

A Fog Communication Framework with Priority Queueing for Time-Critical Health Alerts in IoT Healthcare

*Zaid Shakir Kadhim Al-Shammari

College of Engineering, Al-Karkh University of Science,
Haifa Street (Kawasan Qحماة / Hamada Palace Baghdad, Iraq.
zaid.shaker.elc@kus.edu.iq

Article history: Received August 01, 2026| Revised August 28, 2026| Accepted August 31, 2026

Abstract – In a cloud-only Internet of Medical Things (IoMT) system, every urgent health alert must cross the wide area network twice before a clinician sees it. Those two crossings are costly. Under moderate load they use up most of the clinical time budget, and under heavy load most alerts arrive too late to be useful. A fog communication framework is presented here that addresses the problem by moving the alert decision to a gateway near the patient and by giving alert traffic strict priority over routine telemetry. The priority scheme is not a new algorithm. It is the standard three band PfifoFast queue of the Linux traffic control layer, so it can be deployed today on ordinary hardware. A single analytical model joins a log-distance wireless link, a Cobham non-preemptive priority queue at the gateway, and a hard alert deadline, and is solved in closed form for end to end latency, deadline-meet ratio, and backhaul saving. Three independent evaluators are then used: the analytical model, a 25-seed discrete event simulation, and a packet-level NS-3 simulation of the IEEE 802.15.4 access tier running the same PfifoFast queue. Traffic comes from a real vital signs dataset of 200 020 records, whose measured risk balance sets the critical and routine traffic mix. At an offered load of 0.90 the proposed design cuts mean critical-alert latency by 84.0%, from 151.7 ms to 24.2 ms, and raises deadline compliance from 60.6% to 100% against a cloud-only FIFO baseline. A paired Wilcoxon test confirms the difference ($p = 5.96 \times 10^{-8}$). A factorial ablation shows that fog placement and priority scheduling help for different reasons, and that both are needed. Compared against Earliest Deadline First, which is optimal in theory, the simple stock queue reaches the same deadline reliability at far lower implementation cost. A transient test in which the critical share of traffic rises sixfold shows the guarantee still holds. Backhaul traffic falls by 95.3% at the same time.

Keywords: Fog Computing, Internet of Medical Things, priority queueing, deadline-aware scheduling, IEEE 802.15.4, NS-3, latency bounded alert delivery.



[Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.](https://creativecommons.org/licenses/by-nc-sa/4.0/)

I. INTRODUCTION

Wearable sensors have made continuous monitoring of vital signs routine [1], [2]. The IoMT

market reached USD 57.62 billion in 2023 and is projected to grow at 25.9% a year to 2030 [3]. But monitoring only helps if the network can carry an alert to a clinician in time. When a sensor detects that a patient is deteriorating, the reading it produces is not ordinary data. It is a message with a deadline. An alert that arrives late is, in clinical terms, an alert that did not arrive [4], [5].

Most commercial and academic health-IoT platforms send everything to a distant cloud. This is a poor fit for alerts. Every alert must travel to the cloud and the reply must travel back, so each one pays two wide area crossings. A single crossing is already tens of milliseconds under good conditions, and it grows quickly when the path is busy [6]. At moderate load those two crossings alone can consume the whole clinical budget. At heavy load most alerts miss the deadline. The problem is therefore about the network, not about detection. The question is not whether deterioration can be spotted. It is whether the alert can be delivered on time.

Two lines of earlier work address parts of this. The first moves the decision function from the cloud to a gateway close to the patient and reports large cuts in latency and bandwidth against a cloud-only baseline [7], [8], [9], [10], [11]. Recent reviews confirm these gains across latency, bandwidth, scalability, and energy [12], [13], [14]. What these studies do not do is model the network itself. They do not describe the wireless link, the queue discipline at the gateway, or how the two traffic classes compete for the same server. Their latency numbers are single points, without confidence intervals and without a load sweep [3], [15], [16]. The second line studies scheduling. It classifies tasks by urgency and assigns them to fog nodes, and it shows that serving urgent tasks first lowers latency and energy [17], [18], [19], [20]. That line, however, looks at which node serves a task rather than how packets are ordered inside a node. It also compares against other placement heuristics instead of against a scheduling optimum such as Earliest-Deadline First (EDF) [21]. So both ideas have

support. Their separate effects have never been measured apart from one another, and neither has been checked against a modern deadline-aware scheduler.

Three gaps follow. First, no study models the wireless link, the gateway queue, and the alert deadline together in one framework. Without that, a reported latency gain cannot be traced to placement, to discipline, or to the two acting together. Second, no study separates the effect of these two choices, so the value of each one is unknown. Third, the priority queues that real systems actually use, such as the Linux three-band PfifoFast queue [22], have never been tested against the theoretically optimal EDF scheduler in a fog healthcare setting. It is therefore an open question whether a simple stock queue is enough. A fourth and more general weakness is that most studies in this area report a single simulation run with no confidence intervals and no packet-level check [3], [15], [17].

