Chapter 3

Criticality and Utility-Aware Fog Computing System for Remote Health Monitoring

3.1 Introduction

In remote health monitoring system, a patient is equipped with WBAN sensors, capable of collecting health data, and transmitting it to an LD [38,84,85] for further processing. The LD stores and forwards the health data to a FS for remote health monitoring over 5G networks [15, 42], as shown in Fig. 3.1. However, rural people have a higher rate of poverty [86] and thus, they cannot afford LDs or IoT sensors on their own. Thus, there is a need for a low cost remote health monitoring system. However, the profit of MC should also be considered for patients' health monitoring services to encourage the participation of MCs.

Due to criticality, medical data has to be monitored on time without delay involved. For example, health data of patients with chronic illnesses like lung and heart diseases need real-time and continuous assessment; and their monitoring should be prioritized over other diseases. Many critical sensitive diseases' data cannot be computed on

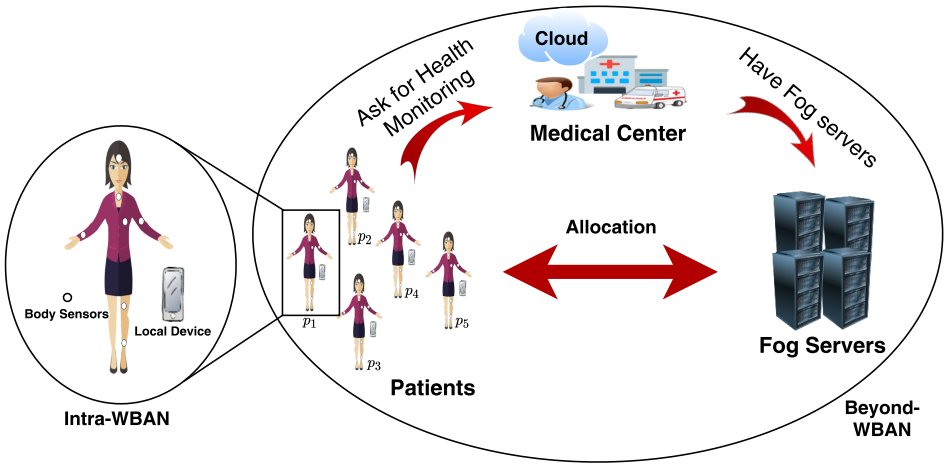


Fig. 3.1. Remote health monitoring system.

low resourced LDs while achieving desired delay constraint. Hence, assistance of FS has emerged to compute patients' health data efficiently while achieving desired delay constraint [87]. Table 3.1 [38], [88] provides data size and desired delay of different WBAN sensors based on the IEEE Standard 802.15.6-2012. From the table, it can be seen that if all sensors (i.e., ECG, EMG, artificial retina, audio and video) sense data from a patient, and send it to an LD for computation. The desired delays of artificial retina, audio and video do not achieve while doing so[1]. However, FS can reduce latency to a greater extent, allowing real-time remote health monitoring within a desired delay [6, 7].

Table 3.1: Different types of WBAN sensors

Sensor type	Data size	Desired delay (ms)	Delay at LD (ms)	Delay at FS (ms)
ECG	72 kb	250	208.33	4.53
EMG	80 kb	250	214.52	4.68
Artificial retina	175 kb	250	287.86	6.35
Audio	0.25 Mb	100	345.79	7.66
Video	2.5 Mb	100	2083.33	47.17

Motivated by the above scenarios, we propose intra-WBAN and beyond-WBAN based system. Intra-WBAN consists of sensors deployed on the patients' bodies, and the LD collects data from them, whereas beyond-WBAN consists of different LDs that

[1]Evaluation parameters are given in Table 3.2.

send patients' data to FSs. Moreover, we formulate FS assisted beyond-WBAN based remote health monitoring system to minimize the cost of patients while keeping profit of MC into deliberation. Inspired by [8], we aim to use dedicated LDs not only to gather the patients' health data sent by sensors but also to compute the data locally while achieving desired delay. In nutshell, contributions of this chapter are summarized as below:

- Formulate an optimization problem by maximizing the system utility, defined as a linear combination of MC's profit and patients' cost. Moreover, offer a flat-type pricing scheme to measure the profit of MC.
- Propose swapping-based heuristic to maximize the system utility under the constraint of permissible latency for computation of patients' data in polynomial time complexity.
- Through extensive simulations and analysis on real-world data, proposed heuristic is found to achieve an average utility of 94.5% of the optimal solution.

3.2 System Model and Problem Formulation

We consider a remote health monitoring system, provided by MC to a set of patients $\mathbb{P} = \{1, \ldots, p \ldots, P\}$[2] in coordination with FSs, as shown in Fig. 3.1. The proposed framework is equivalent to student project assignment problems in colleges where students approach a professor for project assignment and the professor assigns different projects to the students (see Fig. 3.2). However, there are limits on the total number of students a professor can allocate

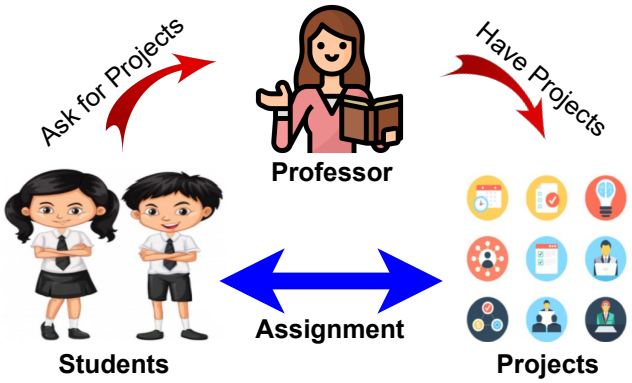


Fig. 3.2. Student project assignment.

[2]Key symbols and their descriptions are provided in the list of symbols.

and the number of students assigned to a project. Similarly, patients (LDs) request an MC for health monitoring, and the Cloud Server (CS) employed by the MC allocates patients' data to FSs (see Fig. 3.1). However, there are limits on the number of patients that a CS can allocate as well as the number of patients whose data can be allocated to an FS.

The problem setting is divided into two parts: intra-WBAN and beyond-WBAN as described in the following:

3.2.1 Intra-WBAN

Consider a set of sensors $\mathbb{S} = \{1, \ldots, s, \ldots, S\}$ deployed on patient's body. Each sensor collects data with different criticality classes and transmits it to an LD for further analysis. To facilitate this phenomenon, we consider medical criticality for prioritizing different health data in a resource-constrained health monitoring scenario.

Let the medical criticalities of health data collected by sensors be a set $\mathbb{X} = \{x_1, \ldots, x_s, \ldots, x_S\}$. If the data collected by sensor s is more critical than that of sensor s', then $x_s > x_{s'}$. Two sensors' data of the same criticality class can have different medical criticalities; thus $x_s \in [0, \infty)$ [8]. Let θ_p^s be the health parameter value sensed by sensor s from patient p, and let $\theta_{low,s}$ and $\theta_{up,s}$ be lower and upper limits of reference range- defined as the range of health parameter values under normal conditions for a healthy person[3][4]. These reference range helps determine whether a person's physiological health parameters fall within a normal, healthy range. In other words, they serve as a reference to detect potential health problems or imbalances in the body. For instance, in a medical scenario, the normal body temperature range for a healthy person is between 34.1 °C and 37.9 °C, with 34.1 °C as the minimum and 37.9 °C as the maximum [89]. A body temperature outside this range may indicate a fever (high

[3]A detailed explanation for calculating $\theta_{low,s}$ and $\theta_{up,s}$, i.e., lower and upper limits of reference range is given in Appendix A of [38].

