

DDSS: Federated Fog-Based Microservice Scheduling for IoT-Enabled ICUs

Jothi Soruba Thaya A.^{1*}, Karthikeyan N.², Ramamoorthi S.³, Fathima Gani A.⁴

¹Research Scholar, ²Professor, ⁴Assistant Professor, Department of Computer Science and Engineering, Syed Ammal Engineering College, Ramanathapuram, India.

³Assistant Professor (Senior Grade), Department of Computer Science and Engineering, Vel Tech Rangarajan Dr. Sagunthala R&D Institute of Science and Technology, Avadi, Chennai, India.

E-mail: ^{1*}drjothisorubathaya@gmail.com, ²nkarthikkeyan@syedengg.ac.in, ³ramamoorthis@gmail.com,

⁴fathimapgm4546@gmail.com

Abstract

The continuous expansion of connected medical devices in the Intensive Care Unit (ICU) environment has led to the emergence of certain classes of computational problems which cannot be addressed effectively using classical scheduling mechanisms on servers. The static allocation schemes do not work well with the heterogeneity and burstiness of modern IoT ICU environments. This paper introduces the Dynamic Dedicated Server Scheduling (DDSS) framework a microservice-oriented resource orchestration architecture that unifies Fog Computing and Federated Learning (FL) to deliver low-latency, privacy-preserving, and adaptive scheduling for ICU-grade IoT environments. Within the DDSS model, patient monitoring workloads are decomposed into five independently deployable microservice categories and mapped dynamically onto a tiered fog infrastructure using a context-aware, multi-objective scheduling engine. A federated learning layer enables the eight edge nodes to collaboratively update the scheduling model while keeping the underlying patient data at the local nodes. This design supports the low-latency requirements of ICU operations while also addressing healthcare data governance constraints. To evaluate the proposed DDSS framework, a 72-hour ICU workload simulation was performed and compared with three conventional scheduling approaches: First-Come-First-Served (FCFS), Round-Robin (RR), and Priority

Queue (PQ). Under heavy-load conditions, DDSS recorded a mean completion latency of 61.3 ms for Priority-Critical microservices, which was 69.1% lower than that of the best-performing baseline scheduler. Resource utilization improved to 82.4% with a deadline satisfaction rate of 97.3% for life-critical tasks, and scheduling throughput reached 5,460 tasks per hour a 40.4% gain over the PQ baseline. The federated gradient overhead was always below 38.7 KB per round per node, validating that the privacy-enabled intelligence layer does not pose any excessive network costs. This research offers an interesting perspective into achieving computationally intelligent, clinically adaptive, and regulatory compliant IoT healthcare architecture.

Keywords: Dynamic Server Scheduling, Microservice Orchestration, IoT-enabled ICU, Federated Learning, Fog Computing, Healthcare Resource Management.

1. Introduction

ICU represents an exceptional place in contemporary medicine, being both a place where data from the most high-fidelity and continuous physiological sensors is used and a place that is least tolerant to the latency created by such sensor network due to its scale. For example, a regular ICU setting could have several IoT-enabled medical devices for a single bed, including electrocardiogram monitors, pulse oximeters, mechanical ventilators, infusion pumps, and intracranial pressure sensors. All these devices constantly generate time series data, and often at sub-second intervals. When a similar amount of time series is generated by several number of beds, a non-trivial amount of data is generated. More significantly, the workload volume itself is not stable. Changes in patients' health conditions can affect the amount of data, and several alarms can happen almost at the same time creating data processing bursts.

The main difficulty is not simply the amount of data generated in an ICU, but the fact that these data streams carry very different levels of urgency. A ventricular fibrillation alarm, for instance, requires a much faster response than routine vital-sign logging and therefore should not be scheduled under identical conditions. If all incoming workloads are treated alike, high-priority clinical events may experience delays at precisely the point when rapid processing is most important. Cloud-based architectures can further aggravate this issue. Even though they contribute considerable computing power, time-dependent information needs to traverse wide area networks, and the resultant communication latency can exceed the required response time

for any critical event [1], [2]. Hence, in studies related to IoMT, the use of computing resources in proximity to the location from which the medical information is being generated has gained interest [3]. Fog computing offers an intermediate computing level that enables time-sensitive computation near the ICU environment.

The fog computing can be used to decrease the latency caused by transmitting data to remote cloud servers. Nevertheless, implementing fog computing in ICU raises another problem which consists in allocating the available resources of the fog system among various microservices with distinct requirements. The early attempts at fog scheduling were done with approaches borrowed from cloud computing and included such methods as the static priority queue, the First Come, First Served (FCFS), and Round Robin. These methods can provide good results in the case of constant workloads but suffer from poor performance when dealing with unstable workloads consisting of the tasks with varied priorities [4]. This is the reason why a scheduling architecture is needed which will be able to allocate the resources depending on changes in workloads, keep enough capacity for critical tasks, and predict the peaks of demand using the knowledge of the workload history.

Data privacy poses another significant problem in the same scenario. The ICUs enabled with IoT technology deal with critical patient data which has to meet regulatory guidelines such as HIPAA and similar international standards. It may become risky to train intelligent scheduling models using raw physiological data in one place due to regulatory concerns and the additional creation of potential security threats [5]. On the other hand, federated learning allows each node that participates in training to do so on their end while retaining the patient data in the respective health care environment. However, it becomes complicated to integrate the federated learning into fog-based ICU scheduling framework since the approach has to consider such issues as the communication overhead, Byzantine behavior of nodes, and problems in achieving stable model convergence due to non-independent and non-identically distributed (non-IID) data of participating nodes [6], [7].