Those gaps are addressed here. A three-tier IoT, fog, and cloud architecture is developed in which the alert decision runs locally at the gateway and alert traffic is kept separate from routine telemetry by a non-preemptive strict-priority queue. That queue is the stock three-band primitive of the Linux and SDN traffic-control stack [22], so nothing custom is needed to deploy it. One end-to-end model is derived that splits alert latency into access, propagation, transmission, queueing, processing, and notification terms, and closed-form critical-class delays are obtained from the Cobham priority result [23], [24]. Traffic is driven by a real vital-signs dataset of 200 020 records, whose measured risk balance fixes the critical and routine mix. Evaluation is then carried out on three independent evaluators: the analytical model, a 25-seed discrete-event simulation, and a packet-level NS-3 simulation [25] of the IEEE 802.15.4 access tier [26] running the same PfifoFast queue. A factorial ablation covers the four combinations of placement and discipline, and EDF [21] is added at both placements as a fifth and sixth comparison point.

The contributions are as follows. First, a joint analytical model of a three-tier IoMT network is presented, coupling a log-distance wireless link with a non-preemptive priority queue at the gateway and yielding closed-form latency, deadline, and backhaul expressions. Second, a real vital-signs dataset is converted into a traffic model with a measured class mix and a light triage rule, and both the proposed system and the baseline are specified as algorithms. Third, a three-evaluator campaign assesses the same load sweep with 95% confidence intervals and a paired significance test. Fourth, a factorial ablation separates placement from discipline, and an EDF benchmark shows that the stock priority queue matches the optimal scheduler on deadline compliance while placement supplies the large gain over the cloud baseline.

Nothing in this list is a new scheduling algorithm or a new protocol. What is new is the

combination and the way it is tested. The mathematics is new in that the wireless link, the priority queue, and the deadline are composed into a single closed-form expression for a fog-assisted IoMT path, alongside the matching cloud-only expression. The method is new in that the ablation separates placement from discipline on the same real traffic, and shows a stock Linux queue reaching the reliability of an optimal scheduler. The architecture is new in that the discipline is instantiated on standard traffic-control primitives, with no kernel work, no per-packet deadline tags, and no preemption, so the results are not confined to a simulated abstraction.

The rest of the paper is organised as follows. Section 2 reviews related work and positions this study against it. Section 3 develops the architecture and the communication model. Section 4 describes the dataset, the traffic model, and the experimental setup. Section 5 reports the results. Section 6 discusses them and states the limits of the study. Section 7 concludes.

II. RELATED WORKS

Three bodies of work are relevant here: fog and edge architectures for healthcare IoT, priority and deadline-aware scheduling for IoMT, and analytical or simulation modelling of latency for time-critical traffic.

Fog and edge placement for healthcare is well established. An early fog-assisted home monitoring system was reported in [8], and the promise and difficulty of fog-driven eHealth was set out in [9]. A three-layer architecture that applies threshold-based notification at the gateway and keeps long-term storage in the cloud appears in [15]. More recent work refines the pattern. A hybrid fog-edge architecture in [3] reports a 70% latency cut and a 60% bandwidth saving against a cloud-only baseline. An adaptive edge-fog network for emergency response is proposed in [27], and the operational role of fog in healthcare is surveyed in [1]. Several reviews collect the resulting quality-of-service gains and the open problems [28], [29], [30]. One feature is common to all of it. The network is treated as a black box. Latency gains are given as single numbers for one traffic profile, with no load sweep, no confidence intervals, and no model of the wireless link or the gateway queue. From these reports alone a gain cannot be attributed to placement, to scheduling, or to their interaction.

Scheduling in fog-based IoMT is the second thread, and it usually works at the level of task-to-node assignment rather than packet ordering inside a node. A framework that sorts tasks into critical, moderate, and normal classes and then assigns them to fog nodes by a weighted score balancing latency and energy is given in [17]. A priority-based scheme for edge health monitoring appears in [31], and federated learning is applied to secure and reliable IoMT scheduling in [16]. A semi-greedy deadline-aware and energy-efficient scheduler for fog systems is described in [32]. Work on SDN control is also relevant. Flow based fog nodes

are modelled as multi queue polling systems in [33], which reports an 80% improvement in the share of delay sensitive flows processed; SDN priority based execution is used to cut IoT-fog latency in [34][35]. Also, priority queueing is applied to dynamic resource scaling in [35]. Offloading strategies more broadly are reviewed in [20] and refined for latency in [18], with implementation vehicles such as [19]. In real-time scheduling theory the preemptive EDF scheduler of remains the reference for optimality [21]. Comparing priority schemes against EDF or its variants is routine practice. The gap is narrower than it first appears. The IoMT scheduling literature either studies node selection and compares against other node-selection heuristics [31], [32], or applies priority queueing without an ablation that separates placement from discipline [35].

Analytical modelling of fog latency has a long history. The non-preemptive priority result of [24] and the M/M/1 results of [36] are the tools used in this paper. While, the Linux traffic-control layer supplies the standard implementation of the three-band PfifoFast discipline [22]. In healthcare specifically, a model for minimising fog-tier latency is given in [37], and fog data center performance under load is modelled in [38], [39]. Newer efforts include an M/M/C/K priority queue model for IoT-edge-fog-cloud monitoring and a queueing-network evaluation of eHealth combining 5G, fog, and SDN slicing [40], [41]. IEEE 802.15.6 was analysed for body area networks, and a 125 ms latency limit was set for critical medical traffic [42]. Traffic adaptive priority schemes for the same setting were proposed and validated in NS-3 [43]. Packet level validation with NS-3 is widely used practice in IoT emulation [25]. Also, the performance analysis of

the IEEE 802.15.4 access technology used here goes back to [26]. Where analytical models and NS-3 are used together, however, they are usually applied to one system configuration rather than to a factorial comparison. No prior study appears to have combined an analytical priority queue model, a discrete event simulation with confidence intervals, and a packet level NS-3 run to assess the same load sweep across several disciplines and placements for fog assisted health alerting.

