



OPEN A secured remote patient monitoring framework for IoMT ecosystems

Pronaya Bhattacharya¹, Anwesha Mukherjee², Bharat Bhushan³, Shashi Kant Gupta⁴,
Thippa Reddy Gadekallu^{5,6,7✉} & Zhu Zhu^{8✉}

Recent advancement in the Internet of Medical Things (IoMT) allows patients to set up smart sensors and medical devices to connect to remote healthcare setups. However, existing remote patient monitoring solutions predominantly rely on persistent connectivity and centralized cloud processing, resulting in high latency and energy consumption, particularly in environments with intermittent network availability. There is a need for real-time IoMT computing closer to the dew, with secured and privacy-enabled access to healthcare data. To address this, we propose the *DeW-IoMT* framework, which includes a dew layer in the roof-fog-cloud systems. Notably, our approach introduces a novel roof computing layer that acts as an intermediary gateway between the dew and fog layers, enhancing data security and reducing communication latency. The proposed architecture provides critical services during disconnected operations and minimizes computational requirements for the fog-cloud system. We measure heart rate using the pulse sensor, where the dew layer sets up conditions for remote patient monitoring with low overheads. We experimentally analyze the proposed scheme's response time, energy dissipation, and bandwidth and present a simulation analysis of the fog layer through the iFogSim software. Our results at dew demonstrate a reduction in response time by 74.61%, a decrease in energy consumption by 38.78%, and a 33.56% reduction in task data compared to traditional cloud-centric models. Our findings validate the framework viability in scalable IoMT setups.

Keywords Dew computing, Cloud computing, Sensors, Internet of medical things, Healthcare 5.0

The Internet of Medical Things (IoMT) connects wearable and implantable sensors to gateways that interface with cloud platforms (e.g., AWS, Microsoft Azure, Blynk Cloud) for data storage and analytics. This ecosystem leverages technologies such as AI, Big Data, Blockchain, and advanced wireless networks (e.g., 5G and beyond)^{1,2}. According to Acumen Research, IoMT sensor market revenue is projected to grow from 6,193 million USD in 2022 to 11,973.24 million USD by 2030 at a CAGR of 7.6%³. Healthcare is set to evolve into Healthcare 5.0-encompassing telehealth, telesurgery, and virtual clinics enhanced by augmented and virtual reality^{4–6}—and will further benefit from distributed and explainable AI, blockchain, 6G, AR/VR, and the metaverse to improve real-time patient monitoring and telepresence⁷. Ultimately, this will create an interconnected healthcare system that leverages resources intelligently to enhance the quality of care and improve patient outcomes⁸.

Despite these advances, current IoMT assisted remote patient monitoring solutions predominantly rely on centralized cloud-based processing, which suffers from high latency, increased energy consumption, and vulnerability under intermittent connectivity. This limitation motivates the need for a more decentralized approach that brings computation closer to the data source. IoMT in healthcare 5.0 requires resilient cloud-based architectures. However, owing to computational bottlenecks and high-end latency on the cloud, research has moved towards the edge and fog-based analytics⁹. The demand for processing elements is made close to

¹Department of Computer Science and Engineering, Amity School of Engineering and Technology, and Research and Innovation Cell, Amity University, Kolkata, West Bengal, India. ²Department of Computer Science, Mahishadal Raj College, Mahishadal, Purba Medinipur, West Bengal, India. ³Computer Science and Engineering Department, School of Engineering and Technology, Sharda University, Greater Noida, India. ⁴Centre for Research Impact & Outcome, Chitkara University Institute of Engineering and Technology, Chitkara University, Rajpura, Punjab, India. ⁵College of Mechanical Engineering (College of Robotics Engineering), Jiaxing University, Jiaxing 314001, China. ⁶Division of Research and Development, Lovely Professional University, Phagwara, Punjab, India. ⁷Centre for Research Impact & Outcome, Chitkara University Institute of Engineering and Technology, Chitkara University, Rajpura, Punjab, India. ⁸School of Haiyan, Jiaxing Vocational and Technical College, Jiaxing City 314036, Zhejiang Province, China. ✉email: thippareddy@ieee.org; zzzhupearl@gmail.com

the sensor nodes. Thus, a hierarchical architecture is formed, where the fog layer connects networking devices (gateways and switches) for analytics. Fog is the resource manager between the cloud and edge computing node, closest to the physical sensor node¹⁰. However, the COVID-19 pandemic has introduced requirements for quick, seamless, and ultra-responsive health monitoring and decision frameworks for critical patients¹¹. In IoMT networks, the fog/edge nodes face bottlenecks owing to low power, limited storage, and resource constraints. Thus, there is a general requirement to make quick decisions in state-of-network disconnections at cloud/fog nodes so that timely help can be supplied in emergencies. Thus, Dew Computing (DC) forms a viable solution to assist IoMT with the support of fog/cloud computing analytics¹².

DC allows the computing paradigm to shift towards the end device, where tightly-coupled microservice operations are supported to give quick decisions with extremely low latency. With DC, Roof Computing (RC) layer can be integrated to provide an end-to-end resource and security support to the IoMT ecosystem^{13,14}. RC layer acts as a gateway between the dew and the fog layer and ensures quick data processing and collection from dew client nodes. Further, security principles can be integrated at the RC layer, which provides secured and privacy-preserving access to the IoMT data¹⁵. This is crucial as medical data contains sensitive attributes, and its disclosure and alteration might create life-threatening effects. RC layer also minimizes the amount of data transmission and improves the infrastructure build cost of the Cloud Computing (CC) nodes. The RC layer can be built as a service (software) on top of the DC layer or as a hardware device (access point or gateway) node. The RC layer allows data aggregation from multiple dew devices, and specific user queries are forwarded to the fog layer if not resolved at the RC layer itself. A similar hierarchy is followed for the fog-cloud system. Thus, the DC-RC link forms a scalable near-fog-cloud ecosystem to support resilient and scalable operations at low latency in IoMT setups. Furthermore, the RC layer serves as a critical intermediary by forwarding the preprocessed data and local intelligence gathered at the dew layer to the fog layer, where more complex analytics are performed. The fog layer, with its enhanced computational resources, refines this data to detect anomalies and execute decision-making tasks that exceed the processing capabilities of the roof. Subsequently, essential insights and aggregated data are transmitted to the cloud, where heavyweight models provide long-term analysis and storage. This integrated flow across the RC, fog, and cloud layers ensures low latency, enhanced security, and overall system resilience in the IoMT ecosystem.

The choice of computing mechanism depends on the application requirements and constraints. Table 1 presents a comparative analysis of different computing paradigms in real-world scenarios. The far-edge (dew-fog-cloud) computing system is suitable for applications that require low latency and moderate bandwidth, such as real-time monitoring and control systems. The near-edge (dew-roof-fog-cloud) computing system is suitable for applications that require higher bandwidth and low to moderate latency, such as healthcare, IoT, and federated analytics. The dew-edge-cloud computing system is suitable for applications that require low latency and high bandwidth, such as cloud gaming and high-performance computing.

For IoMT applications, the dew-roof-fog-cloud computing system is the best fit. IoMT applications require low latency and high bandwidth for real-time data processing and analysis, which the dew and roof layers can provide. Additionally, security and resiliency are crucial for IoMT applications, and these requirements can be met by the fog and cloud layers, respectively. Furthermore, energy consumption is a critical consideration for IoMT devices. The dew-roof-fog-cloud computing system can minimize energy consumption by processing and analyzing data locally at the dew and roof layers before transmitting it to the fog and cloud layers for further analysis and storage. Therefore, the dew-roof-fog-cloud computing system is the best choice for IoMT applications, providing the optimal balance of energy efficiency, security, and resiliency.

Thus, the DC layer hierarchically communicates with the CC layer via the roof-fog layer. The CC-RC-DC operation opens up exciting avenues for distributed data accessibility at the local device, with no Internet availability¹⁶. Dropbox support is an example of a DC layer, where a user carries out offline operations and is transferred to the CC server during reconnection. Thus, DC extends the reach of cloud servers through an independent and hybrid computing behavior, which is self-adaptive and extremely scalable, even in the presence of heterogeneous offline IoMT nodes. In the case of IoMT, the DC layer can be installed on lightweight Arduino and Raspberry Pi devices, where critical indicators are monitored offline, and remote monitoring and analytics are carried out at the CC layer. However, owing to the federated nature of the DC layer, dew-intelligence (or AI analytics at the dew)^{17,18} requires an adaptation towards data sharing challenges and regulations. In such cases,

Parameters	Far-Edge (dew-fog-cloud)	Near-Edge (dew-roof-fog-cloud)	Dew-Edge-Cloud
Latency	10–20 ms	1–5 ms	< 1 ms
Bandwidth	1–100 Mbps	10–1000 Mbps	100–1000 Gbps
Reliability	Low to moderate	High	Moderate to high
Security	Low	High	High
Scalability	High	High	Low to moderate
Energy consumption	Low to moderate	Moderate to high	High
Computing resources	Low to moderate	Moderate to high	High
Cost Efficiency	Moderate	High	Low
Data Privacy	Moderate	High	High
Fault Tolerance	Low to moderate	High	Moderate to high

Table 1. Analysis of various computing mechanisms in applicative scenarios.

federated learning is the optimal choice, and tiny models are instantiated on the roof layer¹⁹. The local learned gradients are shared to update the global models in the cloud. Thus, it leverages dual support with independence and collaborative learning, where local data is not shared.

Research motivation

In our proposed system, we implement an innovative strategy that leverages the interaction of cloud, roof, and dew layers, with robust support provided by the fog layer. The dew layer creates a localized, distributed environment for data acquisition, processing, and analysis in collaboration with the fog layer, thereby reducing latency and lowering bandwidth demands. The roof layer, in turn, supplies cloud-like services—such as high-speed connectivity, substantial processing power, and extensive storage—to the dew layer, ensuring that it can access necessary resources without losing its localized advantages. Furthermore, fog computing is deployed at the network edge to perform real-time data processing and analysis, which speeds up response times and minimizes the volume of data sent to the cloud.

Novelty

IoMT setups forward massive sensor data to the fog layer for analytical support. Due to the heterogeneity of end devices and link variability, variable delays in data aggregation deem time-based analytics more accurate. Possible solutions of layered fog nodes¹⁰, federated edge learning^{19,20}, and streaming analytics²¹ are proposed. Still, they need to be more robust to handle the variability and velocity of ingested data. Thus, the services switch to best-effort mode, which is not viable for mission-critical healthcare IoMT setups. Thus, in this work, a framework *DeW-IoMT* is presented that addresses these challenges and performs operations with low latency. Our framework is constructed hierarchically, where the dew-roof-fog-cloud computing paradigm is proposed for IoMT. The DC layer supports distributed resource management, while the roof layer provides data security. This approach offers greater flexibility and user control at ultra-low latency, high mobility, and data security. Additionally, the RC layer integrates security services and forwards quick resource requirements to the dew layer, ensuring resilience against attacks. Furthermore, the DC layer can be fine-tuned to support patient-specific needs, resulting in a more personalized healthcare system. Our proposed framework offers a reliable and accurate solution for mission-critical IoMT setups, paving the way for a smarter and more efficient healthcare system.

Article contributions

The research contributions of the article are presented as follows.

- We propose a hierarchical architecture, *DeW-IoMT*, to support remote patients with quick, scalable, and dependable critical support.
- The patient healthcare indicators (pulse, heart rate, and electrocardiogram signal) are recorded at the dew node (Arduino Uno board). They are communicated to the alert module through a serial connection, where critical indicators are monitored and compared against the threshold values.
- The performance of the proposed framework is evaluated for parameters like response time, energy dissipation, and bandwidth usage against recent approaches to indicate the framework's efficacy.

Layout

The article is divided into four sections. Section “[Related work](#)” presents the related work, which presents the state-of-the-art (SOTA) approaches of cloud, fog, edge, and dew computing in IoMT. Section “[DeW-IoMT: the proposed framework](#)” presents the schematics of the proposed *DeW-IoMT* scheme, which discusses the framework, its components and connections, and the testbed setup. Section “[DeW-IoMT: performance evaluation](#)” presents the performance evaluation of the proposed scheme, and finally, section “[Conclusions](#)” concludes the article with the future scope of the work.

Related work

Recently, many schemes have been proposed to leverage improved healthcare services in IoMT ecosystems. These schemes include computing paradigms like cloud, fog, edge, and dew computing for task processing, analytics, management, resource allocation, and offloading. Table 2 presents a comparative analysis of the different schemes/frameworks, where we compare our proposed schemes with the existing SOTA approaches.

Approaches for job allocation^{31,34,35}, and fog node selection^{16,32,40} are the most discussed for in healthcare setups, where critical indicators are analyzed for making relevant predictions. Flexibility and wise decision-making are still difficult, though. Karmakar et al.²⁵ proposed a biosignal monitoring scheme, named *MedGini* for Internet of Health Things (IoHT), where the emphasis was laid on the time and space-efficient dissemination of healthcare data. For the same, the authors exploited the requirement of IoT-based wireless connectivity, where the data was temporarily stored on local dew nodes and later synchronized with the cloud servers. The synchronization process of DC-CC is optimal through the use of the Gini Index and Shannon entropy, which removes erroneous signals, and outliers in the collected data. The scheme was compared for cost and network parameters on real-time collected data.