[4]We considered a time-slotted system where everything occurs at time slot t. For simplicity, the time index t is omitted from all variables, assuming they all refer to the time slot.

temperature) or hypothermia (low temperature) [90]. Then, the health severity index of patient p's data collected by sensor s is defined as follows [38, 91]:

$$\mathfrak{s}_p^s = \left| \frac{(\theta_{up,s} - \theta_p^s)^2 - (\theta_p^s - \theta_{low,s})^2}{(|\theta_{up,s}| + |\theta_{low,s}|)^2} \right|. \tag{3.1}$$

Health severity index defines the deviation of a patient's health data value from its normal reading. Higher health severity index indicates more severe data. For instance, ECG data is more critical than temperature data in healthcare [91]. We further define criticality index, c_p^s for a patient p and sensor s as the product of health severity index and medical criticality as follows:

$$c_p^s = x_s \mathfrak{s}_p^s. \tag{3.2}$$

Moreover, LD normalizes the health severity index between 0 and 1 using the min-max normalization technique [92]. Then, p^{th} patient's criticality is defined as the average of criticality indices of sensors, as follows:

$$\rho_p = \frac{1}{S} \sum_{s=1}^{S} c_p^s. \tag{3.3}$$

A higher criticality value indicates a more severe condition of a patient. After receiving data at LD, there is a need to make a decision for its computation either at LD or FS in order to achieve the system's constraints. Moreover, computation capacity of all LDs is considered to be uniform and equal to Υ. Let u_p be a binary variable defined as:

$$u_p = \begin{cases} 1, & \text{system selects LD for computation of patient } p\text{'s data;} \\ 0, & \text{otherwise.} \end{cases} \tag{3.4}$$

Computation time at LD for a patient p is calculated as:

$$T_p^{c,l} = \frac{\beta_p}{\Upsilon}, \tag{3.5}$$

where β_p is the required CPU cycles for computing patient p's health data. In addition, we assume that each sensor is allocated a different amount of LD's resources.

In the next section, we discuss the transmission of health data to the FS[5] and its computation at the FS.

3.2.2 Beyond-WBAN

Let a set of FSs $\mathbb{F} = \{1, \ldots, f, \ldots, F\}$ be equipped with a respective computation capacity of a set $\mathbb{L} = \{\Gamma_1, \ldots, \Gamma_f, \ldots, \Gamma_F\}$. However, to determine whether or not the system selects FS for computation of p's data, we define a binary variable q_p as follows:

$$\mathrm{q}_p = 1 - u_p. \tag{3.6}$$

Let $\mathbb{H}$ be a $P \times F$ binary matrix that indicates the choice of FS for computing patient p's data, as given below:

$$\mathbb{H} = \begin{bmatrix} h_1^1 & h_1^2 & \cdots & \cdots & h_1^F \\ h_2^1 & \cdots & \cdots & \cdots & \cdots \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ h_P^1 & \cdots & \cdots & \cdots & h_P^F \end{bmatrix}, \tag{3.7}$$

$$h_p^f = \begin{cases} 1, & \text{FS } f \text{ computes patient } p\text{'s data;} \\ 0, & \text{otherwise.} \end{cases} \tag{3.8}$$

Transmission rate between patient p and FS f underlying cellular 5G is computed as follows [15]:

$$r_p^f = \omega V_p^f \log_2 \left(1 + SINR_p^f\right), \tag{3.9}$$

[5]Transmission of data in intra-WBAN is beyond the scope of this chapter. However, the existing approach [8], can be utilized for intra-WBAN communication.

where ω, V_p^f and $SINR_p^f$ are channel bandwidth, number of allocated PRBs[6] and Signal-to-Interference-plus-Noise Ratio (SINR)[7] between patient p and FS f, respectively. We assume each patient communicates over a distinct channel; thus, interference is not considered[8].

Transmission time between patient p and FS f can be:

$$T_p^{tr,f} = \frac{\eta_p}{r_p^f}, \tag{3.10}$$

where η_p is the size of patient p's data. Like [94], [95], we assume that each patient uses the same amount of FS's resources. Thus, computation time for a patient p at FS f can be calculated as:

$$T_p^{c,f}(\mathbb{H}) = \frac{\beta_p}{\gamma_p(\mathbb{H})}, \tag{3.11}$$

where $\gamma_p(\mathbb{H})$ is the fraction of FS f's resource utilized by the patient p, calculated as follows:

$$\gamma_p(\mathbb{H}) = \frac{\Gamma_f}{\mathrm{n'}_{p,f}}, \tag{3.12}$$

where $\mathrm{n'}_{p,f}$ is the number of patients utilizing the FS f for their computation if patient p is also utilizing it without violating the constraints.

One of our objectives is to minimize the cost of the patients, defined as the weighted (i.e., criticality) sum of computation and transmission time as follows:

$$\mathscr{H} = \sum_{p\in\mathbb{P}} \rho_p \left(\sum_{f\in\mathbb{F}} \left(h_p^f T_p^{tr,f} + T_p^{c,f}(\mathbb{H}) \right) + u_p T_p^{c,l} \right). \tag{3.13}$$

Thus, to minimize the cost, we have to lower down the latency for the patient in the beyond-WBAN scenario.

[6]PRB is the smallest unit of radio resource that can be assigned to a device [15].

[7]We assume that the channel exhibits flat fading. However, this can be easily extended to frequency-selective fading channels as well [15].

[8]Interference can be solved by applying methods given in [15], [93].

To determine revenue model of MC, we consider a flat-type pricing[9] scheme [96, 97] as described in the following:

3.2.2.1 Flat-type pricing scheme

Let the MC charges l unit price per time slot for computation at LD. If computation is done at FS, the MC charges $\mathfrak{m}$ unit price per time slot. Generally, FS charges more than that of LD, i.e., $\mathfrak{m} > l$. Thus, revenue of the MC is calculated as:

$$\chi = \sum_{p \in \mathbb{P}} (u_p l + \mathrm{q}_p \mathfrak{m}). \tag{3.14}$$

Let k be the fixed expenses of each FS per time slot and $\mathfrak{g}$ be the expenses of MC per CPU cycle for computation on the FS. Then, the expenses of MC can be calculated as follows:

$$\phi = kF + \mathfrak{g} \sum_{p \in \mathbb{P}} \mathrm{q}_p \beta_p. \tag{3.15}$$

Now, the profit of the MC can be calculated as follows:

$$\Delta = \chi - \phi = \sum_{p \in \mathbb{P}} (u_p l + \mathrm{q}_p \mathfrak{m}) - kF - \mathfrak{g} \sum_{p \in \mathbb{P}} \mathrm{q}_p \beta_p. \tag{3.16}$$

The profit of MC from a patient p depends on whether the patient p's data is allocated to an FS or LD. Let the maximum possible value of β_p be β_{max} (β_{max} can be approximated by the MC before deciding the values of $\mathfrak{m}$ and l), then profit of the MC can follow the following constraint:

$$\mathfrak{m} - l \geq \mathfrak{g} \beta_{max} + \frac{kF}{P}. \tag{3.17}$$

The above constraint ensures that if a patient's data is allocated to an FS, then the

[9] We adopt flat pricing scheme due to its simplicity and applicability in scenarios where data size variations are not significant. Dynamic pricing, which is based on the computational resource utilization of patients' data, is beyond the scope of Chapter 3. However, it is discussed in Chapter 4, where we propose a dynamic pricing scheme that accounts for the computational requirements of patients' health data to deliver health monitoring services effectively.

profit of the MC will be higher than if it is allocated to LD, independent of the CPU cycles needed for computing patients' data, as stated in Lemma 3.1.

Lemma 3.1 *The profit of the MC either increases or remains constant as more patients' data is allocated to FS instead of LD for their computation.*

Proof: Let P' be the number of patients whose data is allocated to FS. Then, the profit of the MC is given by:

$$\Delta_1 = P'\mathfrak{m} + (P - P')l - kF - \mathfrak{g}\sum_{p\in\mathbb{P}} \mathrm{q}_p\beta_p. \tag{3.18}$$

Take any patient p' utilizing LD and allocate it's data to any FS for computation of health data. So, the new profit in this scenario (assuming allocation of all other patients' data remains the same) is given by:

$$\Delta_2 = (P' + 1)\mathfrak{m} + (P - P' - 1)l - kF - \mathfrak{g}\sum_{p\in\mathbb{P}} \mathrm{q}_p\beta_p - g\beta_{p'}. \tag{3.19}$$

Now, $\Delta_2 - \Delta_1$ is given by:

$$\Delta_2 - \Delta_1 = \mathfrak{m} - l - \mathfrak{g}\beta_{p'}. \tag{3.20}$$

From Eqs. (3.17) and (3.20), we have:

$$\Delta_2 - \Delta_1 \geq 0. \tag{3.21}$$

From Eq. (3.21), we conclude that the profit increases or remains constant as we increase the number of patients whose data is allocated to FSs for computation. □

As per Lemma 3.1, the profit only depends on patients whose data is allocated to FSs. So, if all patients' data is allocated to FSs, the profit does not depend on how their

data is allocated to FSs. Thus, the utility depends only on the cost of the patients as defined in Eq. (3.13).