Table 1 places this work against five recent studies on the axes a communication reviewer would check. The five were chosen because they represent the different threads above: the fog architecture line [15], the hybrid fog-edge line [3], the IoMT task-scheduling line [17], the SDN offloading line [33], and the IoHT priority queue line [40]. The comparison is not about the metrics each of them set out to optimise. The framework in [17] is a strong multi-objective node selector, integrates security [3], and both [33] and [40] introduce priority queueing at the fog. The question here is narrower: does each study model the link, the queue, and the deadline together, does it separate placement from discipline, does it compare against a scheduling optimum, and does it report confidence intervals and a packet-level check. The last four rows of the table record axes on which some of these studies are stronger than this one, so the picture is not one-sided.

III. SYSTEM MODEL

The system has three tiers. Body worn sensors sample vital signs and send them over a short range IEEE 802.15.4 link to a local gateway. The gateway is the fog node. It runs a light decision function on each

Table 1, This work compared with five recent fog-healthcare and IoMT-scheduling studies.

Features	Elhadad et al. (2022) [15]	Islam et al. (2025) [3]	Akbar et al. (2022) [17]	Arefian et al. (2023) [33]	Farooq et al. (2025) [40]	This work
Focus	3-tier fog	Hybrid fog-edge	Task scheduling	SDN offloading	IoHT priority Q	Fog priority comms
Wireless link modelled	No	No	No	No	No	Yes
Gateway queue modelled	No	No	No	M/G/1 polling	M/M/C/K	Cobham priority
Alert deadline metric	No	No	No	Delay-sensitive %	Latency only	$P(T \leq D)$
Placement ablation	Implicit	Implicit	No	No	No	Yes
Discipline ablation	No	No	No	No	No	Yes (6 cells)
EDF benchmark	No	No	No	No	No	Yes
Load sweep	No	No	Partial	Yes	Partial	Yes
Confidence intervals	No	No	No	No	No	Yes (25 seeds)
Packet-level check	No	No	No	No	No	Yes (5 seeds)
Real vital-signs traffic	Partial	No	No	No	No	Yes
Scale tested	Concept	1500 dev.	72 patients	Yes	Multi-fog	20 nodes
Energy measured	No	Modelled	Modelled	No	No	Indirect

reading, applies a non-preemptive priority discipline, dispatches alerts for critical readings straight to the clinical unit, and forwards aggregated routine telemetry onward. The cloud handles storage, historical analytics, and the dashboard. In the baseline system the cloud also performs the classification and alerting that the gateway performs in the proposed system.

Each reading falls into one of two classes. Readings labelled High Risk form the critical class, index 1, which carries hard-deadline alerts. Readings labelled Low Risk form the routine class, index 2, which carries best effort telemetry. Let λ be the aggregate arrival rate at the gateway and let p_c and p_r be the critical and routine fractions. Both come from measurement on the real vital-signs dataset used later in the paper. That is the standard and conservative choice for the combined output of many independent low-duty-cycle wearables. The per-class rates are then presented in Eq. (1).

$$\lambda_1 = p_c \lambda, \quad \lambda_2 = p_r \times \lambda, \quad \lambda = \lambda_1 + \lambda_2 \quad (1)$$

The end-to-end latency of a reading splits into six additive terms given in Eq. (2) :

$$T_{eze} = T_{access} + T_{prop} + T_{tx} + T_{queue} + T_{proc} + T_{notify} \quad (2)$$

Here T_{access} is the medium-access and scheduling delay on the sensor hop, T_{prop} is propagation, T_{tx} is packet serialisation, T_{queue} is the queue wait at the serving node, T_{proc} is decision time, and T_{notify} is the delay in getting the alert to the clinical unit. The two systems compared here differ in which nodes contribute the queue, processing, and propagation terms. Propagation on the local hop is negligible. Over the wide area path it is not, and one crossing is written T_{wan} .

The sensor hop is described by a log-distance path loss model, which gives the achievable rate and hence the transmission time and reliability of each reading. Path loss at separation d is presented in Eq. (3)

$$PL(d) = PL(d_0) + 10n \log_{10} \left(\frac{d}{d_0} \right) + X_\sigma \quad (3)$$

Where d_0 is a reference distance, n is the path-loss exponent, and X_σ is zero-mean Gaussian shadowing. Received power follows from the link budget in Eq. (4).

$$P_r = P_t + G_t + G_r - PL(d) \quad (4)$$

With transmit power P_t and antenna gains G_t and G_r . The signal-to-noise ratio is $\gamma = P_r / (kT_0BF)$. By the Shannon bound the achievable rate is given in Eq. (5).

$$R = B \log_2(1 + \gamma) \quad (5)$$

So a packet of L bits takes $T_{tx} = L/R$. Residual bit errors are handled by an automatic repeat request (ARQ) scheme. With packet error rate PER the expected number of transmissions per delivered packet is $1/(1 - PER)$, giving an effective transmission time as indicated in Eq. (6).