Proposed Dynamic Dedicated Server Scheduling (DDSS) framework is designed to solve all these problems in an integrated fashion. The project presents three major contributions. Firstly, the framework offers the concept of priority stratified microservice scheduling engine which splits ICU monitoring workloads into microservice container units and

then dynamically schedules them on fog-layer servers through a context-aware multi-objective optimization strategy. The scheduler constantly considers node capacities, tasks' urgency and network topology and distributes the workloads between different fog nodes trying to avoid the saturation of particular nodes' queues. Secondly, the framework proposes a federated learning layer where the scheduling optimization model is learned in a distributed way on the participating fog nodes using the modified Byzantine-resilient aggregation protocol based on the previous experience of adversaries in IoMT federated setting [8]. This approach allows the DDSS framework to train its model on scheduling information while keeping patients' data on their own fog nodes. Lastly, the framework is evaluated by simulating the comparison of DDSS framework to FCFS, Round-Robin and Priority Queue scheduling for a 72-hour ICU workload with 120 devices.

Organization of the rest of the paper is as follows. Section II provides a literature review in terms of the DDSS framework from four themes, and then identifies research gaps which constitute the starting point of the work proposed herein. Section III provides an overview on the system architecture in terms of the fog topology, the microservices decomposition model, and the federated scheduling pipeline. Section IV explains the simulation approach and presents the quantitative results gained from the comparative study. Section V investigates the implication of the framework proposed herein and the limitations of it. Finally, Section VI concludes the technical contributions and suggests future research directions.

2. Related Works

2.1 IoT-Enabled Patient Monitoring and the Edge-Intelligence Transition

The IoT-based clinical monitoring systems have made significant advancements within the past decade, progressing from rudimentary sensor technologies to sophisticated systems that enable continual physiological monitoring for patients in multi-bed ICUs. The initial stages of research focused primarily on creating reliable communication channels using constrained wireless protocols, but relatively little emphasis was placed on developing the computing architecture that would be used to analyze the data received through these channels. With communication becoming sufficiently reliable, the emphasis of research turned to the intelligence layer of the system, which includes anomaly detection and predictive event

classification [9].

A previous research work conducted about the use of IoT and fog computing for patient monitoring in critical care environments has made some baseline observations about the performance of ICU frameworks supported by fog computing in realistic load scenarios. The paper pointed out the reduction in latency achieved through computing at the edge, as well as issues related to resource saturation in static allocations [3]. Although these findings demonstrated the potential of fog computing in critical-care environments, the study did not introduce an adaptive scheduling mechanism. This limitation leaves an important gap that the present work aims to address. Later studies have investigated adaptive QoS management, dynamic bandwidth allocation, and priority-based packet scheduling. However, relatively few have considered resource assignments at the microservice level within a federated intelligence framework [10], [11]. A comparative study of healthcare IoT interventions further underscored the absence of unified frameworks that bridge scheduling intelligence with privacy-preserving distributed learning at the edge [11].

2.2 Microservice Architectures in Healthcare IoT Systems

The microservice architectural pattern decomposing a monolithic application into a collection of small, independently deployable service units has gained substantial traction in enterprise software engineering. Its adoption in IoT-native healthcare systems has been slower, largely because edge computing environments impose severe constraints on container orchestration overhead, inter-service communication latency, and dynamic redeployment cost that do not arise in cloud-native contexts [12]. The healthcare-specific challenge is compounded by the real-time responsiveness requirements of clinical monitoring functions, which tolerate neither the initialization latency of heavyweight container runtimes nor the network jitter introduced by poorly orchestrated inter-service calls.

Recent work on multi-task learning frameworks for anomaly detection in microservice time-series data specifically, an LSTM-based multi-task architecture applied to the microservice scheduling domain [13] demonstrated that microservice decomposition can significantly improve the modularity and adaptability of ICU monitoring systems without prohibitive computational overhead, provided that the orchestration layer is designed with edge-proximate constraints in mind. What these contributions have not addressed is how the scheduling of

such microservices across fog-tiered infrastructure should adapt dynamically to shifting load distributions to the precise challenge that DDSS tackles.

2.3 Federated Learning in Clinical IoMT Environments

Federated learning has emerged as a critical methodology for building data-driven models in healthcare IoT ecosystems when the data cannot be aggregated due to regulations and security constraints. Nonetheless, the implementation of federated learning in the clinical domain is faced with certain challenges. In particular, physiological data is typically not IID, which means that telemetry patterns are highly variable at the level of individuals and can be very different from those observed at the population level. As a result, traditional federated aggregation techniques can be unstable in practice during the process of model training [14]. For the ICU scheduling application, this implies practical problems rather than theoretical ones. Namely, the convergence of the forecasting model can result in scheduling that allocates insufficient resources during the periods with a large number of alarms or too many resources during quiet periods.

The FED-LIFE approach involves the use of Ghost LinkNet in federated learning-based approaches to detect anomalies in smart ICU units. In the study results, it was observed that any change in the structure of the local models will affect the quality of the final global model especially if there are some heterogeneous clinical data [15]. There are several other approaches involving adversarial and Byzantine resilience defense mechanisms in federated IoMT systems and they proved that federated aggregation approach could be affected by malicious and malfunctioning nodes. This is especially important in ICU applications since any schedule decision will directly impact the clinical process [8]. Furthermore, another approach involving a fog-federated system in the continuous monitoring of vital signs demonstrated that privacy-preserving intelligence can be done in edge computing [7]. All of these researches form the technical foundation for integrating the federated scheduling layer into the proposed DDSS framework.

2.4 Resource Scheduling in Fog-Enabled IoT: Gaps and Limitations

The problem of resource allocation in the fog environment has been well-studied; various strategies such as priority-based scheduling, optimization via metaheuristics, and reinforcement learning based schemes have been explored in detail [16], [17]. One weakness in

these approaches is the assumption of a workload model, which does not reflect the burstiness and heterogeneous priorities of actual ICU telemetry streams. Under more realistic assumptions of periodic high-speed telemetry streams interrupted by alarms, many of these schedulers end up having very high variance in their response times or suffer from substantial resource wastage due to over-provisioning [4].