Authors in²⁶ proposed a Convolutional Neural Network (CNN)-based framework to predict the patient's mental state based on behavioral psychology. The analysis was done on the dew layer; hence, bypass connections made the CNN model lightweight. This also ensured optimized gradient propagation in the CNN, and the results were evaluated on the FER-2013 dataset. Another dataset, CK+, was designed and used, and the model reported an accuracy of 96.12%. Authors in⁴¹ discussed about optimal function placement in serverless computing to optimize resource efficiency and delay. Approaches like ML, heuristics, and local models are discussed. Serverless functions can be integrated at dew layer to optimize resource and instruction usage. In⁴², authors discussed about

Authors	Year	Dew	Fog	Edge	Cloud	Performance Metrics	Advantages	Limitations	Application Area
The proposed scheme	2024	Y	Y	N	Y	Response time, energy, bandwidth	Low latency, enhanced security, scalability	Increased orchestration complexity	Health-care IoMT
Vinu and Diwan ²²	2024	Y	Y	N	N	Consensus delay, throughput	Decentralized, secure via blockchain	High consensus overhead	General IoT
Zhao et al. ²³	2024	Y	Y	N	Y	Cache hit ratio, latency reduction	Efficient caching, improved network performance	Limited to caching improvements	Smart cities IoT
Ghosh and De ²⁴	2023	Y	N	Y	Y	Delay reduction (28%), energy savings (12%)	Reduced network delay and power consumption	Reliant on stable D2D communication	Wireless networks
Karmakar et al. ²⁵	2022	Y	N	Y	Y	Data synchronization efficiency	Effective synchronization using statistical measures	Scalability issues in heterogeneous settings	Internet of Health Things (IoHT)
Podder et al. ²⁶	2022	Y	N	Y	Y	Prediction accuracy, latency	Lightweight CNN, enhanced gradient propagation	Dataset-specific tuning may be needed	Health-care biosignal monitoring
Afaq and Manocha ²⁷	2022	Y	Y	N	Y	Precision, recall, accuracy	Effective prediction using GRU and Naïve Bayes	Limited generalizability across conditions	Health-care diagnostics
Poonia et al. ²⁸	2021	Y	Y	N	Y	Bandwidth usage, scalability	Real-time performance, scalable design	Complex multi-layer coordination	COVID-19 management, Healthcare IoT
Mukherjee et al. ²⁹	2021	Y	N	Y	Y	Communication delay, success rate	Improved connectivity and caching in UAV networks	Limited to UAV-supported scenarios	Smart cities, UAV-based IoT
Jung-yeon and Kaddoum ³⁰	2021	N	N	N	N	Offloading efficiency, resource utilization	Adaptive resource allocation via DRQN	Not applicable for dew-based scenarios	General fog computing in IoT
Zhou et al. ³¹	2020	N	Y	N	N	Long-term delay reduction	Adaptive offloading based on task properties	May underperform in volatile environments	Task offloading in IoT
Fan et al. ³²	2020	N	Y	N	N	Offloading latency, resource allocation	Effective D2D pairing and offloading	Depends on proximity and D2D availability	Edge computing in IoT
Yang et al. ³³	2019	N	Y	N	N	Task completion time	Efficient paired offloading	Scalability may be limited	Fog computing in IoT
Lan et al. ³⁴	2019	N	Y	N	N	Resource allocation efficiency	Fair allocation using statistical measures	Complexity increases with user numbers	Resource management in IoT
Li et al. ³⁵	2019	N	Y	N	N	Energy consumption, offloading delay	Energy efficiency	Overhead due to offloading computation	Energy constrained IoT
Mutlag et al. ³⁶	2019	N	Y	N	N	Scalability, real-time performance	Real-time scalability	High resource costs in large systems	E-health services
Sodhro et al. ³⁷	2019	N	N	Y	Y	Service placement efficiency	Geo-distributed intelligence improves service quality	Complex deployment strategy	Health-care IoT networks
Ray et al. ³⁸	2019	Y	N	N	Y	Real-time context analysis	Context-aware decisions at dew level	Limited scalability in extensive networks	Context-aware IoT applications
Rahman et al. ³⁹	2018	N	Y	N	Y	Local processing speed, storage efficiency	Provides real-time local processing	Dependent on gateway capacity	Embedded IoT applications

Table 2. Comparative analysis of the proposed framework with existing SOTA frameworks.

improper data offloading in serverless computing systems and discusses about different offloading schemes to minimize resource consumption and latency. Afaq and Manocha²⁷ proposed a DC model for diarrhea prediction, where severity values were captured and analyzed through a probabilistic weighted Naïve Bayes algorithm. This classified the abnormality conditions, and the output was sent to a multi-scaled Gated Recurrent Unit (GRU) unit to analyze the correlation between the patient's eating habits and environmental conditions. The model was compared for precision, recall, and accuracy, and it was found that GRU has a precision of 93.26%, and the precision value of Naïve Bayes was reported to be 97.15%. The decision-making was kept at the dew layer for serious patients, and monitoring was done at the cloud layer.

Poonia et al.²⁸ proposed a three-layered (dew-fog-cloud)-based healthcare monitoring framework, named as CONFRONT, which helped in the preliminary avoidance of COVID-19 virus sabotage to the human body. The dew layer made the scheme scalable and allowed low-cost sensor wearables to be used with fast uptime. For high computational requirements, fog offloading was supported, which resolved most queries, and high computational requirements were only forwarded to the cloud layer. Ghosh and De²⁴ proposed a DC architecture for D2D communication scenario for wireless networks. For the same, the authors have proposed a coalition game and operate cooperatively with the dew storage layer in the event of disconnections. The proposed scheme has a reduced delay of $\approx 28\%$, and energy consumption has been reduced by 12%. Jazayeri et al.⁴³ introduced a FC based IoT architecture, termed as mobile fog that can offload data to cloud based on number of executable modules under operates. It forms a best fit scheduling for reducing latency among tasks, and sequentially decides about the placement of tasks for execution (whether at mobile fog or cloud) layers. In⁴⁴, authors discussed about local data processing and content caching mechanisms at edge nodes based on metrics like traffic, and available bandwidth. Jeyaraj et al.⁴⁵ have proposed the performance of IoT applications where dew-cloud layers provide resources. The work discussed resource provisioning, load balancing, and workload allocation in heterogeneous networks. Marjan Gusev⁴⁶ proposed an edge-assisted offline dew storage server that allows end devices to communicate for resource management. For the same, an AI-driven classifier is designed to understand critical tasks on the edge and dew server. A use-case of cardiologist prediction for data collected

from wearable sensors is presented. Catalani et al.⁴⁷ presented a deep learning model for IoMT wearables that monitors cardiac problems. The model interprets the electrocardiogram signals from ECG sensors. The model is fine-tuned and is executed on an embedded processor. Further, multicore architecture is considered to improve the classifier execution time. The tradeoff is measured in terms of latency and event detection ratio. Authors in²⁹ proposed a scheme named *Dew-Drone*, which allowed Unmanned Aerial Vehicles (UAVs) to communicate data captured from smart cities⁴⁸, industrial applications, and society 4.0 through a dew-cloud infrastructure. An opportunistic communication framework was realized for faster UAV response in conditions of intermittent network connectivity. The scheme discussed a hardware testbed, and UAV delivery accuracy was reported to be significantly high, with a success rate of 91.4%. The dew buffer size was kept at 150 MBs, and the obtained latency to UAV-ground communication was 20.04 milliseconds (ms), which is minimal. proposed resource optimization in Dew-fog layers itemutlag2019enabling⁴⁹, where energy consumption was considered the cost factor to solve the optimization problem. The approaches involved real-time data processing by sensor-based applications. Authors in⁵⁰ proposed a low-powered transmission protocol to leverage the connectivity of near-edge nodes and conserved energy dissipation.

In⁵¹, the authors exploited the concept of neighborhood edge networks to solve computations at the edge nodes for IoT health architectures. The edge nodes communicated with fog nodes for mobility control, rapid response, and storage requirements. Zhao et al.²³ presented a load balancing of fog servers for vehicular networks using a dew server. They introduced a innovative clustering algorithm with caching at dew node, based on the file request and popularity. The testing showcases the strategy's robustness, especially in maintaining cluster leadership and improving cache hit rates. Authors in³⁹ proposed an edge-gateway assisted scheme for body sensor networks, where a geo-distributed intelligence layer was constructed between the cloud and edge node. A fog layer was induced to support offloading, which improved the scheme's reliability. Sodhro et al.³⁷ worked on the Quality of Service (QoS) of edge computing applications in healthcare setups. Authors in⁵² proposed the *ubehealth* framework, which used edge computing, IoT, and deep learning to handle different issues with smart health. The architecture provided better network service quality. Deep learning was implemented to estimate the network traffic to maximize data rates, caching, and routing choices. Authors in⁵³ proposed a clustered approach based on the Hybrid Whale Particle Swarm Optimization (HWPSO) algorithm in an IoT-fog architecture.

In summary, while cloud and FC have been extensively explored to reduce latency and improve data processing in IoMT applications, DC has emerged to empower local device intelligence and offline capabilities. However, the lack of a dedicated RC layer in most existing solutions limits on-site data aggregation and secure resource management. Recent initiatives underscore the importance of a formalized RC framework to bridge the gap between edge and cloud processing⁵⁴. Our proposed DeW-IoMT framework integrates these paradigms into a unified, hierarchical architecture that leverages the benefits of dew, roof, fog, and cloud computing. This comprehensive approach not only enhances performance and security but also paves the way for scalable and resilient IoMT deployments.

DeW-IoMT: the proposed framework

This section presents the schematics of the dew-roof-fog-cloud-assisted layered framework, named *DeW-IoMT*. Figure 1 presents the framework details.

The framework is designed specifically for timely response to critical patient health requirements; thus, the local operations are executed locally at the dew layer. The suggested framework seeks to deliver a user-centric, adaptable, customized, and quick response to tasks. The details of the layers and associated components are presented as follows.

- *IoT devices*-In the framework, we consider that there are n patients registered in a hospital, designated as $P = \{P_1, P_2, \dots, P_n\}$. Any i^{th} patient P_i is equipped with k sensors (wearables or implants), denoted by $\{s_1, s_2, \dots, s_k\}$. A sensor node (s_j) is presented as a three tuple as follows: $s_j = \{s_{id_j}, s_{type_j}, s_{status_j}\}$, where s_{id_j} , s_{type_j} , and s_{status_j} denotes the sensor ID, the type of the object for which the data collection takes place (such as temperature, pulse, etc.), and the status of the sensor, i.e., active or idle, respectively, and $j \in k$. The sensor IDs are registered and mapped to P_i through a mapping function $M(F)$. The data collected from the sensors, denoted as $\{D_{s_1}, D_{s_2}, \dots, D_{s_k}\}$ is sent to the Arduino controller, which is at the dew layer *DL*. We consider 8, 16, and 32-bit microcontrollers, whose prime operation is to transmit data to the dew service node. Depending on the end application requirement, the sensors are networked via Wi-Fi, Bluetooth, and Long Range Radio (LoRa) networks. We assume that sensor nodes are placed near an associated hotspot node (H_s) for sending the data to the DC layer *DL*. The *DL* nodes might be hospital-owned nodes or rented by third-party providers in close proximity. Specifically, we use a 9 Volt (V) adapter, and a client-server communication is set up at *DL*.
- *Dew Layer*-The dew layer *DL* operates as a distributed hotspot network access⁵⁵. The hotspot range uses the 2.4 GHz unlicensed band. However, end-IoT devices D_n can operate with ranges from 3.6–9.60 GHz band. The communication protocol is IEEE 802.11n, and for long-range communication, we shift to IEEE 802.11ax, with a data rate of 900 Mbps. In our work, we considered Wi-Fi, with an associated data capacity of 50 Kbps for communicating channels. At *DL*, we consider q dew nodes, represented as hotspot nodes $\{H_1, H_2, \dots, H_q\}$. They communicate with the Raspberry Pi, AML-S905X, UDOO Blot v3 boards for communication. Any H_q node can monitor any P_i patient critical indicators, denoted as $C(P_i)$, and support disconnected/intermittent Internet connectivity to the cloud layer nodes.
- *Roof Layer*-At the roof layer (*RL*), security and privacy mechanisms, like encryption and role-based access control, are integrated. It ensures that data from the *DL* is securely transmitted to the fog-cloud layers. The sensitive data attributes are hidden, and the data is encrypted and protected from unauthorized access. In our framework, we consider that there are r roof layer nodes, denoted by $\{R_1, R_2, \dots, R_r\}$. These nodes are

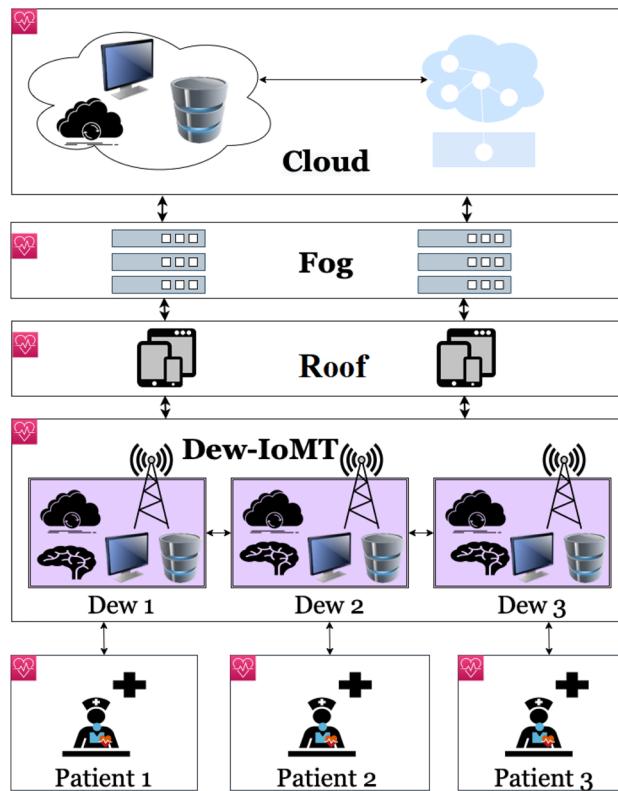


Fig. 1. *Dew-IoMT*: The proposed framework (drawn using draw.io desktop software version v26.2.2 <https://github.com/jgraph/drawio-desktop/releases/tag/v26.2.2>).

connected to the dew layer nodes and the cloud layer nodes via wired or wireless networks, such as Ethernet or Wi-Fi. Each R_i node has a unique ID, denoted by r_{id_i} , and is responsible for specific tasks, such as data encryption or access control. The roof layer nodes also have resources, such as computing power or storage, which can be allocated to the dew layer nodes as needed. These resources are denoted as R_{res_i} , where R_{res_i} is the resources available at node R_i . To assure privacy-based access control, we denote the set of authorized users for patient P_i as $A(P_i)$, and the set of sensitive data for P_i as $S(P_i)$.