$$E[T_{tx}] = \frac{L}{R(1 - PER)} \quad (6)$$

With the access link characterised, attention turns to the serving nodes. Each serving node, whether gateway or cloud server, is modelled as a single-server queue with Poisson arrivals and exponential service at rate μ , that is an M/M/1 system with offered load $\rho = \lambda/\mu$. In the cloud-only baseline both classes share one first-in first-out queue. Its mean queue wait and sojourn time are calculated using Eq. (7).

$$W_{FIFO} = \frac{\rho}{\mu(1 - \rho)}, \quad T_{FIFO} = \frac{1}{\mu - \lambda} \quad (7)$$

In the proposed system the gateway serves the critical class ahead of the routine class, but never interrupts a service already in progress. Writing $\rho_k = \lambda_k/\mu$ for the per-class load and σ_k for the cumulative load through class k , the mean residual service time for exponential service is $R_0 = \rho/\mu$, and the Cobham formula gives the mean queue wait of each class as in Eq. (8).

$$W_k = \frac{R_0}{[(1 - \sigma_{k-1})(1 - \sigma_k)]} \quad (8)$$

For the two class case the critical and routine waits become as estimated in Eq. (9).

$$W_1 = \frac{\left(\frac{\rho}{\mu}\right)}{(1 - \rho_1)}, \quad W_2 = \frac{\left(\frac{\rho}{\mu}\right)}{[(1 - \rho_1)(1 - \rho)]} \quad (9)$$

The proposed design is based on Eq. (9). The critical class wait W_1 depends only on the critical load ρ_1 , not on the total load ρ . It therefore stays bounded even as routine traffic pushes the system towards saturation. Under the shared FIFO queue of the baseline the critical class delay instead inherits the full system term $(1 - \rho)^{-1}$ of Eq. (7), which diverges as ρ approaches 1. Composing the access, queue, processing, and notification terms gives the mean end-to-end latency of a critical alert under each system. In the proposed system the alert is decided and dispatched locally, so no wide-area crossing is paid in Eq. (10).

$$T_{crit}^{fog} = T_{access} + W_1 + \frac{1}{\mu} + T_{proc}^{fog} + T_{notify} \quad (10)$$

In the cloud only baseline the reading must reach the cloud and the alert must come back, so two crossings and the shared queue wait are added (Eq. (11)).

$$T_{crit}^{cloud} = T_{access} + 2T_{wan} + W_{FIFO} + \frac{1}{\mu} + T_{proc}^{cloud} + T_{notify} \quad (11)$$

The fog decision function is a light rule that runs in microseconds, so T_{proc} is negligible beside the millisecond scale communication terms. The main reliability metric is the deadline-meet ratio, the probability that a critical alert arrives

within a clinically motivated deadline D is estimated with Eq. (12).

$$R_D = \Pr(T_{e2e} \leq D) \quad (12)$$

Network efficiency is captured by the backhaul offload ratio. The baseline forwards every reading to the cloud. The proposed system handles critical readings locally and batches routine readings in groups of A before forwarding, so the share of original traffic

reaching the backhaul is p_r/A and the offload ratio is given in Eq. (13).

$$\eta = 1 - \frac{p_r}{A} \quad (13)$$

These three elements give a complete picture of the alert path. The wireless link fixes how long each reading takes to reach the gateway. Once it arrives, a queue wait follows before service begins, and that wait is what the priority discipline is designed to shape. Deadline and backhaul then turn those times into the criteria used from here on.

IV. EXPERIMENTAL SETUP

The previous section laid out the model. In this section the dataset is described, along with how a stream of network packets is generated from it. The two systems under test are then specified as algorithms, and the simulation environment, parameters, and metrics used to evaluate them are set out.

A. Dataset and Traffic Generation

A public human vital signs dataset of 200 020 records the Kaggle data source is used for the evaluation. Each record is an independent subject. It holds six primary measurements, namely heart rate, respiratory rate, body temperature, oxygen saturation, and systolic and diastolic blood pressure, together with basic demographics, four derived indicators, and a binary risk label. There are no missing values and the sexes are almost equally represented. What matters most for the traffic model is the risk label, which splits the records into 105 115 High Risk cases at 52.55% and 94 905 Low Risk cases at 47.45%. Those two fractions define the critical and routine classes. The dataset also carries a timestamp field, but the records are cross-sectional rather than a continuous per-patient stream, which is why the arrival process is modelled separately rather than read from the data. Table 2 summarises the physiological attributes.

Preprocessing is minimal because the data are complete and consistent. A wearable transmits the six primary vitals plus a subject identifier and a timestamp. The derived indicators are recomputed at the receiver and are not sent over the air. Encoding heart rate, respiratory rate, and the two blood-pressure values as small integers, and temperature and oxygen saturation as half-precision floats, gives a 16-byte application payload. Adding the 25-byte physical and MAC overhead of an IEEE 802.15.4 frame gives an on-air packet of $L = 41$ bytes. Both classes use the same packet size, so they differ only in scheduling priority and not in transmission cost.