3.2.3 Problem Formulation

In remote health monitoring system, the patient with a higher criticality should be monitored in less time. Thus, lower computation and transmission time are required for the patients to minimize their costs and ensure proper monitoring. Moreover, MC tries to maximize its profit by providing monitoring services under the condition that no patient faces any delay. However, both objectives cannot be achieved at the same time. Therefore, we consider utility as the linear combination of profit of the MC and the cost of patients, that signifies the importance of profit for the monitoring service as well as the cost of patients, as discussed in the following:

$$\mathcal{V} = \lambda_1 \Delta - \lambda_2 \mathscr{H}, \tag{3.22}$$

where λ_1 and λ_2 are positive weights assigned to the profit of MC and the cost of patients, respectively, and $\lambda_1 + \lambda_2 = 1$. The weights are taken as inverse units of profit and latency cost, respectively, so that utility becomes unit-less. The values of λ_1 and λ_2 depend on the system requirements and priorities in real-world deployments. If the system prioritizes profit maximization, λ_1 should be set higher than λ_2 (i.e., $\lambda_1 > \lambda_2$), thereby giving greater importance to the profit of the MC. Conversely, if the system emphasizes criticality awareness, λ_1 should be set lower than λ_2 (i.e., $\lambda_1 < \lambda_2$), focusing on criticality-aware decision-making. For a system aiming to balance both profit and criticality equally, λ_1 and λ_2 can be set to equal values (i.e., $\lambda_1 = \lambda_2$). Thus, these weights should be chosen based on the system being considered. Furthermore, we consider a constraint on the permissible latency, denoted as δ, to ensure that no patient has a delay more than $\frac{\delta}{\rho_p}$. As a result, the permissible latency should be lower for higher criticality patients. Therefore, the optimization problem of the proposed model

is formulated as follows:

$$\mathbf{P1:} \arg\max_{\mathbb{H}} \ \mathcal{V} \tag{3.23}$$

Subject to the constraints:

$$\sum_{f\in\mathbb{F}} \left(h_p^f T_p^{tr,f} + T_p^{c,f}(\mathbb{H})\right) + u_p T_p^{c,l} \leq \frac{\delta}{\rho_p}, \forall p \in \mathbb{P}, \tag{3.23a}$$

$$l \leq l_{max}, \mathfrak{m} \leq \mathfrak{m}_{max}, \tag{3.23b}$$

$$\mathfrak{m} - l \geq \mathfrak{g}\beta_{max} + \frac{kF}{P}, \tag{3.23c}$$

$$\sum_{f\in\mathbb{F}} h_p^f = \mathrm{q}_p, \forall p \in \mathbb{P}, \tag{3.23d}$$

$$\sum_{p\in\mathbb{P}} h_p^f \leq F_f^{max}, \forall f \in \mathbb{F}, \tag{3.23e}$$

$$\sum_{p\in\mathbb{P}} \sum_{f\in\mathbb{F}} h_p^f \leq \mathscr{K}_{max}. \tag{3.23f}$$

Eq. (3.23a) refers to the latency constraint. Eq. (3.23b) puts constraint on service charges. Eq. (3.23c) refers to constraint defined in Eq. (3.17). Eq. (3.23d) ensures that every patient's data is allocated to at most one FS. Eqs. (3.23e) and (3.23f) define the maximum number of patients whose data can be allocated to an FS and the maximum number of patients whose requests can be handled by the CS, respectively.

The formulated problem **P1** is a Binary Integer Programming problem in $\mathbb{H}$ decision variables, that is generally NP-hard to solve as its feasibility problem is strongly NP-complete [98]. Due to high conditionality and complexity of the formulated problem, this chapter proposes a sub-optimal solution for the maximization problem based on swapping-based heuristic in the following section.

3.3 Proposed Solution

To avoid high computation charges at FS, each patient would like to compute the health data at LD. However, they may not be able to satisfy the latency constraint given in Eq. (3.23a) while doing so (see Table 3.1). Thus, a sub-problem here is to allocate these patients' data to FSs. Let $\mathbb{P}^v$ be the set of patients that violate the latency constraint if their data is computed at the LD. Formally,

$$\mathbb{P}^v = \{p \in \mathbb{P} : \rho_p T_p^{c,l} > \delta\}. \tag{3.24}$$

The MC allocates patients from the set $\mathbb{P}^v$ to FSs. Next objective is to allocate a subset of the remaining patients such that it maximizes the utility under the system constraints. Due to limited resources, it is not possible to allocate all of the patients' data to FSs. Doing so may result in violation of the latency constraint given in Eq. (3.23a) and significant increase in the cost of patients, resulting in lower utility. Let $\mathrm{n}_{p,f}^{max}$ be the maximum number of patients (see Theorem 3.1) that can utilize the FS f for their computation if patient p utilizes it without violating the constraint given in Eq. (3.23a).

Theorem 3.1 *Maximum number of patients that can utilize the FS f for their computation, if patient p utilizes FS f, is given by:*

$$\mathrm{n}_{p,f}^{max} = \left\lfloor \left(\frac{\Gamma_f}{\beta_p}\right) \left(\frac{\delta}{\rho_p} - \frac{\eta_p}{\omega V_p^f \log_2\left(1 + SINR_p^f\right)}\right) \right\rfloor. \tag{3.25}$$

Proof: According to Eq. (3.23a), if a patient p utilizes FS f, then,

$$T_p^{tr,f} + T_p^{c,f}(\mathbb{H}) \leq \frac{\delta}{\rho_p}. \tag{3.26}$$

From Eqs. (3.10) and (3.11), we get

$$\frac{\eta_p}{r_p^f} + \frac{\beta_p}{\gamma_p(\mathbb{H})} \leq \frac{\delta}{\rho_p}. \tag{3.27}$$

After solving the inequality and putting the value of r_p^f,

$$\mathrm{n'}_{p,f} \leq \left(\frac{\Gamma_f}{\beta_p}\right)\left(\frac{\delta}{\rho_p} - \frac{\eta_p}{\omega V_p^f \log_2\left(1 + SINR_p^f\right)}\right). \tag{3.28}$$

Thus, the maximum value of $\mathrm{n'}_{p,f}$ is the greatest integer value of the right-hand side expression. Hence, proved. □

We employ *Brakerski-Gentry-Vaikuntanathan* (BGV) [99] homomorphic encryption scheme in the proposed health monitoring system between LD and FS to maintain the privacy and security of the health data. Here, LD encrypts patient p's data before transmitting it to FS. Our proposed model employs encryption parameters as suggested in [100].

To solve the formulated problem, we propose Utility Maximization Patient Monitoring (UMPM) algorithm. We consider that the MC has CS that executes the UMPM algorithm to allocate patients' data to FSs. Fig. 3.3 shows data flow between patients, LDs, FSs, and MC. Labels 2, 3, 4 and 7 in Fig. 3.3 are handled by control signals (e.g., beacons [15]). However, labels 1, 5 and 6 are handled by data signals. UMPM algorithm begins with an initial allocation of patients' data to FSs and then iteratively re-positioning the patients' data by swapping their allocated FSs to achieve higher utility as shown in Fig. 3.4. The elaboration of each sub-algorithm is described in the following:

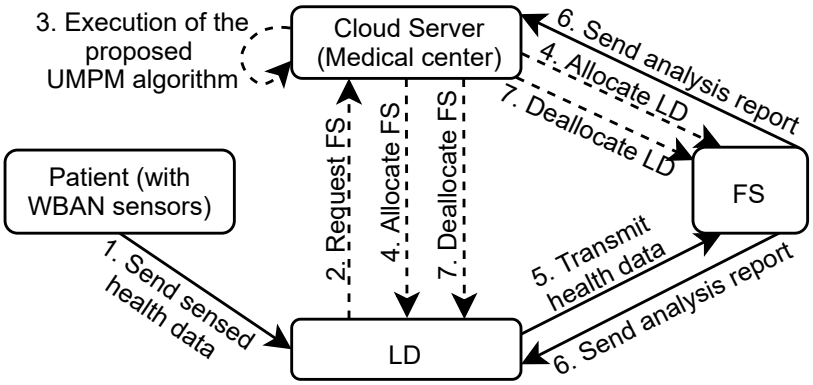


Fig. 3.3. Data flow diagram of remote healthcare architecture.