- *Fog Layer*-At the fog layer *FL*, the collected data which are not immediately critical are forwarded. At this later, we consider that the data collected, which have *health indicators*, denoted by I_f , mapped to s_k , which is normalized for every indicator between range $[0, 1]$, and is sent to the training classifier. However, we consider lightweight models for healthcare analysis at the fog, and the results are sent back to the *DL* for display. This saves critical time for analysis at the cloud node, as critical predictions are returned from the fog node itself.
- *Cloud Server*-Once the network connection becomes stable, the collected data is sent to the cloud server for analytics. At the cloud layer *CL*, heavyweight analytics models are used. CNN models like RestNet-50 and Inception v3 are normally used for imaging analysis. Similarly, time-series data is analyzed through deep learning algorithms like Long-Short Term Memory (LSTM) units or GRUs. To induce the interpretability of the results, Explainable AI (XAI) techniques are used⁷. XAI techniques include Shapley Additive Explanations or Local Interpretable Model Agnostic Explanation (LIME) explainers used to indicate which parameter I_f had more impact on the output prediction $O(p)$.

A sequence diagram interaction

Figure 2 represents the information flow between the different layers in the framework, based on the discussions presented in Fig. 1.

In this diagram, the IoMT sensors continuously capture patient vital indicators, which are then transmitted to the Arduino-based dew layer via serial communication. The dew node executes functions such as `Collect()`, `Validate()`, and `Critical_Case()` to check if the sensed data exceed predetermined thresholds. Upon detecting abnormal values, the dew layer immediately triggers local alerts and forwards the data through its communication interface. The data then flows to the roof layer (implemented on a smartphone), where it is aggregated, secured using advanced encryption standard (AES) encryption and role-based access control, and subsequently forwarded to the fog layer. The fog layer performs lightweight analysis and, if necessary, passes critical information to the cloud server for deeper analytics and long-term storage. This coordinated, multi-layer approach ensures reg-time response and resilience even under intermittent connectivity.

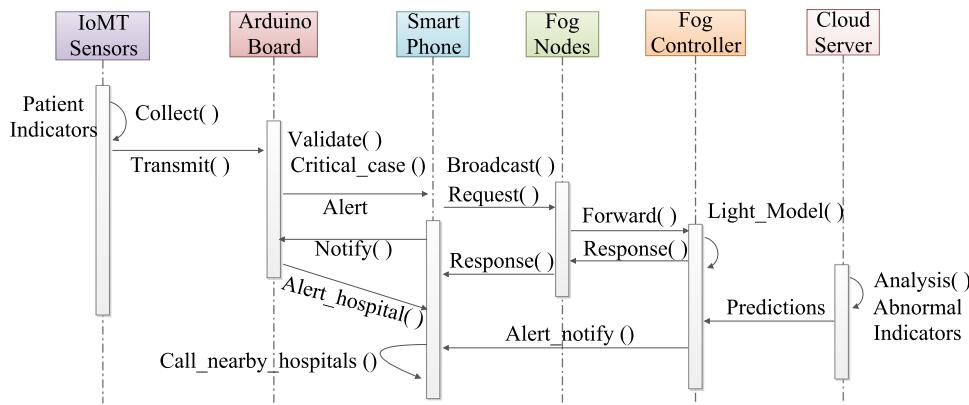


Fig. 2. Sequence Diagram: Communication between different entities in the framework (drawn using draw.io desktop software version v26.2.2 <https://github.com/jgraph/drawio-desktop/releases/tag/v26.2.2>).

Components and testbed setup

In this subsection, we discuss the details of the hardware components used in the framework and the testbed setup. The detail is presented as follows.

Components

The basic components used in the framework are as follows.

- *IoMT Sensors (Data collection)*-We take plug-and-play IoMT sensors to record the patient vital indicators. We connect the sensors to our Arduino Uno R3 board to measure the patient's heart rate, pulse, blood pressure, and temperature. The sensors are equipped with a noise-removal circuit, removing unwanted signals that might change readings. These sensors are connected to an Arduino board at pin number $A0$.
- *Jumper Wires (Connections)*-Electrical cables connect the breadboard or circuit parts with other components without soldering. These jumper wires make all connections of our circuit.
- *Arduino Uno R3 (Dew Layer)*-Arduino UNO R3 is a microcontroller board based on ATmega328p, and it is designated as our dew layer DL . We read data from k sensors, where a matching rule algorithm is stored, which checks the measured I_f against threshold indicators, denoted by Th_I . In case $I_f > Th_I$, then an alert is raised, and the nearest hospitals are informed for intensive patient care. In case of non-serious conditions, the data is forwarded to RL through the uplink U_{rf} , with data rate $R_{U_{rf}}$. The communication is done through serial communication via the Universal Asynchronous Receiver-Transmitter (UART) protocol. Thus, the decision at the DL does not require any Internet connectivity, and in such cases, the data is stored in a dew database, denoted by $Db(DL)$.
- *LCD Display (Alerts at dew)*-The alerts are displayed on a 16×2 LCD display, which displays 32 information characters. Each character is made of 40 pixels. Thus, 40×32 , which is 1280 pixels, is displayed for messages.
- *Raspberry Pi (Roof Layer)*-Raspberry Pi is used as the roof layer device in our framework. It is a single-board computer with a Broadcom system-on-a-chip (SoC). It has a 64-bit quad-core ARM Cortex-A72 CPU, clocked at 1.5 GHz, which provides high processing power. It has a RAM of 4 GB, which provides sufficient memory to run complex applications. It has built-in wireless connectivity, including Wi-Fi and Bluetooth, based on IEEE 802.11 b/g/n/ac protocols. Additionally, it has Gigabit Ethernet for wired networking support. The received data is encrypted and is represented as $E_k(R_{U_{rf}})$, where K represents the symmetric AES encryption key. For any user U , and access control role, Rl_u is defined with a specific object (a resource the user can access) as O_K . The access rights are defined as tuples that map R_u with O_K . As an example, any authorized doctor ($Rl_u = \text{Doctor}$) can access patient temperature object $O_K = \text{Temp}$ in read mode only. The details can be shown as a tuple $Tu = (\text{Doctor}, \text{PatientTemp}, \text{Read})$. Based on the temperature, the critical actions can be written in the patient health record as $Au = (\text{Doctor}, \text{PatientTemp}, \text{Write})$, where explicit identifiers (name of patient, address) are hidden. This is appended with previous data, encrypted, and sent to the fog node for analysis. The offloading strategy of Raspberry Pi is decided based on its computational power and storage capacity. In low storage, data can be offloaded to ESP32 (fog node) or Blynk Cloud for task computation. This is done by establishing a communication link directly with the fog-cloud nodes.
- *ESP32 chip (Fog Node)*-The ESP32 acts as the fog node in our framework and involves a Tensilica Xrtenza 32-bit processor with 1-2 cores. The operating clock frequency is 240 MHz, allowing wireless connectivity based on IEEE 802.11 b/g/n/i protocol. It has a ROM of 448 KB, fast RAM of 520 KB, and 8 KB for sleep modes. It can support external Static Random Access Memory (SRAM) with flash encryption. Thus, with sufficient memory and networking support, it can execute lightweight analytics and support communication with the cloud layer CL via the uplink U_{fc} , with data rate $D_{U_{fc}}$. It is connected with DL (Arduino) using the Tx2 RX2 pins of ESP32.
- *Blynk Cloud (Cloud Node)*-The healthcare data received from Arduino will be sent to the Blynk cloud, where monitoring and analysis can be done. The cloud node uses Blynk Cloud, an open-source cloud-based IoT

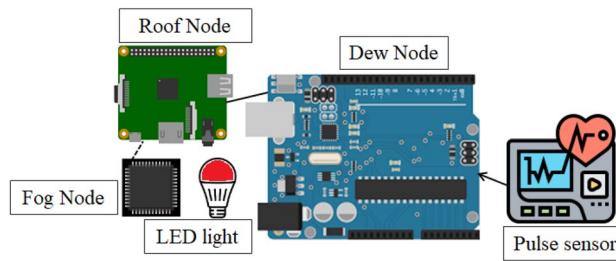


Fig. 3. Testbed setup of the proposed scheme (drawn using draw.io desktop software version v26.2.2 <https://github.com/jgraph/drawio-desktop/releases/tag/v26.2.2>).

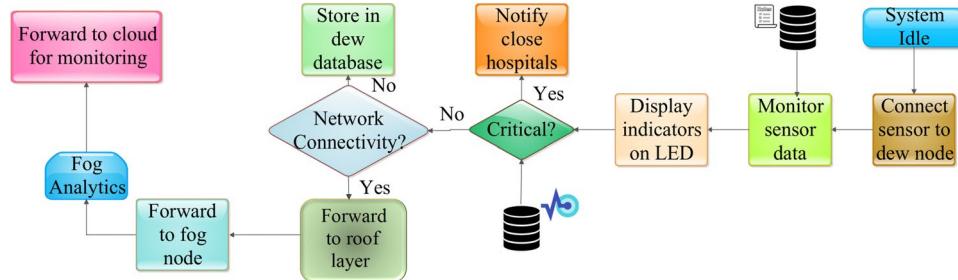


Fig. 4. The operational flow of the framework (drawn using draw.io desktop software version v26.2.2 <https://github.com/jgraph/drawio-desktop/releases/tag/v26.2.2>).

solution. It can connect to any smartphone with an Android version of 4.2 or higher and an iOS version of 9 or higher, with in-built networking support.

Testbed setup

The testbed setup in Fig. 3 shows the connection of the pulse sensor (PS) with Arduino.

- The dew node is an Arduino Uno. The PS's data pin is connected to the Arduino's A0 pin, and the Arduino is connected to the Raspberry Pi via USB.
- The roof node is a Raspberry Pi 3 Model B+. The DHT11 sensor's data pin is connected to GPIO17 (pin 11), and the LED's anode is connected to GPIO27 (pin 13) and its cathode to GND (pin 6). The Raspberry Pi communicates with the fog node via WiFi.
- The fog node is an ESP32 chip. Its RX2 and TX2 pins are connected to GPIO15 and GPIO4 of the Raspberry Pi, respectively. The ESP32 is also connected to the BME280 sensor via I2C, with SDA connected to GPIO21 (pin 40) and SCL connected to GPIO22 (pin 38).
- The cloud layer is Blynk Cloud, which receives data from the fog node via WiFi.

The control flow of the proposed framework

This subsection discusses the operational flow of the *DeW-IoMT* framework. Fig. 4 presents the details.

The framework considers that sensors (pulse, blood pressure, and ECG) are connected to P_n , and the data is sent to Arduino UNO R3. Based on the proposed monitoring and alert algorithm, as depicted in Algorithm 1, it measures the indicators and compares them to normal threshold values of health indicators.

The normal indicators are 60–120 beats/min for the heart, 120–140 mmHg for pressure, and 98.6 F for temperature. If the measured indicator exceeds the threshold, it is displayed on the LCD, and the data is forwarded to ESP32 via the UART protocol at 9600 bps.

The operational flow considers the system is initially in an idle state. Once data is received from sensor nodes $X = \{Pu, Te, Pr\}$, the dew node generates alerts based on the threshold conditions displayed as messages on the LCD. If network availability exists, the data is forwarded to the fog layer FL , and lightweight analysis is done to present basic prediction results. Further, the data is sent to CL for constant monitoring. The structural elements at the dew layer are as follows.