The triage function τ maps the vital-sign vector of an incoming reading to a class. Analysis of the dataset shows that its risk label is produced by a deterministic rule over the vitals, which τ reproduces exactly. The rule is provided by Eq. (14).

$$\tau(v) = \text{critical if } (HR > 90.5 \text{ bpm}) \vee (BMI \leq 18.5) \vee (BMI > 30.0), \text{ routine otherwise} \quad (14)$$

Table 2, Statistical summary of the physiological attributes.

Attribute	Unit	Mean $\pm$ SD	Range
Heart rate	bpm	79.53 $\pm$ 11.55	[60, 99]
Respiratory rate	breaths/min	15.49 $\pm$ 2.29	[12, 19]
Body temperature	°C	36.75 $\pm$ 0.43	[36.0, 37.5]
Oxygen saturation	%	97.50 $\pm$ 1.44	[95, 100]
Systolic blood pressure	mmHg	124.44 $\pm$ 8.66	[110, 139]
Diastolic blood pressure	mmHg	79.50 $\pm$ 5.76	[70, 89]
Pulse pressure	mmHg	44.94 $\pm$ 10.40	[21, 69]
Mean arterial pressure	mmHg	94.48 $\pm$ 4.80	[83.3, 105.7]
BMI	kg/m ²	25.00 $\pm$ 6.45	[12.5, 44.4]

A reading is treated as critical when the subject shows tachycardia or an abnormal body-mass index, and as routine otherwise. The rule is easy to read physiologically and is realised at the gateway as a depth-three decision tree of seven nodes. Because τ is an exact fixed-depth rule and not a learned statistical model, it runs in microseconds. That is what justifies treating the fog processing term as negligible. It must be said plainly that this near-perfect fidelity is a consequence of how the dataset labels were built. In this paper τ serves as a faithful and very cheap way to generate the two traffic classes for a communication study. It is not a validated clinical detector, and discussion section returns to that point.

Traffic is generated as follows. Each simulated arrival draws a reading at random from the dataset, inheriting its payload and, through τ , its class. The measured class proportions therefore set the per-class arrival rates through Eq. (1), with $p_c = 0.526$ and $p_r = 0.474$. Because the dataset carries no genuine temporal process, arrivals follow a Poisson process. The aggregate rate λ is swept so that the offered load ρ runs from 0.30 to 0.95, which covers light load through to near saturation. The resulting stream of class-tagged readings is the common input to both systems.

B. Systems Under Test

Algorithms 1 and 2 specify the two systems. Algorithm 1 is the proposed fog system. The gateway classifies each reading locally, serves the two classes under a non-preemptive priority discipline, dispatches critical alerts with no wide-area crossing, and batches routine telemetry before forwarding it. Algorithm 2 is the cloud-only baseline. The gateway forwards every reading unclassified over the wide-area network, and all decision-making, alerting, and storage happen at a single shared first-in first-out queue in the cloud.

Algorithm 1 Fog-assisted priority scheduling and alerting

Require: reading stream S from access link; triage rule τ ; deadline D ; aggregation factor A
Ensure: timely local dispatch of critical alerts; aggregated routine telemetry to cloud

```

1: initialize high-priority queue  $Q_H \leftarrow \emptyset$ ,
   low priority queue  $Q_L \leftarrow \emptyset$ , batch  $B \leftarrow \emptyset$ 
2: for each reading  $x$  arriving from
   the access link do
3:    $(v, t_{arr}) \leftarrow \text{Decode}(x)$ 
4:    $c \leftarrow \tau(v)$ 
5:   if  $c = \text{critical}$  then
6:     enqueue  $x$  into  $Q_H$ 
7:   else
8:     enqueue  $x$  into  $Q_L$ 
9:   end if
10: end for
11: while server is idle and  $(Q_H \cup Q_L) \neq \emptyset$  do
12:   if  $Q_H \neq \emptyset$  then
13:      $x \leftarrow \text{dequeue}(Q_H)$ 
14:   else
15:      $x \leftarrow \text{dequeue}(Q_L)$ 
16:   end if
17:   process  $x$ 
18:   if  $\text{class}(x) = \text{critical}$  then
19:     generate alert  $a(x)$ 
20:     dispatch  $a(x)$  to clinical unit
       locally
21:   else
22:      $B \leftarrow B \cup \{x\}$ 
23:     if  $|B| = A$  then
24:       forward  $\text{Aggregate}(B)$  to cloud;
        $B \leftarrow \emptyset$ 
25:     end if
26:   end if
27: end while

```

Algorithm 2 Cloud-only FIFO baseline

Require: reading stream S from access link; deadline D
Ensure: alerts generated and returned from the cloud

```

1: for each reading  $x$  arriving from
   the access link do
2:   transmit  $x$  to cloud over WAN
3:   enqueue  $x$  into cloud FIFO queue  $Q$ 
4: end for
5: while server is idle and  $Q \neq \emptyset$  do
6:    $x \leftarrow \text{dequeue}(Q)$ 
7:    $v \leftarrow \text{Decode}(x)$ 
8:    $c \leftarrow \text{classify}(v)$ 
9:   store  $x$ 
10:  if  $c = \text{critical}$  then
11:    generate alert  $a(x)$ 
12:    transmit  $a(x)$  to clinical unit
       over WAN
13:  end if
14: end while

```

The contrast between the two algorithms is the source of the performance gap analysed later. In Algorithm 1 a critical alert is produced and delivered entirely inside

the local domain, so its latency is governed by the bounded priority-queue wait of Eq. (9). In Algorithm 2 every reading and every resulting alert crosses the wide-area network and shares one unprioritised queue whose wait grows without bound as the load approaches saturation.