3.3.1 UMPM Algorithm

We define $\mathcal{V}_{diff}$[10] as a difference between utility before and after a patient p's data is allocated to FS f, keeping the allocation of other patients' data as it is as follows:

$$\mathcal{V}_{diff} = \lambda_1(\mathfrak{m} - l - \mathfrak{g}\beta_p) - \lambda_2 \sum_{p' \in \mathbb{P}^f} \frac{\rho_{p'}\beta_{p'}}{\Gamma_f} + \lambda_2 \left(\rho_p \left(\frac{\beta_p}{\Upsilon} - \left(\frac{\beta_p(\mathrm{n}^f + 1)}{\Gamma_f} + \frac{\eta_p}{r_p^f} \right) \right) \right). \tag{3.29}$$

Set of patients whose data get allocated to FS f is given by:

$$\mathbb{P}^f = \{p \in \mathbb{P} : h_p^f = 1\}. \tag{3.30}$$

Moreover, the number of patients whose data is allocated to FS f can be estimated as follows:

$$\mathrm{n}^f = \sum_{p \in \mathbb{P}} h_p^f. \tag{3.31}$$

The UMPM algorithm begins by calculating the constraint parameter $\mathrm{n}_{p,f}^{max}$ (Eq. (3.25)). It selects the patients that violate latency constraint if their data is computed

[10]The profit of p changes and it affects the cost with a value of the difference of p's local computation time and its transmission and computation time at FS f. Moreover, computation time of other patients whose data is allocated to f also changes.

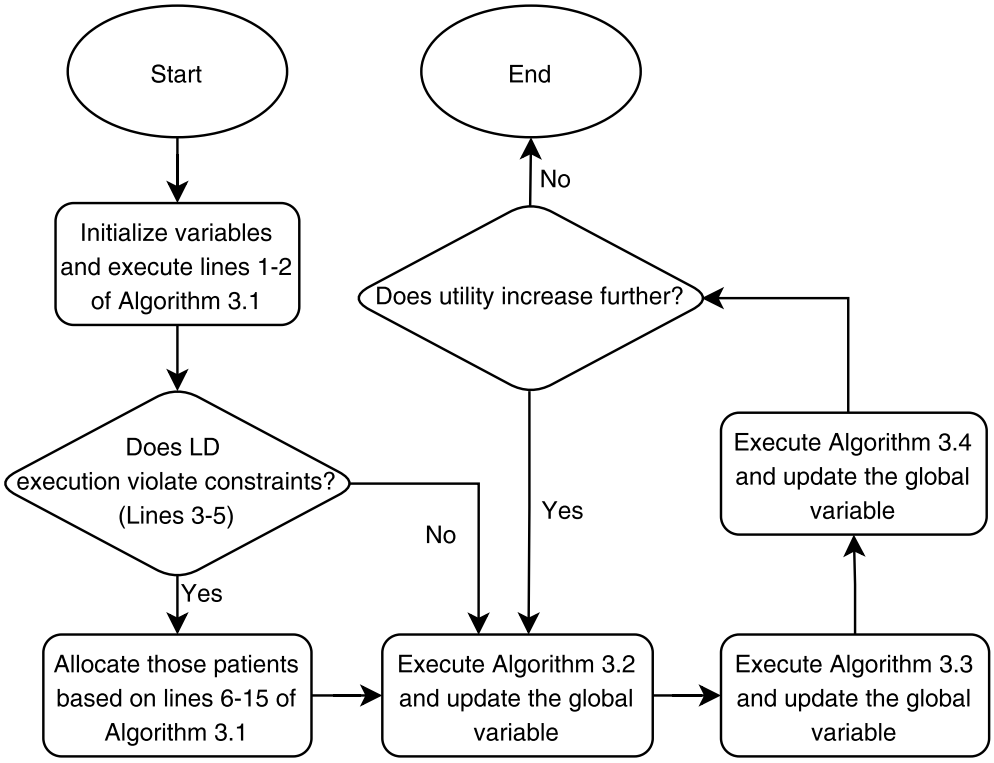


Fig. 3.4. Flow chart of UMPM algorithm.

on LD and sorts them in the order of their decreasing criticalities (lines 3-6). As a result, the algorithm prioritizes the patients with higher criticalities over the lower criticality patients. Then, re-allocation is done using Algorithms 3.2 and 3.3. Then, the algorithm constructs the set of patients whose data is not yet allocated to any FS (line 18). It calls Algorithm 3.4 to allocate more patients' data to FSs. Then, Algorithms 3.2 and 3.3 re-position the allocation of patients' data to FSs. The execution order of the Algorithms 3.2 and 3.3 does not affect the outcome of UMPM algorithm (see Fig. 3.12a). Further, Algorithm 3.4 allocates more patients' data to FSs by improvising the utility. This process repeats until there is no possibility of increment in the utility (Fig. 3.4). The reason for having Algorithms 3.2 and 3.3 is to obtain better utility by swapping the allocation between patients and FSs as described in Subsections 3.3.2 and 3.3.3, respectively.

Algorithm 3.1: UMPM Algorithm

Input: $\mathbb{L}$, Υ, $\mathfrak{g}, \mathfrak{m}, l, \delta$, $\mathbb{H} = \{0\}$, ω; $\forall p \in \mathbb{P}$: ρ_p, β_p, η_p; $\forall p \in \mathbb{P}, \forall f \in \mathbb{F}$: $SINR_p^f$; $\forall f \in \mathbb{F} : \mathbb{P}^f, \mathrm{n}^f = 0$; $\mathcal{V}_{diff}^{max} = -\infty$

Output: Allocation Strategy ($\mathbb{H}$)

```
Calculate n_{p,f}^{max}, ∀p ∈ P and ∀f ∈ F using Theorem 3.1;
Calculate local computation time using Eq. (3.5);
for p ← 1 to P do
    if T_p^{c,l} > δ/ρ_p then
        insert p into P^v;
Sort patients in set P^v in decreasing criticality order;
for p ∈ P^v do
    for f ∈ F do
        if n^f ≥ min_{p'∈P^f ⋃ {p}} n_{p',f}^{max} then
            continue;
        Calculate V_diff as per Eq. (3.29);
        if V_diff > V_diff^{max} and Eq. (3.23f) satisfied then
            V_diff^{max} ← V_diff;
            temp_p ← p, temp_f ← f;
    Allocate temp_p to temp_f, update variable ;        // Binary variable h_p^f
Run Algorithm 3.2;
Run Algorithm 3.3;
P^{rem} ← P − P^v;
Run Algorithm 3.4;
repeat
    Run Algorithm 3.2;
    Run Algorithm 3.3;
    Update P^{rem};
    Run Algorithm 3.4;
until Utility does not increase;
```

3.3.2 Two Way Swap based Algorithm

Utility difference due to two way swap, $\mathscr{H}_{diff}^{tw}$[11] is given as:

$$\mathscr{H}_{diff}^{tw} = \frac{\rho_p \eta_p}{r_p^f} - \frac{\rho_p \eta_p}{r_p^{f'}} + \frac{\rho_{p'} \eta_{p'}}{r_{p'}^{f'}} - \frac{\rho_{p'} \eta_{p'}}{r_{p'}^{f}} + \frac{\rho_p \beta_p \mathrm{n}^f}{\Gamma_f} - \frac{\rho_p \beta_p \mathrm{n}^{f'}}{\Gamma_{f'}} + \frac{\rho_{p'} \beta_{p'} \mathrm{n}^{f'}}{\Gamma_{f'}} - \frac{\rho_{p'} \beta_{p'} \mathrm{n}^{f}}{\Gamma_f}. \quad (3.32)$$

[11]When two patients' data allocated to different FSs are swapped, the change in utility is caused by the difference of their transmission and computation latencies, as considered in Eq. (3.32).