- *Dew Manager*: The dew layer executes the basic checking function `Validate()`, which matches the sensor data to the threshold values via the `Collect()` function. The critical conditions are covered via `Critical_Case()` function, and alerts are forwarded to the smartphone.
- *Communication Interface*: Plug-and-play networking elements are connected via the WiFi interface, which sends the data to the fog nodes via the `Request()` function.

- *Device Handler*: At the fog layer, we consider two functionalities, the basic fog node and the fog controller, which form a request-response protocol via IoT message passing protocol such as Message Queue Telemetry Transport (MQTT), and manage the connection between the dew server block and PS using the message passing.
- *Dew Storage*: In case of disconnection, the data is stored in a dew database, where the critical findings can be analyzed based on dew intelligence. Once the reconnection state happens, the data is forwarded to higher fog-cloud layers.
- *Roof Layer*: In case of reconnection, we consider that the data stored in the dew database is forwarded to the roof layer, where the AES encryption and role-based Access Control (ACL) listing is applied. Further, computationally intensive tasks are divided and sent to the roof for offloading, which computes and sends the result back to the dew layer.

```

1: Input:  $Pu \leftarrow \text{Pulsedata}$ ,  $Te \leftarrow \text{Tempdata}$ , and  $Pr \leftarrow \text{Pressuredata}$ 
2: Output: Display on LCD
3: Procedure READ AND DISPLAY SENSOR DATA( $Pu, Te, Pr$ ):
4:   Read  $X \leftarrow Pu, Te, Pr$ 
5:   DisplayX
6:   Alert( $X$ ) in case  $X > \text{Threshold}$ 
7: Procedure ALERT( $X$ ):
8:   if  $Pu \geq 100$  and  $Pu \leq 120$  then
9:     Display "Heart Rate is normal"
10:  else if  $Pu \geq 120$  then
11:    Display "Heart Rate is fast, needs attention"
12:  else if  $Pu \leq 100$  and  $Pu \geq 50$  then
13:    Display "Heart Rate is slow, needs attention"
14:  end if
15:  if  $Te \geq 100.4F$  then
16:    Display "Fever, needs attention"
17:  else if  $Te \geq 97F$  and  $Te \leq 99F$  then
18:    Display "Normal Range"
19:  else if  $Te \leq 97F$  then
20:    Display "Low body temperature, needs attention"
21:  end if
22:  if  $Pr \geq 140(H)$  and  $Pr \geq 90(L)$  then
23:    Display "High Blood Pressure, needs attention"
24:  else if  $Pr \leq 120(H)$  and  $Pr \leq 80(L)$  then
25:    Display "Normal Range"
26:  else if  $Pr \leq 100(H)$  and  $Pr \leq 60(L)$  then
27:    Display "Low pressure, needs attention"
28:  end if
29: Procedure SEND SENSOR DATA TO ESP32( $X$ ):
30:   Begin Serial Monitor
31:    $Send \leftarrow X$ 
32: Procedure READ SENSOR DATA FROM ARDUINO( $X$ ):
33:   Connect to Wi-Fi
34:   Connect to Cloud Server
35:   Display Data on Cloud

```

Algorithm 1. *DeW-IoMT*: The proposed monitoring and alert system

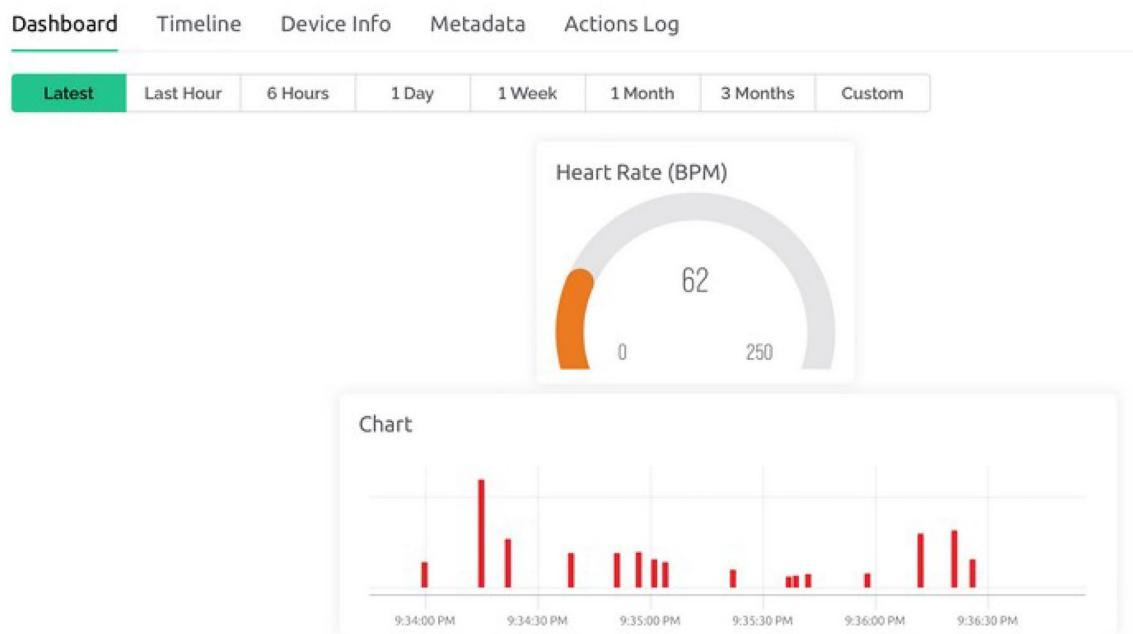


Fig. 5. Monitoring of vital indicators on Cloud.

Communication Protocol between ESP32 and Blynk Cloud: To facilitate efficient and secure data transfer between the ESP32-based fog node and the Blynk Cloud, our system employs the MQTT protocol in accordance with industry standards. The ESP32 is configured as an MQTT client operating over IEEE 802.11n/802.11ax wireless networks, ensuring robust connectivity with low latency. It uses a unique authentication token provided by Blynk to establish a secure session, conforming to IEEE 802.11i (WPA2/WPA3) security standards at the link layer. Sensor data collected by Arduino-based dew devices is first aggregated and pre-processed at the roof layer, then forwarded to the ESP32. Here, the data is published to designated MQTT topics on the Blynk Cloud with a predefined Quality of Service (QoS) level (typically QoS 1 to guarantee message delivery). Simultaneously, the ESP32 subscribes to control command topics to receive real-time configuration updates. To further enhance security, all MQTT messages are encapsulated within a TLS/SSL channel, adhering to the cryptographic standards outlined in IEEE P1363. This combination of secure, standards-based communication protocols minimizes latency, ensures data integrity and confidentiality, and supports reliable real-time monitoring even in challenging network conditions.

Algorithm analysis and practical use-case

The computational overhead of the proposed algorithm is primarily determined by the number of sensor inputs and the operations performed at each layer. The core procedures-such as sensor data collection, threshold validation, and offloading decision-making-exhibit linear time complexity, i.e., $O(n)$, where n represents the number of sensors per patient. This lightweight processing is distributed across the dew and roof layers, thereby reducing the load on the fog and cloud nodes. Consequently, the framework scales efficiently with an increasing number of patients, as each node operates independently without causing system-wide bottlenecks. The inherent modularity of the architecture further facilitates scalability, making it well-suited for large-scale IoMT deployments.

To illustrate the practical applicability of the proposed framework, consider a hospital scenario where multiple patients are monitored concurrently. Each patient is equipped with wearable sensors that record vital signs such as heart rate, temperature, and blood pressure. These sensors send data to an Arduino Uno (the dew layer) located at the patient's bedside. The dew node rapidly validates the readings; if any measurement exceeds the safe threshold, an immediate alert is generated and displayed locally. Simultaneously, the data is transmitted to a nearby smartphone (the roof layer) which aggregates and encrypts the information before forwarding it to an ESP32-based fog node. The fog node performs further analysis to detect anomalies and, if required, instructs the system to notify medical staff. For more complex evaluations, the processed data is then relayed to the cloud for comprehensive analysis and long-term storage. This example demonstrates how the DeW-IoMT framework effectively reduces latency and enhances system reliability, ensuring timely interventions in critical healthcare scenarios.

Monitoring and analysis of data on cloud

This study references a publicly available dataset created by us (*Remote data monitoring file*, Zenodo - <https://doi.org/10.5281/zenodo.1450025>). The dataset is titled *Remote Data Monitoring File* and it provides synthetic sensor readings for heart rate, body temperature, and blood pressure (systolic and diastolic). Based on the dataset, we have analyzed the data on Blynk Cloud and created a minimalistic design of the basic components through a

user-friendly dashboard, which does real-time analytics on the readings to ensure seamless operations. Fig. 5 refers to the created dashboard.

The data captured from the IoT devices at the dew layer is sent to the cloud to provide real-time feedback. This feedback is plotted through appropriate visualization with matching colors and fonts. We have selected the bar chart to display the pulse rate information, as it provides an intuitive and effective medium to visualize and measure changes in data over time. The fonts are kept to maximize the reading visibility, and the web page is designed to be responsive and optimized for device width (different views for desktops, mobile, and palm tops). The layout and size of the visualization ensure that the dashboard is easy to navigate and interact with on all devices. The IoT device is authenticated on the dashboard via tuple T , which contains template_id, template_name, and authentication_code. This allows Blynk Cloud to identify the device and allow rule-based access to the data. Once the pulse data is displayed on the dashboard, cloud nodes perform analytics using AI models to analyze the data further. These analytics can provide insights into trends and patterns in the data that may be difficult to discern with simple visualizations, allowing users to make more informed decisions about their health and wellness.

Security and access control

In the proposed framework, we implement security and access-control mechanisms at RL . We present the possible attack scenarios on the proposed framework and then the security mechanisms built to evade them. The details are presented as follows.

Security attacks

The possible security attacks are as follows.

- Data interception: This attack involves an adversary intercepting and accessing the data being transmitted between the dew and roof layers. The adversary can use this data for malicious purposes such as identity theft, financial fraud, or blackmail.
- Data tampering: This attack involves an adversary modifying the data being transmitted between the dew and roof layers. The adversary can use this to inject false data into the system, leading to incorrect decisions or actions based on that data.
- Denial-of-service (DoS): This attack involves an adversary disrupting the normal functioning of the system by flooding it with a large number of requests or by overloading the resources of the system.

Attack countermeasures: encryption and role-based access control

To prevent these attacks, the integrated security mechanisms are presented as follows.

- Data encryption: Let E represent the encryption function that encrypts the data collected from the sensors. This function takes the plaintext data D as input and produces ciphertext data C as output, such that $C = E(D)$. We implement this function using the Advanced Encryption Standard (AES) mechanism.
- Role-based access control: Let U be the set of all users who have access to the patient data. Each user is assigned a role Rl from a set of predefined roles $\{Rl_1, Rl_2, \dots, Rl_m\}$. Let P_i be the patient whose data is being accessed. We define a role-based access control policy as a function that maps each user $u \in U$ and patient P_i to a set of permissions $P_{u,i} = \{p_{u,i,1}, p_{u,i,2}, \dots, p_{u,i,k}\}$, where $p_{u,i,j}$ is a binary variable that indicates whether user u is allowed to access the j^{th} sensor data of patient P_i . We enforce this policy by using access control lists (ACLs) or by implementing it in the form of a role-based access control (RBAC) system.

Parameters	Value
RAM of Fog node	1 GB
CPU length of Fog node	40000 MIPS
RAM of Roof node	4 GB
CPU length of Roof node	30000 MIPS
RAM of cloud VM	8 GB
CPU length of cloud VM	44800 MIPS
Idle power of Fog node	16×83.25 W
Idle power of Roof node	8×50 W
Idle power of cloud VM	16×83.25 W
Busy power of Fog node	16×103 W
Busy power of Roof node	8×70 W
Busy power of cloud VM	16×103 W
Uplink bandwidth	90–110 Mbps
Downlink bandwidth	140–160 Mbps

Table 3. Simulation parameters (iFogSim).

Resource offloading: dew to roof

In this subsection, we discuss the offloading mechanism of memory-intensive tasks at the *DL* to *RL*. Let T be the set of all tasks at the dew layer, and M be the set of all memory-intensive tasks at the dew that can be offloaded to the roof layer. We assume the roof layer has more resources, such as power, I/O, and bandwidth, than the dew layer. Let Res_d and Res_r denote the available resources at the dew and roof layers, respectively, where $Res_d < Res_r$.

Let us assume that DL_t is the execution time of task t at dew, and RL_t is the execution time at roof layer. The overall latency of the dew-to-roof offloading for a given task t is presented as follows.

$$L_t = \max(E(DL_t), E(DL_t \rightarrow RL_t)) \quad (1)$$

where $E(DL_t)$ denotes the execution of the task at dew fully, and $E(DL_t \rightarrow RL_t)$ denotes the partial transfer of the task from dew to the roof. It also involves the computation and process task formation time at an available bandwidth B_d . Thus, an estimated offloading time T_o would depend on the decision to execute locally or offload.