C. Simulation Environment, Parameters, and Metrics

Two complementary evaluators are employed. The primary one is an event-driven discrete-event simulation (DES) that implements the non-preemptive priority queue of the gateway and the shared FIFO queue of the cloud baseline, then composes the resulting queue wait with the access, propagation, processing, and notification terms. The secondary one is the closed-form analytical model, used as an independent cross-check. Arrivals are drawn as exponential inter-arrival times, service times are exponential, wide-area latency is drawn from a Gaussian about its mean to capture jitter, and access transmission time is inflated by the ARQ factor. The DES is written in Python 3 with NumPy and SciPy and runs on a commodity workstation. The same code was also run in a hosted browser-based Python environment for the transient experiment, which shows it is portable. Every replication uses its own deterministically seeded random stream, so results are exactly reproducible, and the source code is released with the paper.

The fog-side path is checked at packet level by a third and mechanistically independent evaluator built in NS-3. Figure 1 shows the simulated topology. Twenty wearable nodes talk to a gateway over an IEEE 802.15.4 link with unslotted CSMA-CA, IPv6 over 6LoWPAN, and a PfifoFast queue at the wearable egress into which critical readings go to band 0 and routine readings to band 2 by socket priority. Each wearable emits a Poisson stream whose class is drawn from the dataset mix, so offered load, packet size, and class distribution match the DES. The NS-3 study is confined on purpose to the fog-side portion of the path. The wide-area link is not simulated at packet level and is instead accounted for by the analytical model, as the dashed arrow in Figure 1 indicates. For the packet-level deadline check a fog-portion budget of 45 ms is used, which is 30% of the 150 ms end-to-end deadline, so the comparison stays fair.

The non-trivial parameter choices are justified as follows. The access tier runs at the 250 kb/s rate of the 2.4 GHz IEEE 802.15.4 physical layer, and the 3 ms mean access delay agrees with the contention-period medium-access delay reported for slotted CSMA/CA under light load [26]. The 41-byte packet is derived above. The service rate $\mu = 150$ readings/s is applied identically at gateway and cloud. Because the comparison between the systems is relative, its absolute value acts only as a scale factor, and it is chosen so the load sweep spans light load through to near saturation, which is the regime where the two

disciplines differ. The 40 ± 8 ms one-way wide-area latency represents a regional cloud over a multi-hop path and matches the cloud-path delays that make cloud only healthcare pipelines latency bound [3], [5]. It is modelled with Gaussian jitter, and the baseline crosses it twice. The fog processing time follows from the seven-node rule and is effectively negligible, while the cloud is given a larger 1 ms inference time. The 150 ms deadline is deliberately strict. It is tighter than the sub 300 ms requirement reported for continuous real-time vital-sign and ECG transmission [4], [5], reflecting the need to notify a clinician promptly once deterioration is detected.

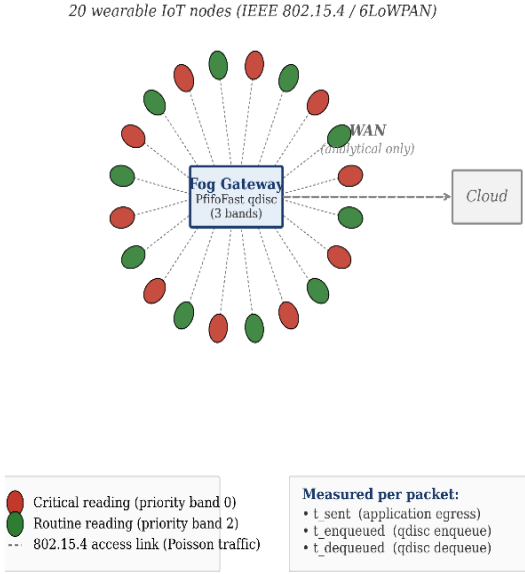


Figure 1, Simulated topology.

The evaluation protocol is as follows. Offered load is swept over the nine values in Table 3. At each point the system is simulated for 25 independent replications of 20 000 served readings. Every reported quantity is given as a sample mean with a 95% confidence interval, computed as $\bar{x} \pm 1.96s/\sqrt{R}$ over the $R = 25$ replications. To confirm that the latency advantage is real rather than a product of simulation variability, the two systems are compared at $\rho = 0.90$ with a paired Wilcoxon signed-rank test over the 25 matched replications. Simulated mean latencies are also compared against the analytical predictions of Eqs. (10) and (11).

Four metrics are reported. The mean critical-alert end-to-end latency is the average of T_{e2e} over all critical arrivals. The deadline-meet ratio R_D of Eq. (12) is estimated as the empirical share of critical alerts delivered within D . The backhaul-offload ratio η of Eq. (13) measures the cut in wide-area traffic. Finally, the latency budget is decomposed into its constituent terms so that the source of the difference between the systems can be seen directly.

Table 3: Communication and evaluation parameters.