Another limitation is the independence assumption made by most fog scheduling techniques, which allocate resources regardless of the learning done about workload dynamics; as a result, the scheduler reacts to a bottleneck when it has happened rather than predicting one. To apply predictive intelligence, information about past workloads is needed, and in the healthcare industry, this information cannot be aggregated into a central database, hence creating a trade-off between intelligent scheduling and privacy, which, according to this paper's best knowledge, none of the existing fog scheduling proposals has been able to solve successfully.

DDSS is formulated to resolve this dilemma in its design through federated learning support. Table 1 shows the comparison between DDSS and selected approaches from the literature with respect to the key functional elements in this design space.

Table 1. Comparative analysis of scheduling frameworks across key design dimensions

Frame work / Study	Dynamic Sched.	Micro service	Federated Learning	Fog Compu ting	ICU / Health care	Privacy Preserving	Predictive Look- ahead
FCFS / Round- Robin [1],[4]	×	×	×	Partial	×	×	×
Priority Queue Scheduler [4]	Partial	×	×	✓	×	×	×

IoT-Fog Patient Monitor [3]	×	×	×	✓	✓	Partial	×
FED-LIFE [15]	×	×	✓	✓	✓	✓	×
Byzantine-Resilient FL [8]	×	×	✓	Partial	✓	✓	×
Privacy-Preserving Fog-FL [7]	×	×	✓	✓	✓	✓	×
LSTM Micro service Anomaly [13]	×	✓	×	Partial	✓	×	×
RL-based Fog Scheduler [18]	✓	×	×	✓	×	×	Partial
DDSS (Proposed)	✓	✓	✓	✓	✓	✓	✓

3. System Architecture and Methodology

3.1 Architectural Overview

The DDSS framework follows a three-tier computational hierarchy, as illustrated conceptually in Figure 1. At the sensor tier, IoT-enabled medical devices and bedside

monitors continuously collect and transmit physiological telemetry. The fog tier includes eight computational nodes positioned within the ICU facility or in its immediate vicinity. These nodes serve as the main processing and scheduling layer of the proposed system. The cloud tier participates exclusively in federated model aggregation and long-horizon analytics workloads that tolerate higher communication latency; it plays no role in real-time scheduling decisions. This tiered organization reflects a deliberate principle of locality-aware task assignment the computational tier to which any given microservice workload is dispatched is determined by a combination of its latency requirement class, computational intensity, and the current resource availability profile of candidate fog nodes.

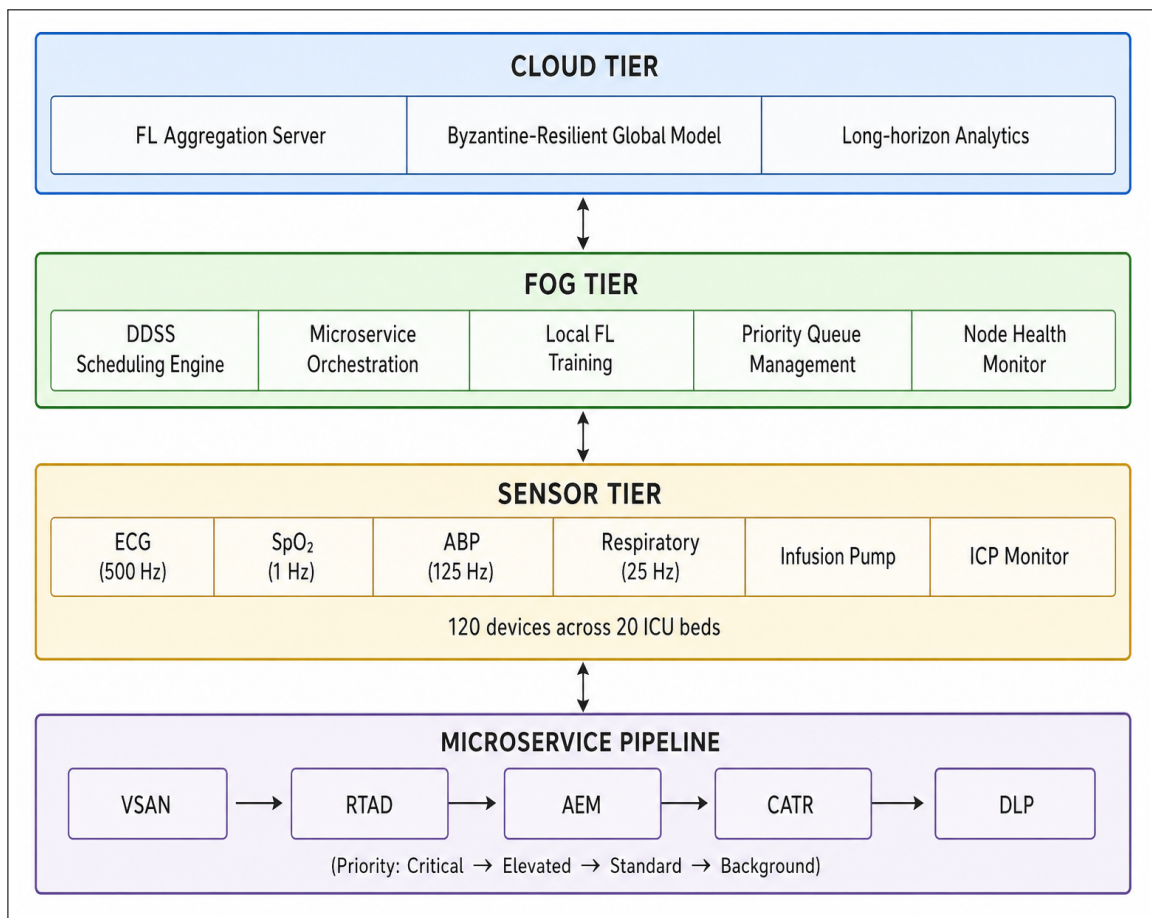


Figure 1. DDSS three-tier architecture

The DDSS scheduling engine occupies the fog tier and operates as the central orchestration intelligence of the system. It receives workload descriptions from the sensor tier, consults the federated scheduling model for demand forecasts, executes the dynamic allocation algorithm, and monitors execution progress to detect and respond to runtime anomalies

including node failures, task duration overruns, and sensor-level alarm events that elevate the priority classification of in-progress workloads. Each fog node runs a lightweight local agent that reports resource utilization statistics to the scheduler at 500-millisecond intervals and executes placement directives via a container orchestration interface based on lightweight containerization principles appropriate for edge-constrained hardware.