The offloading mechanism aims to minimize the overall latency of executing all tasks. We define the objective function as follows.

$$\min \sum_{t \in T} L_t \quad (2)$$

where L_t is the latency of task t . To offload a task $t \in M$ from the dew layer to the roof layer, we use the following equation to estimate the offloading time T_o .

$$T_o = \frac{Si}{B_r} + \frac{Si}{B_d} + \frac{Cmp_t}{R_r} \quad (3)$$

where Si is the task data size, B_r and B_d are the available bandwidths at the roof and dew layers, respectively, and Cmp_t is the computation time of task t .

If the estimated offloading time T_o is less than the execution time of the task t at the dew layer, the task is offloaded to the roof layer. Otherwise, the task is executed locally at the dew layer.

$$\begin{cases} \text{if } T_o < t & \text{Offload task to roof layer} \\ \text{otherwise} & \text{Execute task locally at dew layer} \end{cases} \quad (4)$$

Once a task is offloaded to the roof layer, it is executed there, and the results are sent back to the dew layer. We assume that the communication between the dew and roof layers is reliable and secure.

DeW-LoMT: performance evaluation

The performance evaluation is based on simulation and experimental evaluation. In the simulation, we have considered the *iFogSim* simulator⁵⁶ to set up the roof and the fog environment. Table 3 presents the simulation parameters considered in the *iFogSim* simulator as part of simulation analysis.

Simulation analysis

We have divided the simulation analysis into two parts, the evaluation of the roof layer and the fog layer, as they extend support to the dew layer nearby. The details are presented as follows.

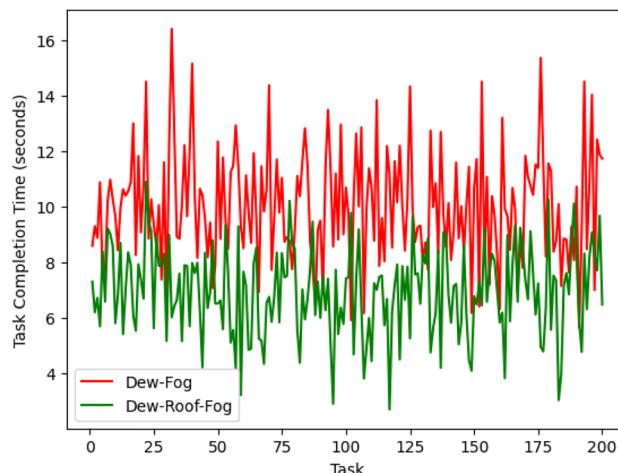


Fig. 6. Analysis of task offloading at roof layer.

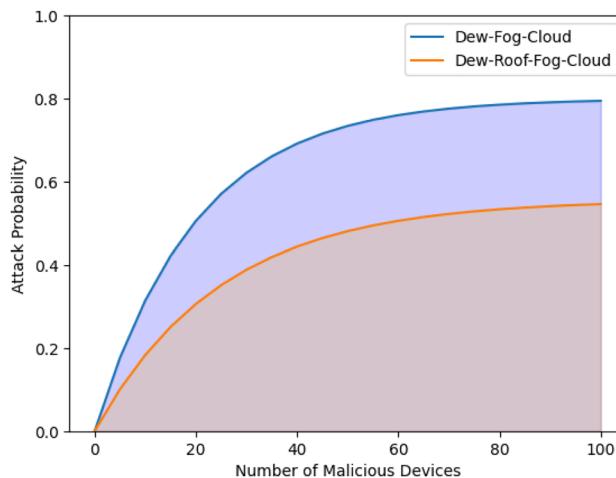


Fig. 7. Attack probability analysis.

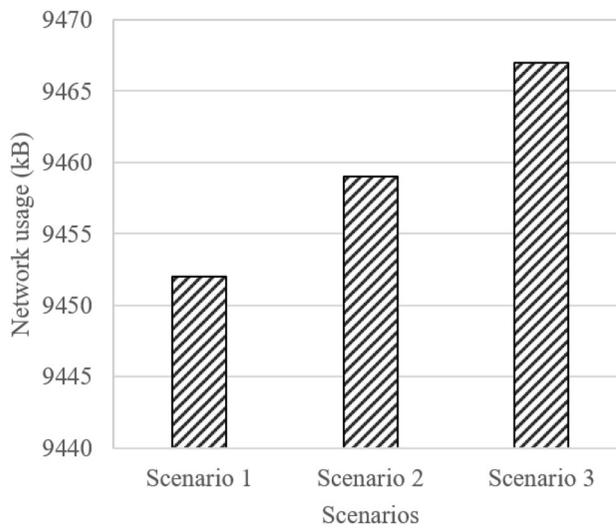


Fig. 8. Network usage of proposed paradigm.

Roof layer analysis

In this subsection, we validate the benefits of including the roof layer in the dew-for computing paradigm. As suggested, the roof layer solves the immediate problem of offloading memory-intensive tasks from the dew to itself, thereby reducing communication latency. We enable the offloading mechanism on the Raspberry Pi (roof node) in the proposed scheme. It allows the roof node to take memory and bandwidth-intensive tasks from the dew layer and perform them. In case of resource requirements, it requests services from the fog controller node. This results in improved performance.

Table 3 lists configuration parameters that primarily vary the uplink and downlink bandwidth, our simulation scenario in iFogSim is designed to isolate the impact of network bandwidth on overall performance while keeping other parameters constant. In our implementation, the roof layer is modeled as a dedicated intermediary node that emulates the characteristics of a Raspberry Pi device. Specifically, the roof node is configured with processing power, memory, and encryption overhead parameters based on real-world specifications (e.g., a 1.5 GHz quad-core ARM Cortex-A72 CPU and 4 GB of RAM). Its communication interface is simulated using IEEE 802.11n/802.11ax standards, ensuring that both its wireless connectivity and security features (such as TLS encryption) are accurately represented. This modeling allows the roof layer to aggregate data from the dew devices and perform preliminary processing before forwarding tasks to the fog layer.

Figure 6 presents a comparative analysis of task completion time. In case the dew-fog architecture does not have the roof layer, the average task completion time is ≈ 10 seconds with a standard deviation of ≈ 2 seconds. The dew-roof-fog architecture has an average task completion time of ≈ 7 seconds with a standard deviation of ≈ 1.5 seconds. Thus, the task completion time in the dew-roof-fog significantly improves by 27.89% against the dew-fog architecture. This is mainly because most of the tasks are completed near dew (roof), which reduces the turnaround time.

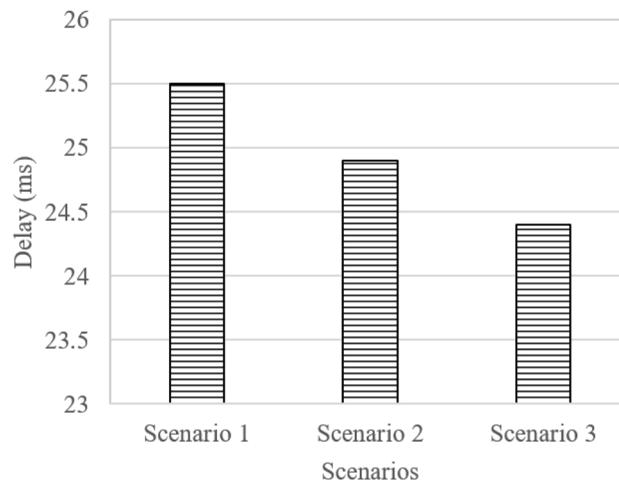


Fig. 9. Delay of proposed paradigm.

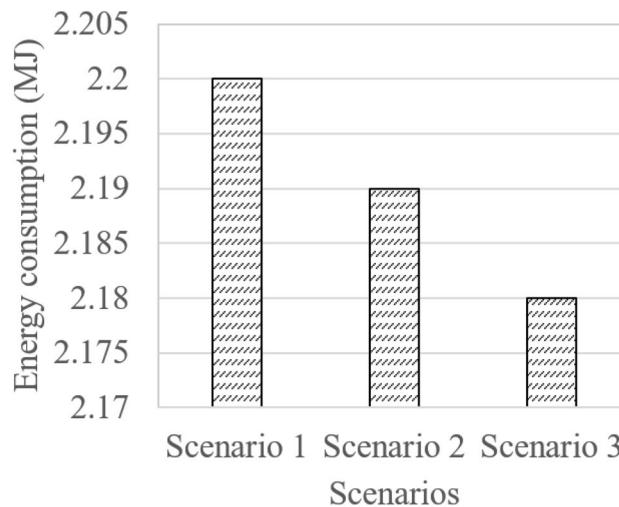


Fig. 10. Energy consumption of proposed paradigm.

Parameters	Value
Sensor delay	5 ms
Latency (Dew Uplink)	2 ms
Latency (Dew Downlink)	1 ms
Latency (Fog)	10 ms
Latency (Cloud)	100 ms
Data processing latency	10 ms
Energy dissipation (per sensor)	2 J
Fog/Dew Nodes	3
RAM (Dew)	1 MB

Table 4. Experimental parameters table.

Next, we present the mitigation of security attacks in the framework. We measure the attack probability A_p of the dew-fog paradigm against the dew-roof-fog computing. A_p measures the likelihood that the system (or network) would be compromised. We fix the number of malicious nodes n in the system for both architectures and measure the impact as the malicious nodes increase. We consider an exponential decay function $\alpha * (1 - e^{\beta n})$ to compute A_p , where α is a constant that represents the threshold of maximum attack probability when there

are no compromised devices in the system, and β represents the attack deviation factor of the previous round, which is denoted as follows.

$$\beta = \frac{A_{p-1} - N_{p-1}}{N_{p-1}} \quad (5)$$

where A_{p-1} and N_{p-1} are the attack and normal probabilities of the previous round $r - 1$.

Figure 7 shows the plot. From the plot, we can observe that the attack probability for both architectures increases as the number of malicious devices increases. However, the dew-roof-fog-cloud architecture has a A_p compared to the dew-fog-cloud architecture for the same number of malicious devices. This implies that including the roof layer in the architecture provides better security against malicious attacks.

Simulation scenarios for iFogSim

The proposed paradigm has been simulated in iFogSim, and the performance is analyzed regarding network usage and delay. Here, we have considered three scenarios depending on the uplink and downlink bandwidth. The uplink bandwidth considered in Scenario 1, Scenario 2, and Scenario 3 is 90 Mbps, 100 Mbps, and 110 Mbps, respectively. The downlink bandwidth considered in Scenario 1, Scenario 2, and Scenario 3 is 140 Mbps, 150 Mbps, and 160 Mbps, respectively. The other parameters' values⁵⁷ are provided in Table 3. The cloud server has higher resources than the fog node, so the cloud VM's RAM size and CPU length are higher than the fog node.

Figure 8 presents the network usage of the proposed paradigm concerning the three scenarios. Network usage refers to the utilization of the system resources in terms of data transmission and reception from the network interfaces, measured in kilobytes (kB). This is observed that the network usage of the proposed paradigm is 9450–9470 kB.

Figure 9 presents the delay of the proposed paradigm concerning the three scenarios. The delay refers to the execution delay of the application, measured in milliseconds (ms). This is observed that the delay of the proposed paradigm is 24–25.5 ms.

Figure 10 presents the energy consumption of the proposed paradigm concerning the three scenarios. The energy consumption is determined by the power consumption of all hosts in a particular time frame and is measured in Megajoule (MJ). In our approach, the communication between cloud-fog requires an energy of \approx 2.18–2.2 MJ.

Experimental analysis

The section presents the details of the performance metrics, the experimental setup, and the results of the scheme. At the dew layer, we have installed appropriate packages to receive real-time health indicators and have used the Table 4 presents the parameters considered in the experiment.

The parameter values are taken by Medhi et al.⁵⁸. We have considered that maximum of three dew nodes are connected to three fog nodes, respectively, and the experiment is conducted 20 times, and the reading is averaged. The uplink bandwidth of dew, fog, and cloud nodes is kept at 50 Mbps, 100 Mbps, and 100 Mbps, respectively. Network parameters like response time, energy dissipation, and bandwidth are the performance metrics in the experiment.

Performance metrics

To analyze the performance of *DeW-IoMT*, response time, energy dissipation, and network bandwidth are considered. As per the simulation table, we have fixed the network bandwidth. Hence, we discuss the effects of response time and energy dissipation only.