Algorithm 3.2: Two Way Swap

Input: Globally accessible $\mathbb{H}$, Information of all patients (as per Algorithm 3.1) and FSs
Output: $\mathbb{H}$

1 **repeat**
2 **for** $f \leftarrow 1$ **to** F **do**
3 **for** *every* $p \in \mathbb{P}^f$ **do**
4 **for** $f' \leftarrow 1$ **to** F **do**
5 **if** $f' == f$ **then**
6 **continue**;
7 **for** *every* $p' \in \mathbb{P}^{f'}$ **do**
8 **if** $\mathrm{n}^{f'} > \mathrm{n}^{max}_{p,f'}$ *or* $\mathrm{n}^{f} > \mathrm{n}^{max}_{p',f}$ **then**
9 **continue**;
10 **if** $\mathscr{H}^{tw}_{diff} > 0$ **then**
11 Swap p and p';
12 Update corresponding values;
13 **until** *No swap increases utility*;

Algorithm 3.3: One Way Swap

Input: Globally accessible $\mathbb{H}$, Information of all patients (as in Algorithm 3.1) and FS
Output: $\mathbb{H}$

1 **repeat**
2 **for** $f \leftarrow 1$ **to** F **do**
3 **for** *every* $p \in \mathbb{P}^f$ **do**
4 **for** $f' \leftarrow 1$ **to** F **do**
5 **if** $f' == f$ **then**
6 **continue**;
7 Compute $\mathscr{H}^{ow}_{diff}$ according to Eq. (3.33);
8 **if** $\mathscr{H}^{ow}_{diff} > 0$ *and* $\mathrm{n}^{f'} + 1 \leq \min_{p' \in \mathbb{P}^{f'} \bigcup \{p\}} (\mathrm{n}^{max}_{p',f'})$ **then**
9 Add p to $\mathbb{P}^{f'}$ and remove p from $\mathbb{P}^f$;
10 Update the values correspondingly;
11 **until** *No swap increases utility*;

At each iteration, the algorithm picks a pair of patients whose data is allocated to different FSs (lines 2-7). Then checks, whether the number of patients' data allocated to those two FSs is greater than the maximum number of patients that can be allocated

to the two FSs (line 8). If it exceeds the maximum number of patients, then the algorithm picks another pair of patients and checks the condition in line 8. Else, it checks whether swapping the allocation of the two patients can increase utility or not (line 10). If utility can be increased, patients are swapped, and updates are taken place in the corresponding values (lines 11-12). This process repeats until no such pair of patients exist (lines 1-13).

3.3.3 One Way Swap based Algorithm

Utility difference due to one way swap, $\mathscr{H}_{diff}^{ow}$[12] is given as:

$$\mathscr{H}_{diff}^{ow} = \frac{\rho_p \eta_p}{r_p^f} - \frac{\rho_p \eta_p}{r_p^{f'}} + \frac{\rho_p \beta_p (\mathrm{n}^f - \mathrm{n}^{f'} - 1)}{\Gamma_f} + \sum_{p' \in \mathbb{P}^f \setminus \{p\}} \frac{\rho_{p'} \beta_{p'}}{\Gamma_f} - \sum_{p' \in \mathbb{P}^{f'}} \frac{\rho_{p'} \beta_{p'}}{\Gamma_{f'}}. \quad (3.33)$$

At each iteration, algorithm picks a patient whose data is allocated to an FS (lines 2-3). It then selects another FS and calculates utility difference if the patient's data is allocated to that FS (lines 4-7). Then, it checks if allocating the patient's data to that FS increases utility (line 8). If utility is increased by satisfying the constraints, the patient's data is allocated to that FS and updates are taken place in the corresponding values (lines 9-10). This process repeats until no such patient exists (lines 1-11).

3.3.4 Patient-FS Allocation Algorithm

The algorithm selects a subset of patients and allocates their data to FSs that satisfy the constraints and maximize the utility. Algorithm 3.4 terminates when there is no improvement in the utility compared to utility obtained in previous iteration. The following Lemma 3.2 provides the utility correlation across different iterations of Algorithm 3.4.

[12]When a patient p's data is reallocated to f' from f, the profit does not change. The change in the cost is calculated as the difference between the transmission times when p's data allocated to f and f'. The computation time of all patients' data allocated to f and f' changes.

Lemma 3.2 *Let $\mathcal{V}^{max}_{diff,i}$ be the $\mathcal{V}^{max}_{diff}$ calculated by the algorithm at i^{th} iteration, then $\mathcal{V}^{max}_{diff,i} \geq \mathcal{V}^{max}_{diff,i+1}$. In other words, the maximum utility difference decreases with each iteration of Algorithm 3.4.*

Proof: Let p and p' be the patients whose data is allocated to FS f at i^{th} iteration and FS f' at $(i+1)^{th}$ iteration, respectively. Then, consider the following two cases:

Case 1: $f = f'$

$$\mathcal{V}^{max}_{diff,i} = \lambda_1(\mathfrak{m} - l - \mathfrak{g}\beta_p) - \lambda_2 \sum_{p'' \in \mathbb{P}^f} \frac{\rho_{p''}\beta_{p''}}{\Gamma_f} + \lambda_2 \left(\rho_p \left(\frac{\beta_p}{\Upsilon} - \left(\frac{\beta_p(\mathrm{n}^f + 1)}{\Gamma_f} + \frac{\eta_p}{r^f_p} \right) \right) \right). \quad (3.34)$$

$$\mathcal{V}^{max}_{diff,i+1} = \lambda_1(\mathfrak{m} - l - \mathfrak{g}\beta_{p'}) - \lambda_2 \sum_{p'' \in \mathbb{P}^f \bigcup p} \frac{\rho_{p''}\beta_{p''}}{\Gamma_f} + \lambda_2 \left(\rho^c_{p'} \left(\frac{\beta_{p'}}{\Upsilon} - \left(\frac{\beta_{p'}(\mathrm{n}^f + 2)}{\Gamma_f} + \frac{\eta_{p'}}{r^f_{p'}} \right) \right) \right), \quad (3.35)$$

where n^f is the number of patients utilizing f before i^{th} iteration and similarly $\mathbb{P}^f$ is the set of such patients. On subtracting Eq. (3.34) from Eq. (3.35), it is clear that,

$$\mathcal{V}^{max}_{diff,i+1} \leq \mathcal{V}^{max}_{diff,i}. \quad (3.36)$$

Case 2: $f \neq f'$

In this case, as both the FSs are different, if $\mathcal{V}^{max}_{diff,i+1}$ would have been greater than $\mathcal{V}^{max}_{diff,i}$, then the algorithm would have picked p' at the i^{th} iteration only, but that is not the case. Hence, $\mathcal{V}^{max}_{diff,i+1} \leq \mathcal{V}^{max}_{diff,i}$.

From both cases, $\mathcal{V}^{max}_{diff,i+1} \leq \mathcal{V}^{max}_{diff,i}$. Hence, proved. □

Algorithm 3.4: Patient-FS Allocation

Input: Globally accessible $\mathbb{H}$, Set of Patients, $\mathbb{P}^{rem}$, $flag = 0$, $temp_p$, $temp_f$, $\mathcal{V}_{diff}^{max} = 0$

Output: Allocation Strategy ($\mathbb{H}$)

```
1  repeat
2      for every p ∈ ℙ^rem do
3          if q_p == 1 then
4              continue;
5          for f ← 1 to F do
6              if n^f ≥ min_{p'∈ℙ^f ⋃{p}} n_{p',f}^max then
7                  continue;
8              Calculate V_diff as per Eq. (3.29);
9              if V_diff > V_diff^max then
10                 flag ← 1;
11                 V_diff^max ← V_diff;
12                 temp_p ← p, temp_f ← f;
13     if flag == 0 then
14         break;
15     Assign patient temp_p to FS temp_f;
16     Update n^{temp_f}, ℙ^rem, and ℍ;
17 until |ℙ^rem| times;
```

3.3.5 Illustration of UMPM Algorithm

Let there be 8 patients and 3 FSs in the system. We consider a constant data size of 2 MB and needed CPU cycles as 600 for each patient. Computation capacity of FSs, F1, F2, and F3 are considered as 22 GHz, 18 GHz, and 20 GHz, respectively. In Fig. 3.5, yellow and blue colors indicate patients and FSs, respectively. The patients' criticalities are shown at the top of the figure. These values are calculated based on simulation Table 3.2 and Eqs. (3.1)-(3.3). Moreover, the costs of patients' data allocations are labeled on respective FSs in Fig. 3.5 after the execution of UMPM algorithm.