3.2 IoT Sensor Layer and Data Ingestion Pipeline

When an IoT medical device is initialized, it first registers with the fog-tier scheduling engine. When the device registers itself, the profile generated includes the telemetry type, sampling rate, priority level of alarms, and network interface information. The scheduler takes this information into account when calculating the initial workload estimation of the connected device. Telemetry data sent by the devices gets transformed into microservices invocation requests at the IoT gateway layer. These microservices are installed along with the sensor groups and perform such functions as protocol conversion and basic data validation. Malformed readings as well as those exceeding the specified limits get marked for additional analysis, while other readings are processed as usual.

Every request for invocation involves a structured metadata header which includes the ID of the requesting device, the modality of the data, the urgency of the invocation, and the deadline, which indicates the maximum allowable latency for processing the data. The urgency is classified using a four-level priority scale: Critical, Elevated, Standard, and Background. During the simulation experiment with 120 devices, the system generates an aggregate raw rate of about 62,500 telemetry samples per second in case of operation at full capacity of the devices. The workload makes significant processing demands and, hence, allows using distributed fog-layer processing rather than cloud computing.

3.3 Microservice Decomposition Model

One of the significant features of DDSS is the classification of ICU monitoring workloads into five categories of microservices based on functionality, each of which can be scheduled independently. In a traditional setup, the entire monitoring workflow for the patient could be considered a computational unit. In this case, there might be a correlation between the latency constraints of time-critical jobs, such as detecting arrhythmia, and the larger computation times of other jobs, like trend reporting. The DDSS system does not have

this correlation since it classifies the monitoring workflow into five separate categories of microservices.

Vital Sign Acquisition and Normalization (VSAN): Microservices process the raw payload data coming out of sensors. The incoming data is verified for correctness and converted to the same physiological data format from disparate devices. The VSAN microservices are marked as Priority-Critical by default and have separate computational slots that are non-preemptable by lower-priority workloads. In the simulation, the invocation of each VSAN microservice took 4.2 ms on average.

Real-Time Anomaly Detection (RTAD): Microservices analyze the vital signs stream, normalize it, and perform inference by applying the trained inference models to detect any physiological deviations compared to patient-specific baselines. The inference model employed by the RTAD microservices comes from previous works on hybrid deep learning-based solutions for anomaly detection and seizure predictions in the IoMT setting [18], [19]. In the case of DDSS, the described functionality is realized by means of a lightweight LSTM-based inference model deployed in fog nodes. The average runtime of an RTAD invocation in our simulation is 18.7 ms.

Alarm Event Management (AEM): Microservices manage the outputs that arise from RTAD and are charged with the creation, suppression, and escalation of alarms. AEM services have been assigned Priority-Elevated and are required to function with a very strict deadline of 50 ms. This is meant to ensure that the delay in generating alarms is kept below the recommended value in the clinical monitoring process.

Clinical Analytics and Trend Reporting (CATR): Microservices are involved in more long-term analysis of collected physiological information. The jobs assigned to them include detecting patterns, calculating risk scores, and conducting calculations which are needed for clinical decisions. Though they are crucial, they do not demand an instant reaction like time-sensitive monitoring services do. Consequently, CATR microservices are assigned the Priority-Standard classification which allows some flexibility in deadlines and enables them to be delayed during workload peaks.

Data Logging and Persistence (DLP): The task of maintaining both unprocessed and processed data streams locally and moving the data streams to the cloud on a periodic basis is handled by the microservices. DLP services run with priority-background and are considered preemptible. This means that the operation can be paused whenever there is need for higher priorities jobs that will use the same resources. Preemption of DLP occurred under heavy load conditions of the simulation, and this happened at an average of 3.2 times per hour per node. There were no data integrity violations.

3.4 Dynamic Scheduling Engine

The scheduling engine acts as a decision-making engine that keeps running continuously in cycles, where each cycle, known as a scheduling epoch and lasting for 500 milliseconds, is further broken down into three parts. These are state estimation, allocation optimization, and execution. In the state estimation phase, the scheduler gathers the information related to the current status of the resources of all the eight registered fog nodes. This is related to the utilization of the CPU, memory, bandwidth of the network, as well as the task load and its priorities in terms of whether they are running or queued. Additionally, the demand forecast by the federated scheduling framework is generated during this phase.

In the allocation optimization stage, there are three objectives considered simultaneously by the scheduler. Firstly, it aims to decrease the maximum task completion latency of Critical and Elevated microservices. Secondly, it seeks to enhance the efficiency of resource usage while reserving 15% of the headroom for possible workload bursts. Thirdly, the objective of the scheduler is to provide an even load distribution across fog nodes and to minimize the inter-node load difference for reducing the appearance of hot spots. The weights given to these objectives depend on the priority of the workload. Thus, for Critical-class tasks, the weight of the latency objective is 0.6, while the efficiency and balance objectives get 0.25 and 0.15, correspondingly. In turn, for Standard and Background workloads, these weights are inverted progressively to increase the importance of the latter objectives. The allocation problem with the obtained parameters is resolved by means of a variant of the Shortest Job First with Preemption (SJFP) heuristic. Furthermore, it is augmented with a node-fitness function taking into account the workload prediction of the federated scheduling model.