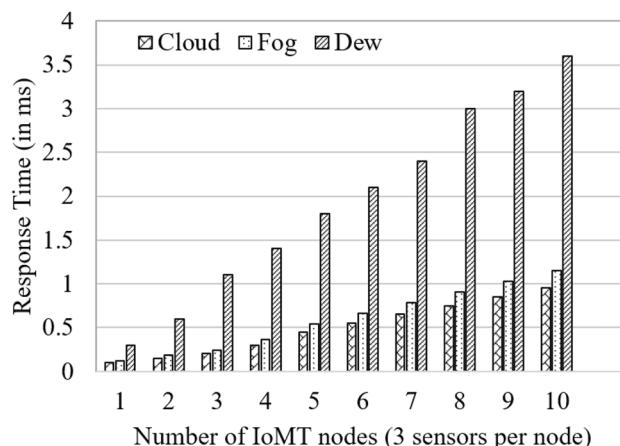
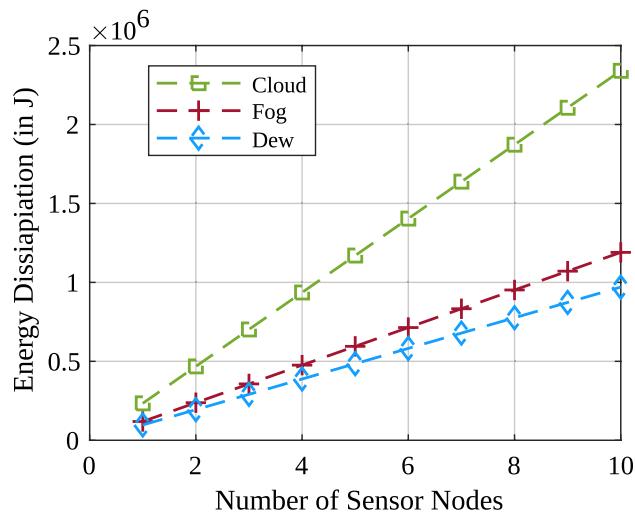
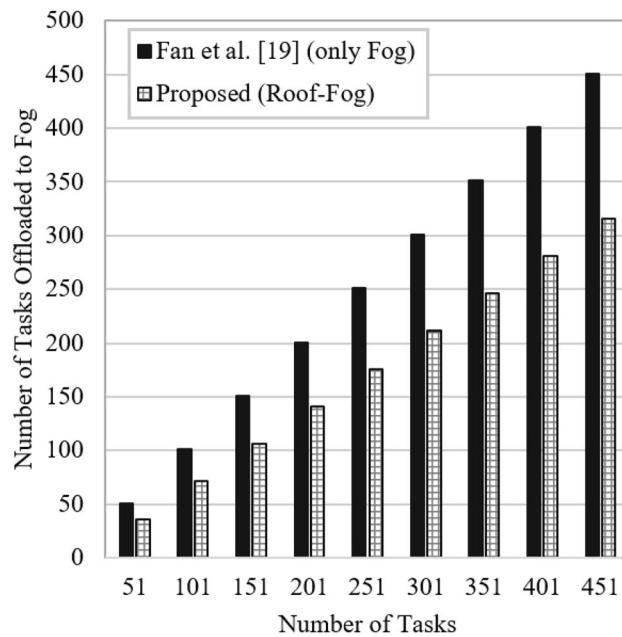


Fig. 11. Response time analysis.

**Fig. 12.** Energy dissipation analysis.**Fig. 13.** Task size offloaded by Fog..

At the fog layer, recommendation analytics L_{rem} is sent on the data d . We consider two links, once between the fog and cloud, with data transmission rate R_{fc} , and between the fog and dew through the roof, where we present transmission rate as R_{fd} . The failure rates are denoted as f_{fc} and f_{fd} . Thus, the latency in analytics from the cloud layer is denoted as follows.

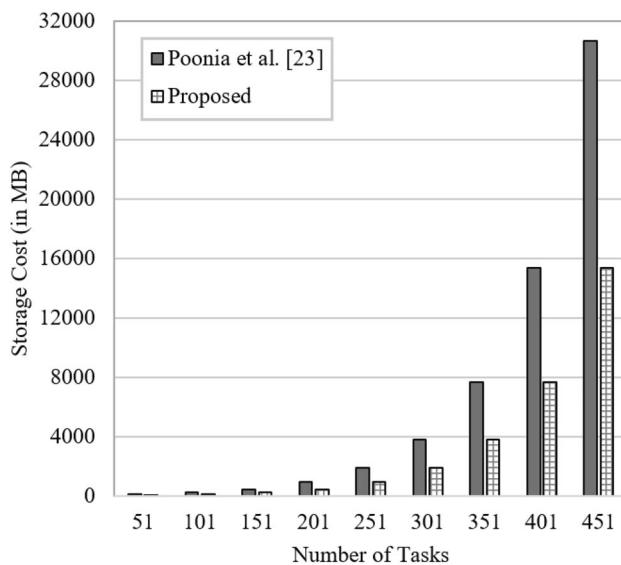
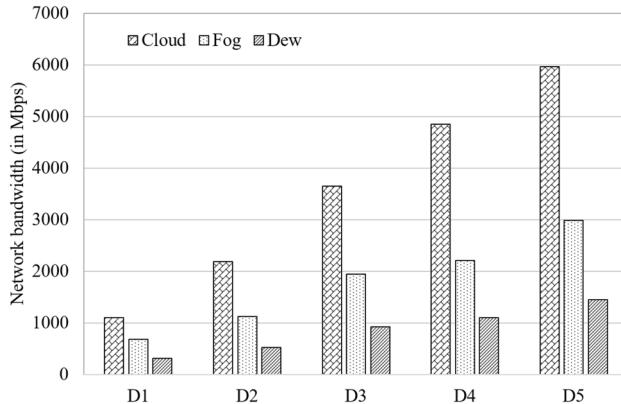
$$L_{cf} = (d/R_{fa})(1 + f_{fc}) + (d/R_{fd})(1 + f_{fd}) \quad (6)$$

From the fog layer, the latency would be as follows.

$$L_{fd} = (d/R_{fd})(1 + f_{fd}) \quad (7)$$

with the trivial condition $L_{fd} < L_{fc}$. In the proposed scheme, the dew layer can store cache updates in case of connectivity failure. It is noted that with the dew layer, the latency reduces by $\approx 17\text{--}25\%$ in comparison with fog-cloud architectures.

For power analysis, we consider a scenario where we assume that the dew node receives data from k IoT nodes, denoted as D_n . The total data packets from i^{th} device is Pw_i , where $1 \leq N_i \leq \sum_{i=1}^k N_k$. In terms of

**Fig. 14.** Storage Cost at Cloud.**Fig. 15.** Network Bandwidth analysis.

power analysis, the power is divided into data packet accumulation, analysis, and transmission requirements. The power consumption of any dew node receiving (accumulating) a packet Pw_k from node D_n is

$$Pw_a = \sum_{l=1}^k Pw_l \quad (8)$$

where Pw_l is the power dissipation of a single node. Similarly, the power dissipation formula can be used for analysis and transmission (denoted as Pw_{ana} and Pw_{trans} respectively). The total power at the dew node is shown as follows.

$$Pw_{total} = Pw_a + Pw_{ana} + Pw_{trans} + Pw_{dis} \quad (9)$$

where Pw_{dis} denotes the power consumption required in the event of disconnection. It is noted that the total power consumption also decreases by $\approx 35\%$ from the cloud-based solutions and $\approx 15\%$ from the fog-edge-based solutions.

Experimental results

The experimental with analytical results are compared for the parameters against cloud-based⁵⁹ and fog-edge-based solution⁶⁰.

Figure 11 shows the response time computation against patient IoT setups (we consider three sensors attached to the patient's body) for each node. Based on R_{fc} and R_{fd} , L_{cf} and L_{fd} is computed. For 5 node setups, recommendation response from the cloud takes ≈ 1.95 seconds (s), from fog 0.6 s, and dew layer, it takes

Metric	Baseline (Mean \pm SD)	Proposed (Mean \pm SD)	p-value (t-test)	ANOVA F-value
Response Time (s)	10.00 \pm 2.00	7.00 \pm 1.50	0.012	5.32
Throughput (Mbps)	50.0 \pm 5.0	65.0 \pm 4.0	0.008	6.45
Energy Consumption (kJ)	1470 \pm 100	1020 \pm 80	0.005	7.10

Table 5. Statistical analysis of key performance metrics.

0.495 s. Thus, there is a significant improvement of $\approx 74.61\%$ from the cloud and $\approx 17.5\%$ from fog computation at the dew layer for 5 IoMT node setups.

Figure 12 compares dew, fog, and cloud layers concerning energy dissipation. As evident, we observe the power requirements as depicted in equation (9). Compared to fog and edge layers, we observe an increase in power consumption at the cloud layer. At 6 node setups, the energy dissipation at the cloud is 1470 kJ, compared to 670 kJ at the fog, and 570kJ at dew, which shows an improvement of 38.78% from the cloud and 14.92% from the fog layer.

We further analyze the number of tasks offloaded to the fog node. A comparative analysis is drawn against Fan et al.³², which considers the task t offloaded to fog. Fig. 13 depicts the results. As depicted in section “Resource offloading: dew to roof”, our approach considers S_i as the task size offloaded to the roof, with the condition $T_o < t$. We consider the percentage of data sent to the cloud in both approaches, with the underlying condition that data generated for each task is constant ($Cons$). Thus, the amount of data sent to the cloud D_c can be approximated as a linear function against several tasks $N(t)$. The same is presented as follows.

$$D_c = Cons \times N(t) \quad (10)$$

As indicated, the percentage of data is computed as follows.

$$D_c = 100 - \frac{T_l}{T_{total}} \quad (11)$$

where $\frac{T_o}{T_{total}}$ denotes the percentage of tasks offloaded to any layer. In our approach, we see that an improvement of $\approx 29.43\%$ is achieved against the approach presented in³², as due to the roof layer, the number of tasks sent to the cloud reduces, and thus D_c is reduced.

Next, we analyze the storage cost of the proposed roof-fog scheme against Poonia et al.²⁸. In our scheme, most of the task is resolved at the roof-fog layer. Hence, a small portion of the task is forwarded to the cloud. Thus, it reduces the size of task execution in the cloud. Therefore, there is a reduction in the storage cost in the cloud. Fig. 14 presents the results. With 351 tasks, the storage cost at the cloud is 7986 MB in²⁸, compared to 3987 MB in our proposed approach. On average, a significant reduction of $\approx 33.56\%$ is obtained on task data stored in the cloud node. Thus, our proposed scheme provides a more efficient, cost-effective, and scalable solution to offload IoMT tasks, leveraging applications’ processing power.

Figure 15 shows the network bandwidth consumption for 5IoMT setups, numbered as $\{D_1, D_2, D_3, D_4, D_5\}$. For three nodes, cloud bandwidth is 3552 Mbps, 1788 Mbps for fog, and 861 Mbps for dew nodes.

Statistical analysis of performance metrics

To ensure the robustness of our experimental results, we conducted statistical hypothesis testing on key performance metrics—namely, response time, throughput, and energy consumption. Our simulation experiments were repeated 20 times to capture variability, and the mean values along with the standard deviations were computed for both the baseline (traditional dew-fog or cloud-centric architectures) and the proposed DeW-IoMT framework (dew-roof-fog-cloud architecture). A two-sample t-test was then applied to each metric to compare the means between the two setups, with a significance level of $\alpha = 0.05$. Additionally, a one-way ANOVA was performed to evaluate the effect of varying network conditions on these metrics. The resulting p-values and F-values confirmed that the improvements observed using the proposed framework are statistically significant.

Table 5 summarizes the statistical analysis. For instance, the average response time in the baseline architecture was found to be 10.00 seconds (± 2.00 s), while the proposed approach achieved an average response time of 7.00 seconds (± 1.50 s) with a p-value of 0.012 and an ANOVA F-value of 5.32. Similarly, throughput improved from 50.0 Mbps (± 5.0 Mbps) to 65.0 Mbps (± 4.0 Mbps) (p-value 0.008, F-value 6.45), and energy consumption decreased from 1470 kJ (± 100 kJ) to 1020 kJ (± 80 kJ) (p-value 0.005, F-value 7.10). These results demonstrate that the enhancements in the proposed DeW-IoMT framework are not due to chance, but are statistically significant.

These statistical results affirm that the proposed DeW-IoMT framework significantly reduces response time and energy consumption while improving throughput, thereby substantiating its effectiveness and scalability for IoMT applications.

Formal security analysis

In section “Security and access control”, we highlighted the security attacks possible in the framework, namely, data interception, data tampering, and DoS attacks. We counter these attacks at the *RL* layer, where the encryption function E encrypts data sent by sensor nodes via the AES algorithm. Rule-based access control

Parameter	25	28	29	Proposed
A1	Y	N	Y	Y
A2	Y	Y	Y	Y
A3	N	Y	N	Y
A4	N	Y	Y	Y

Table 6. A comparative analysis of security functions against related schemes. Attacks A1: Interception; A2: Tampering (Alteration); A3: DoS; A4: Unauthorized Access Y shows parameter is present; N shows parameter is absent.

maps only authorized users to predefined roles⁶¹. This section presents the formal proofs against the attacks to validate the claims.

