac{\text{cost incur in considered approach}}{\text{system cost of [119]}}$. Figure 5.9(a) illustrates the delay of the system in completing a task for the proposed work and the existing work. Least delay is observed for the proposed system as the task is computed in parallel on the connected FDs. Cost of computing the task is also minimum as least amount of data is offloaded and computed at the Cloud, as shown in Figure 5.9(b).



(a) Delay

(b) System cost

Figure 5.9: Illustration of comparison with existing work: EW_1 [18] and EW_2 [43] on task offloading in Fog computing.

5.6 Conclusion

This chapter proposed a transportation system incorporating Fog computing for passenger assistance. Unlike the existing work, TSF considered offloading portions of the task to the neighboring FDs for its parallel execution within a time constraint. We first presented neighboring FDs selection criteria to identify the best suitable FDs for offloading the task. A competitive game is then formulated in which the primary FD decides the fractions of task it offloads to the secondary FDs. We further proposed a Knapsack based algorithm to modify the offloading decision as per current resources availability. TSF is validated by setting up a prototype and performing various experiments to study the impact of task execution deadline, data size, and various game parameters. We find the following conclusions from this work: offloading task to several of Fog devices are suitable only when data size is huge; processing the data at Cloud is required when Fog devices are not sufficient to process it within given time constraints; the system must be able to handle unequal processing power of FDs, and reputation of the devices provide confidence for successful completion of a given task.