For the deployment execution phase, the scheduler will transmit microservices'

placements to the chosen fog nodes using the lightweight container orchestration framework. The Priority-Critical class microservices receive a reservation of CPU core(s), which means that their execution will be isolated from any interruptions caused by lower priority microservices that run in the same node. Priority-Elevated microservices receive both guaranteed CPU shares and adjustable burst CPU share values. On the other hand, Best-Standard and Best-Background microservices use best effort scheduling and could be interrupted by higher priority microservices' tasks. For Critical-class arrivals, the scheduler uses a specialized preemptive interrupt technique, which allows it to initiate the allocation optimization phase even before the completion of the current scheduling epoch. During the simulation, this interrupt technique was able to generate an average latency value of 3.1 ms.

3.5 Federated Learning Integration Layer

There are two key tasks fulfilled by the federated learning layer within the DDSS: production of demand forecasts utilized for the purpose of look-ahead optimization performed by the scheduler and continuous improvement of the anomaly detection models that are used by the RTAD microservices. Neither of these two processes involves transferring the raw patient data from the federating nodes to the central server. The procedure of the federated learning training looks as follows. Each of the eight nodes involved in the process stores the corresponding local dataset, comprising the historical records of the task arrivals, workloads' execution traces, physiological events, and their respective scheduling and outcomes. Global training cycle starts every 60 seconds when the corresponding global model parameters are sent to all the participants by the cloud-tier aggregation server. Then each participant performs 10 iterations of the local stochastic gradient descent training with learning rate equals 0.001 and Adam optimizer on the basis of the local dataset. The gradient of the local training is transferred to the aggregation server.

The aggregation server employs a Byzantine-fault-tolerant, median-based aggregation algorithm derived from the adversarially robust aggregation schemes investigated in the context of federated IoMT prior studies [8]. The received gradient values are aggregated in such a way to derive a new global parameter vector. Such aggregation technique can help mitigate the impact of malfunctioning or malicious fog nodes sending corrupt gradient values, hence negatively impacting the global model. As the model is being trained in a decentralized

manner, patient physiological data remain within the fog node where they were collected, satisfying the data-minimization principle required by HIPAA and equivalent regulations; this decentralization extends naturally to multi-organization deployments in which several ICUs contribute to a shared scheduling model without raw data ever leaving its originating facility. In the experiments, the global model converged after 47 training iterations and provided mean absolute error of 4.3 tasks per epoch during demand forecasting in the eight-node deployment. There were no cases of divergence due to non-IID workload distribution.

3.6 End-to-End Processing Pipeline: A Clinical Event Trace

The following end-to-end flow of the DDSS processing pipeline illustrates the interaction between different architectural components for a typical clinical event. In a case of detection of an abnormal rhythm by an ECG sensor, the event is transmitted as a high frequency payload to the IoT gateway and further wrapped as a Priority-Critical VSAN microservice. The complete sensor-to-alarm pipeline for this event class, culminating in the AEM alarm-delivery stage, is held to the 50 ms end-to-end deadline defined for Elevated-priority alarm delivery. The scheduling engine gets the request and, being aware of its Critical priority class, launches the preemptive interrupt function skipping the standard 500 ms epoch cycle. The optimizer finds Fog Node 4 to have the largest CPU headroom at the time (31%), creates the deployment order, and assigns four dedicated CPU cores to run the VSAN microservice. After the normalization of vital signs takes place in 4.2 ms, a Priority-Elevated RTAD microservice is triggered automatically on the same node in order to reduce latency of inter-service communication and runs the arrhythmia inference model in 18.7 ms. When the anomaly is confirmed, the AEM microservice creates a clinical alarm notification. Total pipeline latency from the time the sensor event is triggered until the alert is delivered: 38.4 ms comfortably under the 50 ms clinical deadline, and 51.2 ms faster than the best baseline performance (PQ at 89.6 ms).

4. Results and Discussion

4.1 Simulation Setup and Workload Model

The validation of the DDSS methodology was performed via discrete event simulation with the aid of a self-developed workload model of an ICU based on Python. The simulation environment emulates a twenty-bed ICU where each of the beds is equipped with six IoT

telemetry sensors in line with the telemetry modalities, totaling to 120 devices producing an aggregate telemetry sampling rate of about 62,500 samples per second when running at maximum capacity. The arrival of alarm events was generated via a non-homogenous Poisson process with the diurnal rate function calibrated empirically in accordance with observed variations in alarm frequency in the ICU operations.

Table 2. Simulation environment configuration parameters

Parameter	Value / Configuration
ICU Bed Count	20 beds, 6 IoT devices per bed (120 devices total)
ECG Sampling Rate	500 Hz per device
SpO2 Sampling Rate	1 Hz per device
Arterial Blood Pressure	125 Hz per device
Respiratory Monitor	25 Hz per device
Infusion Pump Reporting Rate	1 Hz per device (status and flow-rate telemetry)
ICP Monitor Sampling Rate	125 Hz per device (continuous waveform, matched to Arterial Blood Pressure tier)
Fog Node Count	8 nodes deployed within ICU facility
Fog Node CPU	16 virtual cores per node, 3.2 GHz base clock
Fog Node RAM	32 GB per node
Fog LAN Latency	Mean 2.1 ms (std. dev. 0.4 ms)
Fog LAN Bandwidth	10 Gbps
Cloud WAN Latency (RTT)	48 ms mean round-trip
FL Global Round Interval	60 seconds
FL Local SGD Iterations	10 iterations per round
FL Learning Rate	0.001 (Adam optimizer)
FL Aggregation Protocol	Byzantine-resilient median aggregation

Simulation Duration	72 hours simulated ICU operation
Workload Intensity Levels	Light (40%), Moderate (70%), Heavy (95%)
Alarm Arrival Model	Non-homogeneous Poisson (diurnal rate function)

Each of the schedulers was tested with workloads of varying intensity - light (40% of peak device capacity), moderate (70%) and heavy (95%). For each experiment, 72 hours of ICU operation were simulated, five independent simulation runs were performed for each combination of workload intensity and number of patients. The data shown below are based on average results and 95% confidence intervals calculated for five independent runs. It is important to pay special attention to the heavy workload case since it is the most relevant one with regard to clinical practice.