Theorem 1 *The proposed dew-roof-fog-cloud framework is secure against data interception attacks.*

Proof In this attack, an adversary intercepts and accesses the data transmitted over the dew and roof layers. Let D be the data collected from the sensors at the dew layer, and $E(D)$ be the encrypted data using the AES encryption scheme with key K . Let M be the set of all possible messages an adversary can intercept, and C be the set of all possible ciphertexts an adversary can obtain. In such cases, the probability of message m getting intercepted from c is negligible and thus can be ignored. By the definition of AES encryption, we have $E(D) = AES(D, k)$. This means that the ciphertext c is a deterministic function of the message m and the key K , which means $c = E(m, K)$. Thus, the probability function can be written as follows $Pr[M = m|C = c] = Pr[D = d|E(D) = c] = Pr[AES^{-1}(c, K) = d]$. Since AES is a secure encryption algorithm, the probability that an adversary can invert the ciphertext to recover the plaintext is negligible. \square

Theorem 2 *The proposed dew-roof-fog-cloud framework is secure against data tampering attacks.*

Proof In this attack, we consider D and $E(D)$ to be respectively the original and encrypted data. An adversary tampers the data, represented as D' , and $E(D')$ is the encryption of the tampered data. For the same, the adversary performs an interception attack on $E(D)$ and modifies it to get $E(D')$. As AES is a secure encryption algorithm, it is computationally infeasible to obtain D' from $E(D')$ without the encryption key K . Even if the adversary has the encryption key K , it cannot modify the encrypted data $E(D)$ to get $E(D')$ without being detected by the integrity checks in place. Thus, the system is secure against the data tampering attack. \square

Theorem 3 *The proposed dew-roof-fog-cloud framework is secure against DoS attacks.*

Proof In this case, we consider that R_r be the available resources at the roof layer and T_r be the tasks needed. Let $f(T)$ be the resource requirement of a task T , and let $Prc(T)$ be the processing time of a task T . Suppose an attacker tries to launch a DoS attack by flooding the system with a large number of resource requests. However, since the attacker does not have the necessary privileges (owing to role-based access control), their requests will be denied, and the attacker will not be able to consume any resources. Formally, let U_a be the set of users the attacker can impersonate, and let Rl_a be the set of roles the attacker can assume. Let M_a be the access matrix for the attacker, where $M_{a,i,j}$ represents the set of privileges that the attacker has for role j when they assume the identity of user i . Since the attacker does not have the necessary privileges, $M_{a,i,j}$ would be empty for all $i \in U_a$ and $j \in R_a$. Therefore, the attacker cannot perform tasks on the roof layer, and their resource requests will be denied. Thus, the system is secure against DoS attacks. \square

Theorem 4 *The proposed dew-roof-fog-cloud framework is secure against unauthorized access attacks.*

Proof In this attack, we assume the malicious entity attempts to steal credentials by forging (impersonating) another user to gain unauthorized access and perform unwanted actions. In our approach, we have employed the RBAC to manage user privileges. We consider that let U_a be the set of unauthorized users and R_a be the set of unauthorized roles that the attacker can assume. In such case, a role-matrix M_a is the access matrix of the attacker, and let $M_{a,i,j}$ represent the set of privileges that the attacker has for role j when assuming the identity of user i .

As RBAC is implemented, locks L on updating the matrix are allowed to only authorized users. Thus, the access matrix updation to M'_a would return empty for unauthorized users, and thus no data or resource can be manipulated in our approach. Formally, for any unauthorized user $u \in U_a$ and unauthorized role $r \in R_a$, $M_{a,u,r} = \emptyset$. Therefore, the attacker cannot perform unauthorized actions within the system, ensuring the framework's security against unauthorized access attacks. \square

Security cost analysis

In this section, we analyze the proposed framework's communication and security computation cost. Our framework considers that the roof layer allows AES encryption, which involves a symmetric encryption/

decryption process. As indicated in Srinivas et al.⁶², the security cost of symmetric/encryption-decryption is 0.0056 seconds. In our framework, the task t is sent from dew to roof, involving 1 round of encryption and corresponding decryption. Similarly, between roof fog, we would have 1 more round, followed by a final round between fog-cloud. Let us denote the encryption operation as E_{op} and the decryption operation as E_{dop} , then the security computation cost is $3E_{op} + 3E_{dop}$, which is 0.0336 seconds. We compare the proposed scheme against Mukherjee et al.²⁹, which uses asymmetric encryption for node communication. In the E_{op} , an additional operation of computing modular exponentiation M_{exp} is required, which costs 0.0192 seconds. Modular inverse M_{inv} is required at the receiver, which is ≈ 0.00264 seconds. Thus, the overall cost is $E_{op} + M_{exp} + E_{dop} + M_{inv}$, which is 0.05544 seconds, and thus an improvement of 65% is obtained in security cost.

For security communication cost, we consider the message exchanges between different entities as depicted in Fig. 2. The IoMT sensor nodes share health data with a timestamp value of 32 bits. The data is then passed to the dew (Arduino Uno), which forwards it to the roof layer with a random nonce value of 32 bits. Finally, the data is passed over the IEEE 802.11 network, where the associated node attaches a digital signature verified at the receiver end. For real-time response, we use a MAC value of 160 bits. Thus, the overall communication cost is $C_{sn} + C_{dew} + C_{roof} + C_{dew} + C_{cloud}$, where all these operations are performed, which is $32 + 32 + (32 + 160) + (32 + 160) + (32 + 160)$ bits, which is 640 bits. We compare the proposed communication cost with authors in^{25,28}, and we obtain an improvement of 128 bits over these schemes, as they perform the random nonce operation and timestamp both in the communication, which amounts to 64 bits at each link. Thus, an additional overhead of 96 bits is obtained between the dew-fog and fog-cloud systems. Thus, our framework demonstrates an improvement of 15% in terms of communication cost.

Table 6 shows a comparative analysis of performed and validated security functions in the proposed schemes against existing schemes.

Limitations of the proposed framework

The proposed framework *DeW-IoMT* outlines the integration of the dew layer in IoMT setups for critical response and actions. The roof layer ably supports the dew layer, and the roof-fog paradigm provides resource management for the dew. Only high-compute intensive tasks would go to the cloud for analysis (bulky models being trained on a high amount of data). However, there are inherent open challenges in terms of networking, resource management, and security and privacy in the scheme, which limits its effectiveness in real-world setups.

Firstly, the proposed scheme requires a high amount of changes to the existing centralized (fog/edge and cloud infrastructures) in terms of daily operations and control to integrate the dew component. This conversion might be a costly affair and would be time-consuming. Thus, it increases the overall deployment cost of the scheme in highly scalable and dynamic environments and might require extensive training operations. Moreover, the current evaluation is based on simulation, and additional real-world experiments are necessary to capture the full complexity of operational environments. To solve this problem, further research is focused on designing loosely coupled fog/edge and cloud computing paradigms, where the services are handled close to the user level. Moreover, with the rise of robust wireless communication and networking protocols, the networks can handle intermittent connectivity or frequent disconnections. Another direction is the shift towards decentralized peer-to-peer networking, which allows more effective data transmission and processing between the layers.

Secondly, we present a resource offloading approach at the roof layer to support the dew operations. Such a model might not be feasible for all types of tasks. Modern systems have a set of heterogeneous medical data, which requires a differentiated viewpoint to handle the data. Viewing the entire data as a homogeneous task component might not scale well to diverse applications in such cases. Thus, it is imperative to design task evaluation and classification strategies that allow the user to understand the task semantics and decide which task to offload to the roof layer. In future work, we plan to develop dynamic task classification algorithms, possibly using explainable AI techniques, to further refine the offloading decisions based on real-time contextual data. Thus, the future scope is to address the challenges of resource constraints and design efficient algorithms to manage tasks and resource allocation. For the same, explainable AI-based techniques are used to model resource interpretation and usage patterns. The explainable results are then fed to the machine learning model to dynamically allocate resources based on the current and the projected demand in the near future.

Finally, the scheme is heavily skewed toward the security and privacy mechanism focusing on symmetric encryption and role-based access control. However, owing to the vast nature of zero-day attacks, the vulnerability space has increased, and thus it is important to design more resilient security and privacy mechanisms. Future research should consider the integration of advanced cryptographic techniques such as homomorphic encryption and secure multi-party computation to further safeguard sensitive medical data. Thus, advanced techniques, such as homomorphic encryption and secure multi-party computation, can be performed, allowing the computations on encrypted data without decryption. These algorithms also assure the privacy of sensitive patient attributes during data sharing among multiple nodes in operation.

Conclusions

The paper presents the *DeW-IoMT* framework that leverages a dew layer to quickly detect critical health issues by monitoring patient pulse using an Arduino Uno and LCD-even during disconnections. A connected roof layer enhances resilience by securely offloading data, reducing reliance on fog or cloud processing for urgent alerts, while continuous cloud monitoring provides long-term analytics. Simulation and experimental evaluations show significant gains in response time, energy efficiency, and network utilization compared to cloud-centric models, with the roof layer particularly reducing latency and bolstering security.

Future work aims to integrate additional sensors (e.g., temperature and ECG) for on-dew analysis, refine task offloading strategies between fog and cloud layers, and explore improved resource management and advanced security mechanisms to enhance framework adaptability in diverse, real-world scenarios.

Data availability

The datasets generated and/or analysed during the current study are available in the Zenodo repository (<https://doi.org/10.5281/zenodo.14500257>).

Received: 23 November 2024; Accepted: 28 May 2025

Published online: 02 July 2025

References

1. Patel, F., Bhattacharya, P., Tanwar, S., Gupta, R., Kumar, N., & Guizani, M. Block6Tel: Blockchain-based spectrum allocation scheme in 6g-envisioned communications. In: 2021 International Wireless Communications and Mobile Computing (IWCMC), Harbin City, China, 1823–1828 (2021). <https://doi.org/10.1109/IWCMC51323.2021.9498854>
2. Harasees, A., Al-Ahmad, B., Alsobeh, A., & Abuhussein, A. A secure IoT framework for remote health monitoring using fog computing. In *2024 International Conference on Intelligent Computing, Communication, Networking and Services (ICCNS)* 17–24 (2024). <https://doi.org/10.1109/ICCNS62192.2024.10776425>
3. Disposable Medical Device Sensors Market Size by Acumen Research and Consulting. <https://www.globenewswire.com/en/news-release/2022/10/19/2537457/0/en/Disposable-Medical-Device-Sensors-Market-Size.html>. Accessed: 2022-12-07
4. Bhattacharya, P., Obaidat, M. S., Savaliya, D., Sanghavi, S., Tanwar, S., & Sadaun, B. Metaverse assisted telesurgery in healthcare 5.0: An interplay of blockchain and explainable AI. In *2022 International Conference on Computer, Information and Telecommunication Systems (CITS), Piraeus, Greece* 1–5 (2022). <https://doi.org/10.1109/CITS55221.2022.9832978>
5. Zhao, X., Zhang, Y., Yang, Y. & Pan, J. Diabetes-related avoidable hospitalisations and its relationship with primary healthcare resourcing in china: A cross-sectional study from sichuan province. *Health Soc. Care Commun.* **30**(4), 1143–1156 (2022).
6. Li, Q., You, T., Chen, J., Zhang, Y. & Du, C. LI-EMRSQL: linking information enhanced Text2SQL parsing on complex electronic medical records. *IEEE Trans. Reliab.* **73**(2), 1280–1290 (2023).
7. Saraswat, D. et al. Explainable AI for healthcare 5.0: Opportunities and challenges. *IEEE Access* **10**, 84486–84517. <https://doi.org/10.1109/ACCESS.2022.3197671> (2022).
8. Mohanta, B., Das, P., & Patnaik, S. Healthcare 5.0: A paradigm shift in digital healthcare system using artificial intelligence, IoT and 5G communication. In: *2019 International Conference on Applied Machine Learning (ICAML)* 191–196 (2019). <https://doi.org/10.1109/ICAML48257.2019.00044>
9. Deb, P., Mukherjee, A. & De, D. Chapter 21 - mobile health monitoring for senior citizens using femtolet-based fog network. In *Contemporary Medical Biotechnology Research for Human Health Advances in Biotechnology and Bioengineering* (eds Joshi, S. et al.) 197–204 (Academic Press, Cambridge, 2022). <https://doi.org/10.1016/B978-0-323-91251-8.00021-0>
10. Engineer, M., Tusha, R., Shah, A., & Adhvaryu, D. K. Insight into the importance of fog computing in internet of medical things (IOMT). In *2019 International Conference on Recent Advances in Energy-efficient Computing and Communication (ICRAECC), Nagercoil, India* 1–7 (2019). <https://doi.org/10.1109/ICRAECC43874.2019.8994985>
11. Graham, R. & Sloane, N. On the covering radius of codes. *IEEE Trans. Inf. Theory* **31**(3), 385–401. <https://doi.org/10.1109/TIT.1985.1057039> (1985).
12. Hati, S., De, D. & Mukherjee, A. DewBCity: Blockchain network-based dew-cloud modeling for distributed and decentralized smart cities. *J. Supercomput.* **78**(6), 8977–8997 (2022).
13. Navya, C., Salvi, S., Jacob, N. D., & Kumar, S. A roof computing architecture based indoor positioning system for IoT applications. In *2020 Fourth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*, 19–24 (2020). <https://doi.org/10.1109/I-SMAC49090.2020.9243580>
14. Ahammad, I., Khan, M. A. R. & Salehin, Z. U. Software-defined dew, roof, fog and cloud (SD-DRFC) framework for IoT ecosystem: The journey, novel framework architecture, simulation, and use cases. *SN Comput. Sci.* **2**(3), 159 (2021).
15. European industrial technology roadmap for the next generation cloud-edge offering. https://ec.europa.eu/newsroom/repository/document/2021-18/European_CloudEdge_Technology_Investment_Roadmap_for_publication_pMdz85DSw6nqPppq8hE9S9RbB8_76223.pdf. Accessed: 2023-04-23
16. Ray, P. P. An introduction to dew computing: Definition, concept and implications. *IEEE Access* **6**, 723–737. <https://doi.org/10.1109/ACCESS.2017.2775042> (2018).
17. Guberović, E., Lipić, T., & Čavruk, I. Dew intelligence: Federated learning perspective. In *2021 IEEE 45th Annual Computers, Software, and Applications Conference (COMPSAC), Madrid, Spain*, 1819–1824 (2021). <https://doi.org/10.1109/COMPSAC51774.2021.000274>
18. Marozzo, F., Orsino, A., Talia, D., & Trunfio, P. Edge computing solutions for distributed machine learning. In *2022 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech), Falerna, Italy*, 1–8 (2022). <https://doi.org/10.1109/DASC/PiCom/CBDCom/Cy55231.2022.9927824>
19. Verma, A., Bhattacharya, P., Budhiraja, I., Gupta, A. K. & Tanwar, S. Fusion of federated learning and 6g in internet-of-medical-things: Architecture, case study and emerging directions. In *Futuristic Trends in Networks and Computing Technologies* (eds Singh, P. K. et al.) 229–242 (Springer, Singapore, 2022).
20. Kumari, A., Bhattacharya, P., Verma, A., Zhu, Z. & Gadekallu, T. R. Federated edge computing for edge-assisted consumer electronics. *IEEE Consum. Electron. Mag.* <https://doi.org/10.1109/MCE.2024.3503069> (2024).
21. Zschörnig, T., Windolph, J., Wehlitz, R., Dumont, Y. & Franczyk, B. A fog-based multi-purpose internet of things analytics platform. *SN Comput. Sci.* **3**(3), 1–20 (2022).
22. Vinu, S. & Diwan, B. Integrating blockchain and IoT using dew computing and light weight election based consensus for smart city. In *AIP Conference Proceedings*, Vol. 2802(1), 090003 (2024). <https://doi.org/10.1063/5.0182419>
23. Zhao, L. et al. Intelligent caching for vehicular dew computing in poor network connectivity environments. *ACM Trans. Embed. Comput. Syst.* **23**(2), 1–24. <https://doi.org/10.1145/3643038> (2024).
24. Ghosh, S., & De, D. DewGame: D2D communication enabled dew computing for 5G IoT using coalition formation game. *J. Supercomput.* 1–38 (2023).
25. Karmakar, A., Banerjee, P. S., De, D., Bandyopadhyay, S. & Ghosh, P. MedGini: Gini index based sustainable health monitoring system using dew computing. *Med. Novel Technol. Devices* **16**, 100145. <https://doi.org/10.1016/j.medntd.2022.100145> (2022).
26. Podder, T., Bhattacharya, D. & Majumdar, A. Dew computing-inspired mental health monitoring system framework powered by a lightweight CNN. In *Disruptive Technol. Big Data Cloud Appl.* (eds Peter, J. D. et al.) 309–319 (Springer, Singapore, 2022).
27. Afaq, Y. & Manocha, A. Dew computing-assisted cognitive intelligence-inspired smart environment for diarrhea prediction. *Computing* **104**(11), 2511–2540 (2022).