Fig. 3.5(a) shows an initial allocation of patients' data to FSs (lines 1-17 of Algorithm 3.1), in which data of patients P5, P7, and P8 is allocated to F1, F3, and F2, respectively. Algorithm 3.4 allocates remaining patients' data to FSs for further improvising the system utility. Thus, we get an outcome as shown in Fig. 3.5(b) after

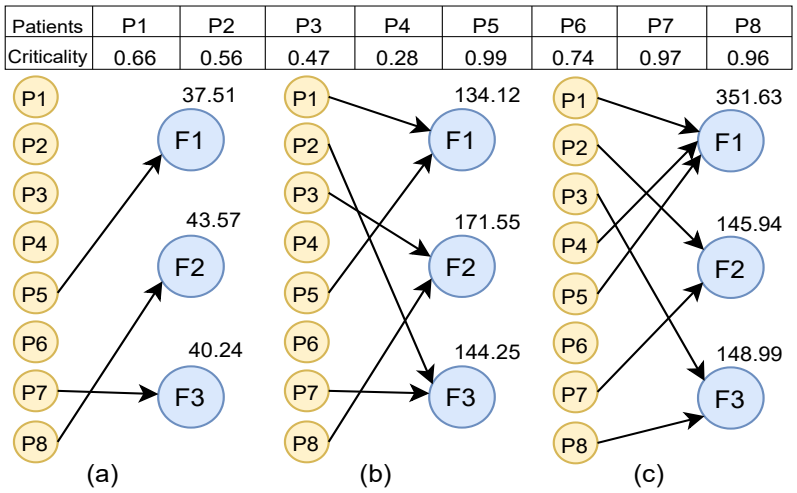


Fig. 3.5. (a) Initial allocation (lines 1-17 of Algorithm 3.1). (b) Allocation after execution of Algorithm 3.4 (lines 18-19 of Algorithm 3.1). (c) Final allocation (lines 20-25 of Algorithm 3.1).

the execution of Algorithm 3.4 (lines 18-19 of Algorithm 3.1). Then, after reiterating Algorithms 3.3, 3.2, and 3.4 (lines 20-25 of Algorithm 3.1), Fig. 3.5(c) is obtained as final allocation with a maximized overall system utility.

3.3.6 Analysis of Proposed Heuristic

This section discusses convergence and time complexity of UMPM algorithm as follows.

Lemma 3.3 *The proposed UMPM algorithm converges.*

Proof: Convergence of UMPM relies on convergence of three sub-algorithms. Algorithms 3.3 and 3.2 swap patients only if utility increases. Else, their execution terminates. As the total possible combinations between patients and FSs are finite, utility will also be a finite value. Thus, both the swap algorithms converge. Algorithm 3.4 converges since the number of iterations is finite, i.e., P. Further, UMPM repeatedly executes two way swap, one way swap, and Patient-FS allocation algorithms. Every iteration converges, and the algorithm goes to next iteration only when utility increases. Therefore, UMPM algorithm converges. □

Theorem 3.2 *Time complexity of UMPM algorithm is* $O(P^2F)$.

Proof: Time complexity of UMPM depends on complexity of three sub-algorithms it calls. Time complexity of Algorithm 3.1 from lines 1-15 is $O(PF)$. Algorithm 3.2 considers P^2 pairs of patients and repeats until it converges. Thus, the number of iterations is bounded by a finite value. Hence, time complexity of Algorithm 3.2 is $O(P^2)$. Similarly, time complexity of Algorithm 3.3 is $O(PF)$ as it considers PF number of possible swaps. In Algorithm 3.4, the number of iterations is bounded by number of patients, i.e., P. In each iteration, PF pairs are considered. Thus, time complexity of Algorithm 3.4 is $O(P^2F)$. Hence, time complexity of proposed UMPM algorithm is $O(P^2 + PF + P^2F)$, i.e., $O(P^2F)$. □

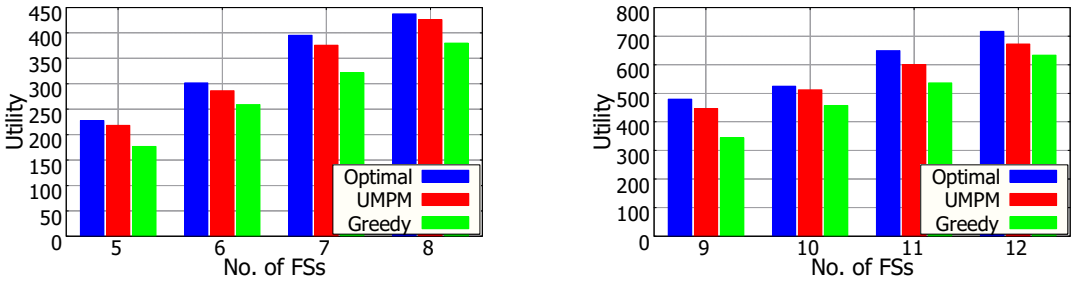
Simulation setup and obtained results are discussed in the following section:

Table 3.2: Simulation parameters

Parameter	Value
P, F	[20-1000], [2-200]
η_p [8]	[1, 3] MB
β_p [8]	[100, 1000] Megacycles
Blood pressure [$\theta_{low,s}$, $\theta_{up,s}$] [89]	91 mmHg, 169 mmHg
Body temperature ($\theta_{low,s}$, $\theta_{up,s}$) [89]	34.1 $^\circ C$, 37.9 $^\circ C$
Heart rate ($\theta_{low,s}$, $\theta_{up,s}$) [89]	51 bpm, 139 bpm
Respiration rate ($\theta_{low,s}$, $\theta_{up,s}$) [89]	11, 29 breath/min
Oxygen saturation ($\theta_{low,s}$, $\theta_{up,s}$) [101]	95 %, 100 %
δ, ρ_p, Υ	250 ms, [0, 1], 2.4 GHz
l, $\mathfrak{m}$, $\mathfrak{g}$, k	100, 200, 0.1, 0 units
ω, Γ_f [8]	[5-15] MHz, [18-22.4] GHz
$SINR_p^f$, V_p^f	[13-20] dB, [5-15]
F_f^{max}, C_{max}	[3-15], [200-1400]
λ_1, λ_2	0.5, 0.5

3.4 Performance Study

Patient's data size and required CPU cycles for computing patient's data are randomly considered between [1, 3] MB and [100, 1000] Megacycles, respectively [8]. The value of δ is taken as 250 ms. The patients' criticalities are considered between [0, 1], as shown



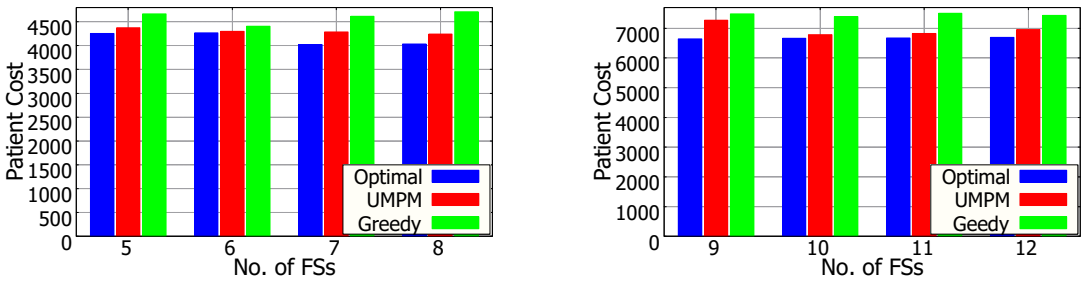
(a) No. of patients = 40 **(b)** No. of patients = 60

Fig. 3.6. Utility comparison among different schemes.

in Table 3.2. Moreover, we have considered 5 physiological sensors' data such as body temperature, heart rate, blood pressure, respiration rate and blood oxygen saturation for evaluation purposes. We used HElib open-source library [102] for implementing homomorphic encryption in our proposed model. Simulation experiments are performed using Windows 10 Home PC equipped with an Intel® Core™ i7-10750H @ 2.60 GHz processor and 16 GB of memory.