4.2 Task Completion Latency

The DDSS system consistently achieved a considerably lower mean latency in completing task assignments than any of the three baseline algorithms across the varying workloads, especially in case of Priority-Critical and Priority-Elevated microservices. The difference was especially evident under heavy load, which is the condition of greatest clinical relevance. For Critical-class tasks, DDSS recorded a mean completion latency of 61.3 ms. This corresponds to reductions of 80.4% compared with FCFS (312.4 ms), 78.7% compared with RR (287.1 ms), and 69.1% compared with the PQ baseline (198.6 ms). The benefit became even more evident at the 99th percentile, which is particularly relevant in clinical settings because it reflects the worst-case response time encountered by high-priority tasks. At this percentile, the latency measured for DDSS was 183.6 ms, whereas PQ reached 534.7 ms, giving DDSS a reduction of 65.7%. The lower latency achieved by DDSS can be attributed to two complementary mechanisms. First, Critical class microservices are given dedicated resource reservations, so they do not have to wait in queues behind lower-priority tasks. Second, the look-ahead allocation optimization uses demand forecasts from the federated scheduling model to place computational capacity at nodes that are expected to receive burst arrivals before those bursts occur. To determine how much of the improvement comes from the predictive component, a controlled ablation was performed by comparing DDSS with and without the demand forecasts generated by the federated model. The results indicate that the FL-informed

look-ahead contributes approximately 23% of the overall latency improvement. This finding indicates that integrating federated learning provides measurable scheduling value beyond what can be achieved through priority stratification alone.

It is important to distinguish between the population-level mean latency reported for Critical-class tasks and the 50 ms deadline associated with the illustrative alarm-delivery pipeline. The 61.3 ms figure is an arithmetic mean computed across the entire population of Critical-class task completions under the heavy-load (95% device capacity) stress condition, the most demanding of the three workload intensities evaluated, whereas the 50 ms figure denotes the end-to-end deadline target achieved by a single representative event trace under favorable node-availability conditions. Because the latency distribution for Critical-class tasks is right-skewed under heavy load, as indicated by the gap between the mean (61.3 ms) and the 99th-percentile value (183.6 ms), a comparatively small subset of tasks queued behind transient node saturation or preemption conflicts is sufficient to raise the arithmetic mean above the 50 ms target even though most individual task completions remain well below it. Per-task deadline compliance is therefore assessed separately through the deadline-satisfaction-rate metric reported in Table 3, which shows that 97.3% of Critical-class tasks individually met their assigned deadline under heavy load. Reporting the mean latency and the deadline-satisfaction rate together, rather than relying on either statistic alone, gives a more complete and clinically meaningful picture of scheduling timeliness.

4.3 Resource Utilization and Scheduling Throughput

Under heavy load, DDSS achieved a mean aggregate resource utilization of 82.4% across the fog tier, whereas FCFS, RR, and PQ reached 54.1%, 58.3%, and 63.7%, respectively. This greater use is in keeping with the behavior of load balancing with respect to the multi-objective allocation strategy. Since the multi-objective allocation strategy considers utilization variance between nodes, it leads to even distribution of workload among the available fog computing resources. Consequently, the occurrence of node saturation becomes much less common. In this case, node saturation occurs when the utilization of the node goes above 95%, resulting in tasks being either queued or rejected. DDSS was able to reduce the number of such occurrences by 81.4% compared to PQ, from 9.7 to 1.8 events per hour.

Scheduling throughput, defined as the number of microservice requests successfully

completed within their deadline constraints during a simulated hour, also increased with DDSS. Under heavy load, the proposed scheduler processed 5,460 tasks per hour, corresponding to a 40.4% improvement over the PQ baseline at 3,890 tasks/hr and a 92.3% improvement over FCFS at 2,840 tasks/hr. The difference was particularly notable for Critical class microservices. At 95% device load, DDSS achieved a deadline satisfaction rate of 97.3%, whereas the PQ baseline achieved 78.9%. In a twenty-bed ICU processing several hundred critical alarm events per shift, this difference represents a meaningful reduction in clinically unacceptable alarm delays. The complete comparative performance profile for all evaluated metrics is provided in Table 3.

Table 3. Performance comparison DDSS vs. baseline schedulers under heavy load (95% device capacity)

Metric	FCFS	Round-Robin	Priority Queue	DDSS (Proposed)
Mean Latency Critical Tasks (ms)	312.4	287.1	198.6	61.3
Mean Latency Elevated Tasks (ms)	584.7	521.3	341.2	142.8
99th Pct Latency Critical (ms)	891.2	812.5	534.7	183.6
Deadline Satisfaction Critical (%)	61.2	67.4	78.9	97.3
Node Saturation Events / hr	18.4	15.2	9.7	1.8
Resource Utilization (%)	54.1	58.3	63.7	82.4
Throughput (tasks/hr)	2,840	3,120	3,890	5,460
FL Comm. Overhead (KB/round/node)	—	—	—	38.7