28. Poonia, A. et al. CONFRONT: Cloud-fog-dew based monitoring framework for Covid-19 management. *Internet Things* **16**, 100459. <https://doi.org/10.1016/j.iot.2021.100459> (2021).
29. Mukherjee, A., De, D. & Dey, N. DewDrone: Dew computing for internet of drone things. *IEEE Consum. Electron. Mag.* <https://doi.org/10.1109/MCE.2021.3139306> (2021).
30. Baek, J. & Kaddoum, G. Heterogeneous task offloading and resource allocations via deep recurrent reinforcement learning in partial observable multifog networks. *IEEE Internet Things J.* **8**(2), 1041–1056. <https://doi.org/10.1109/JIOT.2020.3009540> (2021).
31. Zhou, Z., Liao, H., Gu, B., Mumtaz, S. & Rodriguez, J. Resource sharing and task offloading in IoT fog computing: A contract-learning approach. *IEEE Trans. Emerg. Top. Comput. Intell.* **4**(3), 227–240. <https://doi.org/10.1109/TETCI.2019.2902869> (2020).
32. Fan, N., Wang, X., Wang, D., Lan, Y., & Hou, J. A collaborative task offloading scheme in d2d-assisted fog computing networks. In *2020 IEEE Wireless Communications and Networking Conference (WCNC)*, Seoul, Korea (South), 1–6 (2020). <https://doi.org/10.1109/WCNC45663.2020.9120662>
33. Yang, Y. et al. POMT: Paired offloading of multiple tasks in heterogeneous fog networks. *IEEE Internet Things J.* **6**(5), 8658–8669. <https://doi.org/10.1109/JIOT.2019.2922324> (2019).
34. Lan, Y., Wang, X., Wang, D., Liu, Z. & Zhang, Y. Task caching, offloading, and resource allocation in d2d-aided fog computing networks. *IEEE Access* **7**, 104876–104891. <https://doi.org/10.1109/ACCESS.2019.2929075> (2019).
35. Li, Q., Zhao, J., Gong, Y. & Zhang, Q. Energy-efficient computation offloading and resource allocation in fog computing for internet of everything. *China Commun.* **16**(3), 32–41. <https://doi.org/10.12676/j.cc.2019.03.004> (2019).
36. Mutlag, A. A., Abd Ghani, M. K., Arunkumar, N., Mohammed, M. A. & Mohd, O. Enabling technologies for fog computing in healthcare IoT systems. *Futur. Gener. Comput. Syst.* **90**, 62–78. <https://doi.org/10.1016/j.future.2018.07.049> (2019).
37. Sodhro, A. H., Luo, Z., Sangaiah, A. K. & Baik, S. W. Mobile edge computing based QoS optimization in medical healthcare applications. *Int. J. Inf. Manag.* **45**, 308–318. <https://doi.org/10.1016/j.ijinfomgt.2018.08.004> (2019).
38. Ray, P. P., Dash, D. & De, D. Internet of things-based real-time model study on e-healthcare: Device, message service and dew computing. *Comput. Netw.* **149**, 226–239. <https://doi.org/10.1016/j.comnet.2018.12.006> (2019).
39. Rahmani, A. M. et al. Exploiting smart e-health gateways at the edge of healthcare internet-of-things: A fog computing approach. *Futur. Gener. Comput. Syst.* **78**, 641–658. <https://doi.org/10.1016/j.future.2017.02.014> (2018).
40. Ahmed, E. et al. Bringing computation closer toward the user network: Is edge computing the solution?. *IEEE Commun. Mag.* **55**(11), 138–144. <https://doi.org/10.1109/MCOM.2017.1700120> (2017).
41. Ghorbani, M., Ghobaei-Arani, M. & Asadolahpour-Karimi, R. Function placement approaches in serverless computing: A survey. *J. Syst. Architect.* **157**, 103291. <https://doi.org/10.1016/j.sysarc.2024.103291> (2024).
42. Ghorbani, M. & Ghobaei-Arani, M. Function offloading approaches in serverless computing: A survey. *Comput. Electr. Eng.* **120**, 109832. <https://doi.org/10.1016/j.compeleceng.2024.109832> (2024).
43. Jazayeri, F., Shahidinejad, A. & Ghobaei-Arani, M. A latency-aware and energy-efficient computation offloading in mobile fog computing: A hidden Markov model-based approach. *J. Supercomput.* **77**(5), 4887–4916 (2021).
44. Aghazadeh, R., Shahidinejad, A. & Ghobaei-Arani, M. Proactive content caching in edge computing environment: A review. *Softw. Pract. Exp.* **53**(3), 811–855. <https://doi.org/10.1002/spe.3033> (2023).
45. Jeyaraj, R., Balasubramaniam, A., Ma, A. K., Guizani, N. & Paul, A. Resource management in cloud and cloud-influenced technologies for internet of things applications. *ACM Comput. Surv.* **55**(12), 1–37. <https://doi.org/10.1145/3571729> (2023).
46. Gusev, M. Chapter 30—AI cardiologist at the edge: A use case of a dew computing heart monitoring solution. In *Artificial Intelligence and Machine Learning for EDGE Computing* (eds Pandey, R. et al.) 469–477 (Academic Press, Cambridge, 2022). <https://doi.org/10.1016/B978-0-12-824054-0.00020-4>.
47. Catalani, A., Chatzigiannakis, I., Anagnostopoulos, A., Akrivopoulou, G., Amaxilatis, D., & Antoniou, A. Hardware-assisted and deep-learning techniques for low-power detection of cardiovascular abnormalities in smart wearables. In *2021 IEEE International Conference on Smart Internet of Things (SmartIoT)*, Jeju, Republic of Korea 144–151 (2021). <https://doi.org/10.1109/SmartIoT5235.9.2021.00031>.
48. Wang, T., Tian, J., Fang, K., Gadekallu, T. R. & Wang, W. Ai and digital twin for consumer electronics in smart cities. *IEEE Consum. Electron. Mag.* <https://doi.org/10.1109/MCE.2024.3444312> (2024).
49. Otto, C., Milenković, A., Sanders, C. & Jovanov, E. System architecture of a wireless body area sensor network for ubiquitous health monitoring. *J. Mob. Multimed.* **1**(4), 307–326 (2005).
50. Gia, T. N., Thanigaivelan, N. K., Rahmani, A.-M., Westerlund, T., Liljeberg, P. & Tenhunen, H. Customizing 6lowpan networks towards internet-of-things based ubiquitous healthcare systems. In *2014 NORCHIP 1–6* (IEEE, 2014).
51. Want, R., Schilit, B. N. & Jenson, S. Enabling the internet of things. *Computer* **48**(1), 28–35 (2015).
52. Muhammed, T., Mahmood, R., Albeshri, A. & Katib, I. UbeHealth: A personalized ubiquitous cloud and edge-enabled networked healthcare system for smart cities. *IEEE Access* **6**, 32258–32285 (2018).
53. Majumdar, A., Laskar, N. M., Biswas, A., Sood, S. K. & Baishnab, K. L. Energy efficient e-healthcare framework using HWPSO-based clustering approach. *J. Intell. Fuzzy Syst.* **36**(5), 3957–3969 (2019).
54. Ahammad, I., Khan, A. R. & Salehin, Z. U. A review on cloud, fog, roof, and dew computing: IoT perspective. *Int. J. Cloud Appl. Comput. (IJCAC)* **11**(4), 14–41 (2021).
55. Ray, P. P., Dash, D. & De, D. Internet of things-based real-time model study on e-healthcare: Device, message service and dew computing. *Comput. Netw.* **149**, 226–239. <https://doi.org/10.1016/j.comnet.2018.12.006> (2019).
56. Gupta, H., Vahid Dastjerdi, A., Ghosh, S. K. & Buyya, R. iFogSim: A toolkit for modeling and simulation of resource management techniques in the internet of things, edge and fog computing environments. *Softw. Pract. Exp.* **47**(9), 1275–1296. <https://doi.org/10.1002/spe.2509> (2017).
57. Bera, S., Dey, T., Mukherjee, A., & Buyya, R. E-CropReco: a dew-edge-based multi-parametric crop recommendation framework for internet of agricultural things. *J. Supercomput.* **79**(11), 11965–11999 (2023).
58. Medhi, K., Ahmed, N. & Hussain, M. I. Dew-based offline computing architecture for healthcare IoT. *ICT Express* **8**(3), 371–378. <https://doi.org/10.1016/j.ictex.2021.09.005> (2022).
59. Ali, S. T., Sivaraman, V., & Ostry, D. Authentication of lossy data in body-sensor networks for cloud-based healthcare monitoring. *Fut. Gener. Comput. Syst.* **35**, 80–90. <https://doi.org/10.1016/j.future.2013.09.007> (2014). Special Section: Integration of Cloud Computing and Body Sensor Networks: Guest Editors: Giancarlo Fortino and Mukaddim Pathan
60. Omoniwa, B., Hussain, R., Javed, M. A., Bouk, S. H. & Malik, S. A. Fog/edge computing-based IoT (FECIoT): Architecture, applications, and research issues. *IEEE Internet Things J.* **6**(3), 4118–4149. <https://doi.org/10.1109/JIOT.2018.2875544> (2019).
61. Using the Compute Continuum for Data Analysis: Edge-cloud Integration for Urban Mobility. <https://scalab.dimes.unical.it/papers/pdf/edge-cloud-Marozzo-PDP2023.pdf>. Accessed: 2023-04-23
62. Srinivas, J., Das, A. K., Kumar, N. & Rodrigues, J. J. P. C. Cloud centric authentication for wearable healthcare monitoring system. *IEEE Trans. Dependable Secure Comput.* **17**(5), 942–956. <https://doi.org/10.1109/TDSC.2018.2828306> (2020).

Funding

No funding was received for conducting this study.

Declarations

Competing interests

The authors declare no competing interests.

Ethics approval and consent to participate

This article does not contain any studies with human participants or animals performed by any of the authors.

Additional information

Correspondence and requests for materials should be addressed to T.R.G. or Z.Z.

Reprints and permissions information is available at www.nature.com/reprints.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Open Access This article is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License, which permits any non-commercial use, sharing, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if you modified the licensed material. You do not have permission under this licence to share adapted material derived from this article or parts of it. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by-nc-nd/4.0/>.

© The Author(s) 2025