We could not compare the proposed model with other existing works because none of the existing works have considered patients' criticality, profit of MC and other constraints altogether as per best of our knowledge (see Table 2.1). Gurobi optimization tool is used to get the optimal solution [103]. Moreover, we offered *Greedy* scheme to compare with the proposed model. In greedy scheme, patients that violate latency constraint are sorted in decreasing order of their criticality. Then, patients' data is allocated to FSs one by one. The remaining patients are sorted in decreasing order of criticality, and the allocation process repeats until no improvement takes place in the utility.

System Utility Analysis: Fig. 3.6 considers two cases where the numbers of patients are 40 and 60, and the number of FSs varies from 5 to 12. We observe from the result that the utility increases as the number of FSs increases. UMPM algorithm performs better than the greedy scheme. The utility obtained by UMPM algorithm is



(a) No. of patients = 40 **(b)** No. of patients = 60

Fig. 3.7. Patients' cost comparison among different schemes.

94.5% of the optimal value compared to 77% that of greedy scheme on an average. The reason is that the greedy scheme allocates patients' data in a particular order. Although the greedy scheme considers patients' criticality, it ignores data size and CPU cycles of health data. UMPM algorithm considers all the above factors to reach a sub-optimal utility within polynomial time complexity. Moreover, the optimal solution performs better than the UMPM algorithm because it considers all possible combinations of allocations and selects the best out of them while maximizing the utility.

Patient Cost Analysis: Fig. 3.7 compares the patients' cost of the UMPM algorithm in various scenarios. We observe that UMPM algorithm generally results in lower patients' cost than that of greedy scheme because UMPM considers different parameters, and it allocates and re-allocates patients' data to maximize the utility by minimizing patients' cost. However, greedy scheme does not re-allocate patients' data, resulting in higher patients' cost and lower utility. We also observe that patients' cost obtained by UMPM algorithm is higher than that of the optimal because the optimal solution considers all possible combinations of allocations and selects the best out of them to maximize the utility.

Impact of λ_1 and λ_2 on Utility: Fig. 3.8 compares the utility of optimal, UMPM, and greedy schemes, taking precision level of 5 decimal points of lambda values. Normal system balances profit and cost factors equally, i.e., $\lambda_1 = \lambda_2 = 0.5$.

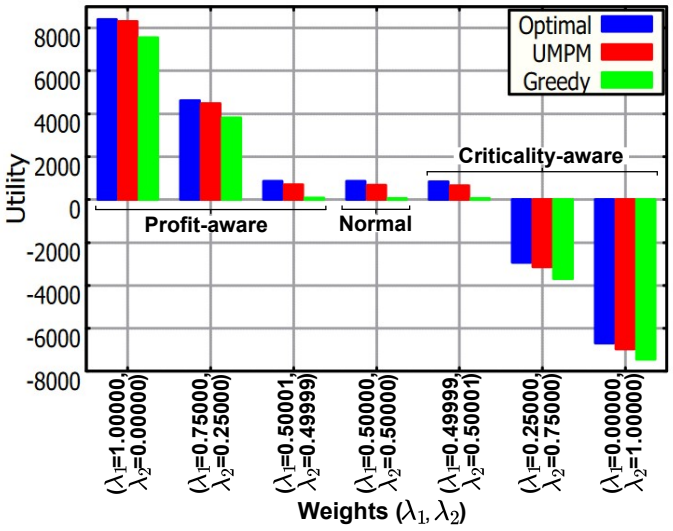
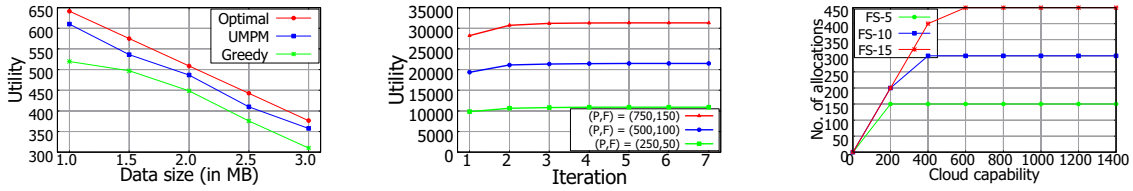


Fig. 3.8. Utility based on λ_1 and λ_2.

Criticality-aware system concerns more about cost factor than profit, i.e., $\lambda_1 < \lambda_2$. Experimental evaluation on precision level of 5 decimal points of lambda values gives highest utility when $\lambda_1 = 0.49999$ and $\lambda_2 = 0.50001$. In fact, the obtained profit and loss factors are the highest and the lowest respectively, when $\lambda_1 = 0.49999$ and $\lambda_2 = 0.50001$ on precision level of 5 decimal points; resulting in a higher utility. Therefore, criticality-aware system reaches its optimal value when λ_2 is closed to 0.5 and $\lambda_1 < \lambda_2$. Moreover, profit-aware system concerns more about the profit factor than the cost, i.e., $\lambda_1 > \lambda_2$. Profit-aware system reaches its optimal value when $\lambda_1 = 1$ and $\lambda_2 = 0$. The reason is that with increase in λ_1 and decrease in λ_2, profit and loss factors become higher and lower, respectively, resulting in a better utility value. Furthermore, UMPM algorithm performs better than greedy in all cases and is closer to optimal value. Therefore, UMPM can be applied efficiently in profit-aware system.

Impact of Data Sizes on Utility: Fig. 3.9a shows the impact of data sizes on system utility. We observe that the system utility decreases as the data size increases. The utility is higher when the patient's data size is small (i.e., 1.0 MB). However, the utility is lower when the patient's data size is large (i.e., 3.0 MB). It is because the required CPU cycles to compute patients' data rise, and transmission time and FS's computation time increase as the data size grows. As a result, the patients' cost increases, resulting in lower utility.

Convergence Analysis: Fig. 3.9b depicts convergence of the UMPM algorithm for three different cases. We observe from the result that the algorithm converges in



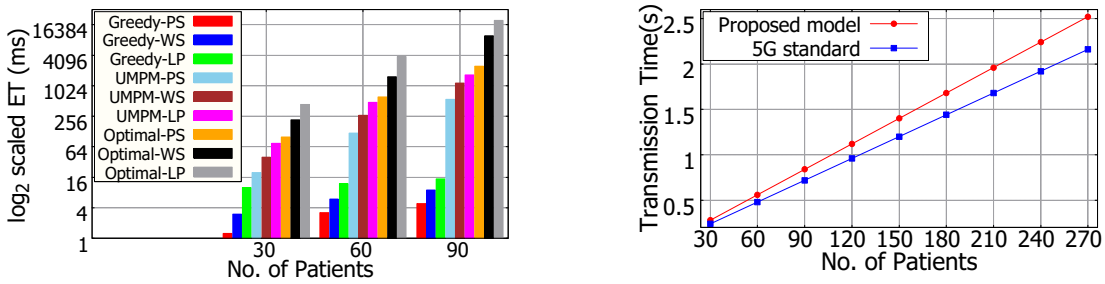
(a) Utility v/s Data sizes **(b)** Convergence analysis **(c)** Patients allocated

Fig. 3.9. Utility v/s data sizes, convergence and allocation analysis.

a few iterations in all three cases. We also observe that the utility increases as the number of patients and FSs increases, as does the number of iterations because the number of possible combinations between patients and FSs increases. However, after certain iterations, the increase in utility is very small and converges in finite iterations for all three cases. Thus, UMPM algorithm converges in finite time.

Patients' Data Allocation Analysis: Fig. 3.9c shows the number of allocations based on the cloud capability ranging from 200 to 1400 patients' requests at a time. We considered 30 FSs, and the number of patients that an FS can handle ranges from 5 to 15. We observe from result that number of allocated patients' data increases when CS's capability rises, but after reaching total capacity of FSs, it becomes constant.