Parameter	Symbol	Value
Node service rate	μ	150 readings/s
Critical / routine fraction	p_c / p_r	0.526 / 0.474 (from dataset)
Access technology	-	IEEE 802.15.4, 250 kb/s
On-air packet size	L	41 bytes (16 B payload + 25 B PHY/MAC)
Mean access delay	T_{access}	3 ms
Wide-area one-way latency	T_{wan}	40 ± 8 ms
Fog processing time	T_{proc}^{fog}	≈ 0.05 ms (7-node tree)
Cloud processing time	T_{proc}^{cloud}	1 ms
Local alert dispatch	T_{notify}	2 ms
Alert deadline	D	150 ms
Routine aggregation factor	A	10
DES replications / readings per run	-	25 seeds / 20 000
NS-3 version / nodes / seeds	-	ns-3.39 / 20 wearables / 5 seeds
NS-3 queue discipline	-	PfifoFast, 3 bands, 1000 pkt limit
Load sweep	ρ	0.30, 0.45, 0.60, 0.70, 0.80, 0.85, 0.90, 0.93, 0.95

V. RESULTS

The results are presented in the order in which the argument is built. The analytical model is first validated against the discrete-event simulation, and the response of both systems to rising load is examined. The two headline metrics, latency and deadline compliance, are then reported, followed by a decomposition that explains where the difference comes from and a measurement of the backhaul saving. A statistical test and effect-size analysis establish that the advantage is real. The fog-side guarantee is then checked independently at packet level in NS-3. A factorial ablation separates the contribution of placement from that of discipline and benchmarks the stock priority queue against the optimal scheduler. Finally, a transient experiment tests whether the guarantee survives a shifting traffic mix. Unless stated otherwise, all reported means carry 95% confidence intervals over the 25 replications defined above, and the analytical expressions serve as an independent cross check.

Validation comes first. In Table 4 the mean critical-alert latency from the discrete-event simulation is compared against the closed-form predictions of Eqs. (10) and (11) across the load range. The two agree to within 1.5% for loads up to 0.93 and within 6% at 0.95, where finite-buffer and finite-sample effects are strongest. That agreement, visible as the almost coincident solid and dashed curves in Figure 2, confirms that the simulation is implemented correctly and that the analytical model predicts system behaviour faithfully. The remaining results can therefore be read with confidence that model and simulation tell the same story.

Table 4, Simulation against analytical mean critical-alert latency, in ms.

ρ	Fog (sim.)	Fog (anal.)	Cloud (sim.)	Cloud (anal.)
0.30	15.4	15.4	96.9	96.8
0.60	18.9	18.9	104.0	104.0
0.80	22.3	22.2	120.1	120.6
0.90	24.2	24.4	151.7	154.0
0.95	25.5	25.7	208.4	220.6

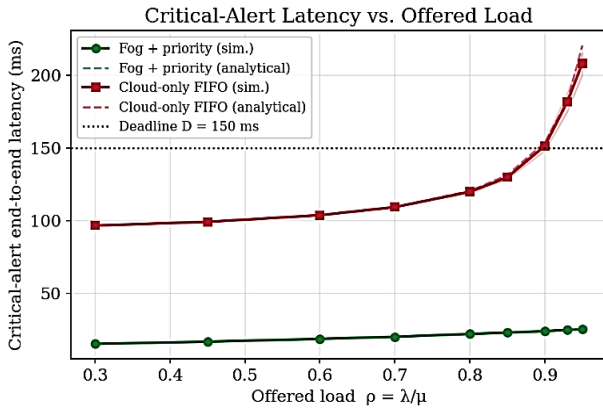


Figure 2, Mean critical-alert end-to-end latency against offered load, with 95% confidence bands and analytical overlays.

Figure 2 shows how mean critical-alert latency behaves as offered load rises from 0.30 to 0.95. Two curves separate almost immediately. The proposed system, the lower green curve, holds latency between 15.4 ms and 25.5 ms across the whole range, and its slope is almost flat. That flatness is the signature of Eq. (9): the critical-class wait depends only on the critical load, so it is shielded from the growth of routine traffic and barely responds as the system fills. The baseline, the upper red curve, tells the opposite story. It starts high, at 96.9 ms, because even at light load every alert already pays two wide-area crossings of about 40 ms each. It then climbs steeply as the shared queue approaches saturation, since under FIFO the critical class inherits the full-system wait of Eq. (7), which diverges as ρ approaches one. The baseline reaches 151.7 ms at $\rho = 0.90$ and 208.4 ms at 0.95,

crossing the 150 ms deadline near 0.90. The vertical gap between the two curves is the quantity of interest, and at the reference load of 0.90 it corresponds to an 84.0% cut in mean latency. The tight confidence bands, barely wider than the plotted markers, show that this separation is stable across the 25 replications.

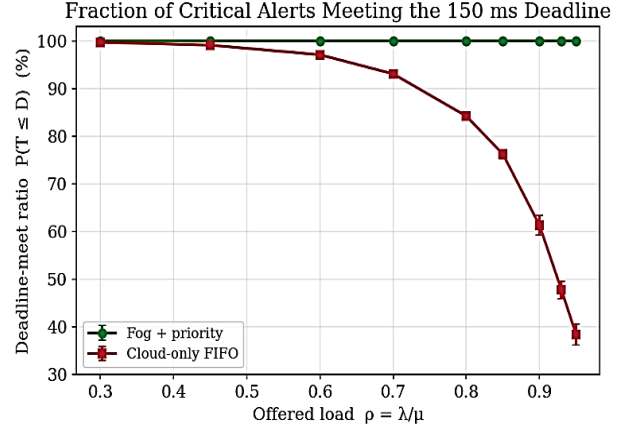


Figure 3, Share of critical alerts delivered within the 150 ms deadline against offered load.