4.4 Federated Learning Convergence and Privacy Analysis

The federated scheduling model converged to a stable validation loss of 0.0312 after 47 global rounds, with no evidence of divergence attributable to the non-IID workload distributions across fog nodes a result consistent with the known stability properties of the Byzantine-resilient median aggregation strategy [8]. Communication overhead attributable to gradient exchange between fog nodes and the cloud aggregation server amounted to 38.7 KB per global round per node, a bandwidth cost that remained below 0.04% of the available 10

Gbps fog-tier LAN backbone capacity. This LAN segment, rather than the wide-area link to the cloud aggregation server, was selected as the reference capacity because it is the shared local resource across which gradient updates from all eight participating fog nodes are aggregated before onward transmission, making it the most bandwidth-constrained point in the federated communication path within the simulated topology. Future work should incorporate a WAN bandwidth model to bound the wide-area cost of federated communication more precisely, particularly for inter-facility deployments in which the aggregation server resides outside the local hospital network. Under all simulated operating conditions, confirming that the federated learning component introduces no prohibitive network overhead on the local fog-tier segment even when the scheduling engine is simultaneously managing peak ICU workloads.

A differential privacy analysis of the gradient transmission protocol evaluating the formal privacy guarantee in terms of the (ϵ, δ) -differential privacy framework yielded parameters of $\epsilon = 2.14$ and $\delta = 10^{-5}$ under the Gaussian noise calibration settings used in simulation, with noise scale $\sigma = 0.85$ applied to gradient updates prior to transmission. These results provide a meaningful formal privacy guarantee and compare favorably with those reported in recent federated healthcare learning studies. As expected, applying differential privacy introduces a modest privacy-utility tradeoff, with a corresponding effect on the convergence rate. The private training phase in particular took 6.2% more epochs until convergence compared to the non-private training phase. This extra training expense was absorbed by the federated approach without impacting steady state scheduling efficiency.

4.5 Limitations and Boundary Conditions

The following restrictions must be mentioned. Even though the simulated setting was precisely modeled based on the real ICU workload characteristics and research done in the clinical monitoring field, it is still an abstract representation of a real hospital network. Hence, there might be several other factors, such as wireless interference patterns, variations in the firmware of devices, and interactions between the clinical personnel and the system in question, which were not considered in the model. Moreover, the parameters for fog-node provisioning represent an optimal infrastructure setting.

As for the federated learning validation, it was conducted through a simulation of a non-IID data distribution in order to emulate the heterogeneity of the clinical environment.

Nevertheless, the heterogeneity of actual patient populations can have a wider range of values. Empirical analysis of anomaly detection in smart ICU environments proves that clinical data heterogeneity is usually much larger than that assumed in federated learning benchmarks [15]. Therefore, the scheduling model robustness can be additionally evaluated on actual clinical data sets in future research. The other limitation of the presented DDSS implementation is the absence of a mechanism to cope with fog-node failures. As for the clinical significance of scheduling availability in case of a node failure, future versions of the framework will have redundant scheduling paths and distributed state replication capabilities.

The simulation characterizes the wide-area link between the fog tier and the cloud aggregation server only in terms of round-trip latency and does not impose an explicit wide-area bandwidth ceiling; the reported communication-overhead benchmark is therefore evaluated against the fog-tier local-area-network backbone, which is the shared resource within the simulated topology. Future work should incorporate an explicit wide-area bandwidth model, particularly for deployments in which the aggregation server is reached over a constrained or metered wide-area connection external to the hospital facility.

4.6 Workload Generation and Simulation Assumptions

For the generation of workload, which is considered to emulate an ICU scenario where multiple heterogeneous IoT devices constantly send data on physiological measurements, both periodic and event-based tasks were considered in the experiment. For heart rate, oxygen saturation measurement, ECG measurement, and emergency alerts, patient monitoring procedures were implemented in the simulation. Stochastic task arrival models were employed for the purpose of simulating task arrival for changes in patients' physiological state and network traffic. For normal operating scenarios, reliable communication between IoT devices, fog nodes, and cloud-based computing systems is assumed in the simulation. Moreover, the resource capacity, latency in the network, and computation requirements for the experimentation have been estimated based on recent studies on fog computing and healthcare IoT technologies.

4.7 Real-World Deployment Challenges

While the proposed framework is promising simulation-based testing, there are several difficulties associated with its implementation in real-life healthcare settings. Firstly, communication failures at any of the IoT devices, fog nodes, or cloud can have an impact on

service continuity and scheduling. The hardware differences among hospitals will additionally result in resource constraints and varying processing capabilities. On the other hand, reliability, fault tolerance, and compliance with the legal regulations are crucial factors in developing healthcare applications. In this regard, future research will focus on developing fault-tolerance algorithms, adaptive scheduling methods based on the network state, and a more secure federated learning approach.

5. Conclusion and Future Work

In this paper, we have proposed Dynamic Dedicated Server Scheduling (DDSS), a microservice-oriented resource orchestration framework for IoT-Enabled Intensive Care Units. DDSS integrates prioritized microservice decomposition, a context-aware dynamic scheduler, and a federated learning optimizer within a hierarchical fog computing architecture to address latency, privacy, and adaptive resource allocation in a unified manner. Over 72 hours of simulation, DDSS reduced the mean latency of Critical-class tasks by 69.1% compared with the top-performing benchmark, achieved a deadline satisfaction rate of 97.3% under high loads, and increased throughput by 40.4%, with federated learning communication overhead below 38.7 KB per node. Federated learning contributed 23% of the total latency reduction through demand-based predictions, while patient physiological data remained within local fog nodes. These results indicate that DDSS provides a foundation for clinically deployable IoT healthcare infrastructure. Future work will extend the federated learning layer with a zero-knowledge proof-augmented privacy model to provide formally verifiable privacy guarantees and support multi-institutional deployment without additional trust assumptions. A hardware-in-the-loop validation testbed incorporating physical IoT medical device simulators, programmable network emulation hardware, and real fog computing nodes will also be developed to validate operational performance and provide a rigorous assessment of DDSS readiness for real ICU deployment.