Execution and Transmission Time Analysis: Fig. 3.10a compares the Execution Time (ET) of optimal, UMPM and greedy schemes in various scenarios using Laptop (LP), Workstation (WS), and the Param Shivay (PS) super computer [104]. For optimal, we consider the ET required by the Gurobi optimization tool to run the proposed heuristic. To improve the readability, the y-axis in Fig. 3.10a is given after applying the log_2 scale to ET. We observe that the greedy scheme completes its execution in less time than that of the optimal and the UMPM approaches in all three machines. However, the utility obtained by the greedy scheme is much lower than that of the UMPM algorithm (see Fig. 3.6). Moreover, the optimal solution takes more time than that of the UMPM algorithm since it considers all possible combinations of allocations and selects the best out of them while maximizing the utility. We also



(a) Execution time **(b)** Transmission time

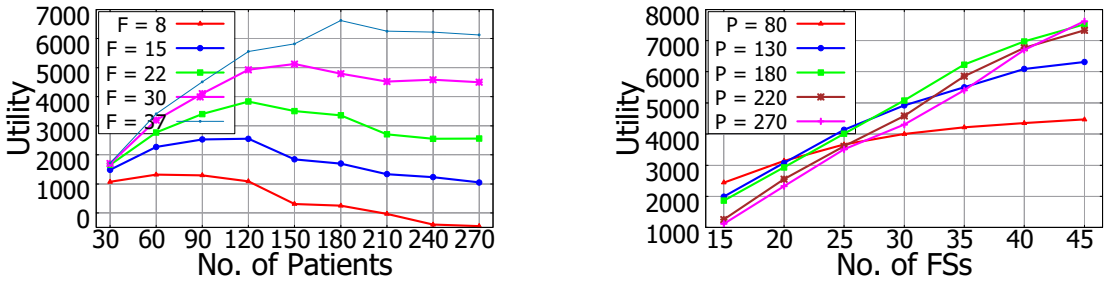
Fig. 3.10. Execution and transmission time analysis.

observe that the ET of the UMPM algorithm is different for different machines. This result shows that the ET depends on machine's configuration, i.e., the configuration of the CS available at MC.

Fig. 3.10b shows the transmission time of the proposed model and the 5G standard. We considered 1 MB data size for each patient and 1 FS to simulate the transmission time. We observe that the transmission time of the proposed model is little higher than that of the 5G standard. The reason for the slight variation is that the proposed model achieves slight less data rate (Eq. (3.9)) between patients and FS than that of the standard method (1 Gbps [105]) due to limited consideration of available PRBs in the system [15].

System Utility using Real World Data: Fig. 3.11 compares the system utility of UMPM algorithm using a real dataset on different settings. We use the Statlog (Heart) dataset [106] which contains 13 attributes and has total of 270 data samples. In our work, we considered three attributes (i.e., blood pressure, ECG and heart rate) as the data collected by sensors from patients. Other simulation parameters are considered the same as given in Table 3.2.

Fig. 3.11a considers five cases where the numbers of FSs are 8, 15, 22, 30, and 37, and the number of patients varies from 30 to 270. We observe that the utility increases when the number of patients increases, but the utility decreases after a certain number



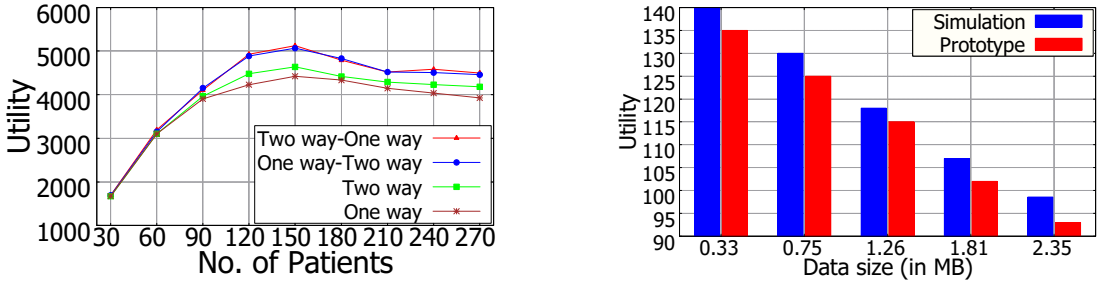
(a) Utility v/s No. of FSs **(b)** Utility v/s No. of patients

Fig. 3.11. Utility comparison of UMPM on various parameters.

of patients. It is because, up to a certain number of patients, the total computation required for the patients is less than the total fog computation power. However, if we increase the number of patients furthermore, the required computation increases but the total fog computation power remains the same. This leads to higher computation time, resulting in higher costs and lower utility. We also observe that the utility is higher for the higher number of FSs. It is because the computation time decreases since total computation power increases as the number of FSs increases, resulting in lower costs.

Fig. 3.11b considers five cases where numbers of patients are 80, 130, 180, 220, and 270, and the number of FSs varies from 15 to 45. We observe from result that the utility increases when the number of FSs increases because the total fog computation power rises, but the required CPU cycles of patients' data remain the same. Thus, FSs can compute patients' data in less computation time. This leads to lower costs, resulting in higher utility. We also observe that the utility is lower for the higher number of patients. The reason is that computation time increases since the number of patients increases, but total computation power remains the same, resulting in higher costs.

Execution Order Analysis: Fig. 3.12a analyses the execution order of swap Algorithms 3.2 and 3.3. We use the Statlog (Heart) dataset as discussed above. From the result, we observe that the utility obtained by performing two way swap followed by



(a) Execution order analysis **(b)** Utility using prototype

Fig. 3.12. Execution order and utility analysis.

one way swap algorithm is almost the same as that of performing one way swap followed by two way swap algorithm. Thus, UMPM algorithm can perform two way and one way swap algorithms in any order. As seen from the result, execution of one way swap or two way swap algorithms one after another increases the overall system utility. We also observe that the utility obtained by performing two way swap algorithm is better than that of performing only one way swap algorithm.

Utility and Encryption Analysis: We use a WS as CS, 2 LPs as FSs, 4 NeuLog sensors, and 4 Raspberry Pi as LDs, as shown in Fig. 3.13. The WS's specification is Core-i7-10700 CPU @ 4.10 GHz processor and 32 GB memory, and that of LPs is same as discussed above. We use Raspberry Pi 4 Model B with 1.5 GHz 64-bit quad-core, ARM Cortex-A72 CPU, and 8GB memory. NeuLog sensor is connected with Raspberry Pi using a USB. Raspberry Pi, LP and WS are connected with a 4G mobile hotspot. NeuLog sensor collects data from the patient and sends it to Raspberry Pi, and Raspberry Pi sends a request to WS for remote health monitoring. The WS allocates patients' data to the LP by executing the UMPM algorithm. Then, Raspberry Pi transmits the health data to the LP for further processing.

UMPM has been evaluated on data sizes varying from 0.33 MB to 2.35 MB (Fig. 3.12b). The result denotes the utility obtained in simulation is higher than that of the prototype model. It is because the prototype model used 4G mobile hotspot for

Fig. 3.13. Prototype setup.

Table 3.3: BGV encryption analysis

Operation	Time (ms)
Encryption	22.99785
Decryption	1.00088

(a) Time

	Data size
Before encryption	84 bytes
After encryption	192 bytes

(b) Data size

transmitting data between Raspberry Pi and LP, unlike the 5G communication used in simulation. Moreover, Raspberry Pi, LPs, and WS use in the prototype model are not dedicated same configured devices as considered in simulation set-up. This leads to higher transmission and computation time, resulting in higher costs and lower utility.

Table 3.3 shows the impact of BGV in terms of data size and encryption/decryption time at the LD. We observed from experiment that encryption and decryption overhead time is very low, as shown in Table 3.3a. As a result, the BGV encryption scheme can be

employed in practice. We also observe an increase in original data size after employing the BGV encryption scheme, as can also be seen in Table 3.3b.

3.5 Summary

In conclusion, this chapter proposed a beyond-WBAN-based fog-assisted remote health monitoring system. A utility maximization problem was formulated, considering the profit of MC and patients' latency costs. Moreover, the UMPM algorithm was presented to maximize overall system utility. Furthermore, this chapter demonstrated through extensive simulations using real-world data and a prototype model that the UMPM algorithm achieves an average utility of 94.5% of the optimal value with polynomial time complexity. However, this chapter did not consider the energy consumption of LDs during health data computation and transmission, a crucial factor that directly affects the lifespan of remote healthcare systems. Thus, there is a need for an efficient, energy-aware resource allocation mechanism that considers factors such as healthcare service provider' profit, patient latency costs, and the prioritization of critical patient health data. Therefore, the next chapter proposes an energy- and latency-aware fog computing-enabled WBAN-based remote healthcare system, considering HSPs' profits and the prioritization of critical patient health data.