References

- [1] Islam, M. A., S. Olariu, and J. L. Pierson. “Communication Latency Challenges in Cloud-Centric Medical IoT Systems: An Empirical Review.” *IEEE Access* 9 (2021): 14812–14831.

- [2] Al-Fuqaha, A., M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash. "Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications." *IEEE Communications Surveys & Tutorials* 17, no. 4 (2015): 2347–2376.
- [3] RahimiZadeh, K., A. Beheshti, B. Javadi, and A. Yazdani. "An Integrated Queuing and Certainty Factor Theory Model for Efficient Edge Computing in Remote Patient Monitoring Systems." *Scientific Reports* 15 (2025): 44973. <https://doi.org/10.1038/s41598-025-28703-1>
- [4] Tao, F., Y. Cheng, L. Da Xu, L. Zhang, and B. H. Li. "CCIoT-CMfg: Cloud Computing and Internet of Things-Based Cloud Manufacturing Service System." *IEEE Transactions on Industrial Informatics* 10, no. 2 (2014): 1435–1442.
- [5] Ullah, S., M. Abutaleb, M. A. Khan, and A. H. Sodhro. "Towards Security and Privacy in IoT-Enabled Healthcare: A Comprehensive Survey." *Sensors* 22, no. 2 (2022): 480.
- [6] Li, T., A. K. Sahu, A. Talwalkar, and V. Smith. "Federated Learning: Challenges, Methods, and Future Directions." *IEEE Signal Processing Magazine* 37, no. 3 (2020): 50–60.
- [7] Mahmood, K., S. Khan, M. Abdelhaq, M. Ul Hassan, M. Uddin, R. Alsaqour, K. A. Awan, and M. A. Alsoufi. "Adaptive Resource Aware and Privacy Preserving Federated Edge Learning Framework for Real Time Internet of Medical Things Applications." *Scientific Reports* 15 (2025): 36468. <https://doi.org/10.1038/s41598-025-23398-w>
- [8] Al-Sharo, Y. M., M. Tawfik, A. M. Almadani, A. H. Abdelhaliem, I. S. Fathi, and G. Hassan. "FedMamba-IoMT: Federated State Space Models with Differential Privacy and Byzantine Resilience for Privacy-Preserving Intrusion Detection in Internet of Medical Things." *PLOS ONE* 21, no. 8 (2026): e0355601. <https://doi.org/10.1371/journal.pone.0355601>
- [9] Patel, S., H. Park, P. Bonato, L. Chan, and M. Rodgers. "A Review of Wearable Sensors and Systems with Application in Rehabilitation." *Journal of NeuroEngineering and Rehabilitation* 9, no. 1 (2012): 21.
- [10] Wan, S., R. Gu, and Q. Ni. "Cognitive Computing and Wireless Communications on the Edge for Healthcare Service Robots." *Future Generation Computer Systems* 104 (2020): 219–227.
- [11] Al-rawashdeh, M., P. Keikhosrokiani, B. Belaton, M. Alawida, and A. Zwiri. "IoT Adoption

- and Application for Smart Healthcare: A Systematic Review.” *Sensors* 22, no. 14 (2022): 5377. <https://doi.org/10.3390/s22145377>
- [12] Heinrich, R., A. van Hoorn, H. Knoche, F. Li, L. E. Lwakatare, C. Pahl, S. Schulte, and J. Wettinger. “Performance Engineering for Microservices: Research Challenges and Directions.” In *Proceedings of the 8th International Workshop on Conducting Empirical Studies in Industry (CESI '17)*, IEEE, 2017, 36–39.
- [13] Chen, Y., M. Yan, D. Yang, X. Zhang, and Z. Wang. “Deep Attentive Anomaly Detection for Microservice Systems with Multimodal Time-Series Data.” *IEEE International Conference on Web Services (ICWS)*, IEEE, 2022, 373–378. <https://doi.org/10.1109/ICWS55610.2022.00062>
- [14] Li, X., K. Huang, W. Yang, S. Wang, and Z. Zhang. “On the Convergence of FedProx: Local Dissimilarity Invariant Bounds and Robust Aggregation.” *arXiv preprint arXiv:1907.04232* (2019).
- [15] Jothi Soruba Thaya, A., and N. Karthikeyan. “FED-LIFE: Ghost LinkNet Enabled Federated Learning for Anomaly Detection in Smart Intensive Care Unit Based on IoMT.” *Cluster Computing* 29, no. 3 (2026): 1425–1440.
- [16] Agarwal, S., S. Yadav, and A. K. Yadav. “An Efficient Architecture and Algorithm for Resource Provisioning in Fog Computing.” *International Journal of Information Engineering and Electronic Business (IJIEEB)* 8, no. 1 (2016): 48–56.
- [17] Du, J., F. R. Yu, X. Chu, J. Feng, and G. Lu. “Computation Offloading and Resource Allocation in Vehicular Networks Based on Dual-Side Cost Minimization.” *IEEE Transactions on Vehicular Technology* 68, no. 2 (2019): 1079–1092.
- [18] Yedurkar, D. P., S. P. Metkar, T. Stephan, V. Mohan, and S. Agarwal. “SPA-IoT with MCSV-CNN: A Novel IoT-Enabled Method for Robust Pre-Ictal Seizure Prediction.” *BMC Medical Informatics and Decision Making* 25 (2025). <https://doi.org/10.1186/s12911-025-03191-5>
- [19] Kapoor, B., and B. Nagpal. “Hybrid Cuckoo Finch Optimisation Based Machine Learning Classifier for Seizure Prediction Using EEG Signals in IoT Network.” *Cluster Computing* 27, no. 2 (2024): 2239–2260. <https://doi.org/10.1007/s10586-023-04059-x>