Latency averages, however, hide what matters clinically, which is the fraction of alerts that actually arrive in time. Figure 3 reports exactly that, the deadline-meet ratio of Eq. (12), as offered load rises. The proposed system holds a flat line at essentially 100%: at every load tested, virtually every critical alert is delivered within the 150 ms deadline. The baseline begins almost as well, meeting the deadline for 99.7% of alerts at light load, but then falls away as congestion builds. It drops to 84.2% at $\rho = 0.80$, to 61.3% at 0.90, and to 38.4% at 0.95. The shape of this decline mirrors the steep latency rise of Figure 2, which is expected, since the alerts that miss the deadline are precisely those whose latency has grown past 150 ms. Put plainly, under heavy load the baseline fails to deliver almost two thirds of time-critical alerts on time, while the proposed system remains fully compliant. This curve carries the clearest clinical meaning of any result in the paper, because a late deterioration alert is in practice a missed one.

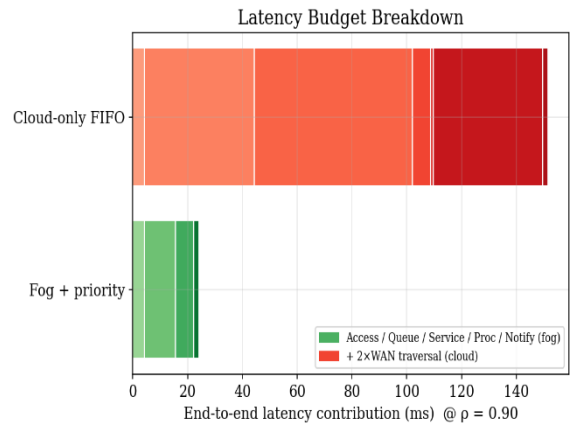


Figure 4, Latency budget at offered load 0.90.

Figure 4 decomposes the mean latency budget at the reference load of 0.90 and makes the cause of the gap explicit. Each bar is split into its component terms from Eq. (2). In the cloud bar, two blocks dominate everything else: the outbound and return wide-area crossings, at about 40 ms each, together account for the great majority of the 151.7 ms total. The access, queue, processing, and notification terms are small by comparison. In the fog bar those two large blocks are simply absent, because the alert is decided and dispatched locally and never leaves the local domain. What remains is a short stack of access and queue terms adding up to roughly 24 ms. The figure therefore shows that the advantage is not won by shaving milliseconds off many terms, but by removing two large fixed costs entirely.

Latency and reliability are not the only concern; the load placed on the wide-area link matters too, especially where bandwidth or battery is scarce. Figure 5 quantifies that benefit, following Eq. (13). The baseline forwards every reading to the cloud, so its backhaul bar is at full height. The proposed system handles critical readings locally and batches routine telemetry in groups of $A = 10$ before forwarding, so it sends only 4.7% of the original readings over the backhaul, a 95.3% cut shown as the much shorter bar. That directly reduces upstream bandwidth use and, because it lowers the number of gateway-initiated wide-area transmissions, the communication energy spent per delivered reading. The saving is a by-product of the same design choice that lowers latency: deciding locally means most traffic never needs to cross the wide-area link at all.

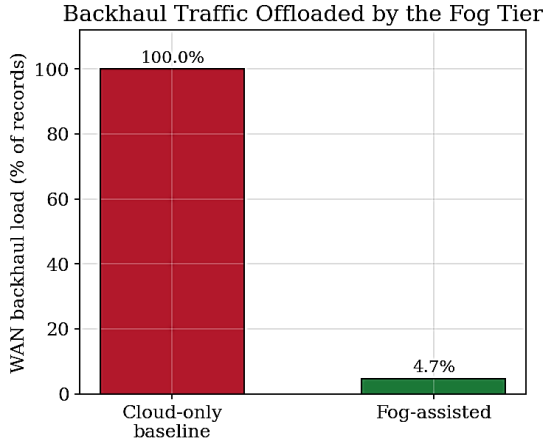


Figure 5, Wide-area backhaul load, cloud-only baseline against the fog-assisted system.

The separation in Figures 2 and 3 is large, but its statistical strength should be established rather than assumed. To confirm that the latency advantage is not a product of simulation variability, the per-replication mean critical-alert latencies of the two systems at $\rho = 0.90$ were compared with a paired Wilcoxon signed-rank test over the 25 matched replications. The difference is highly significant, with $p = 5.96 \times 10^{-8}$.

Effect size was also computed, since a p-value alone says nothing about magnitude. Paired Cohen's d is 20.1 and the independent-samples value is 27.9, both far above the conventional 0.8 threshold for a large effect, and Cliff's Δ evaluates to 1.00. That last figure means every one of the 25 baseline replications produced a higher mean latency than every one of the 25 fog replications. The two systems do not overlap at all in their per-replication behaviour, so the advantage is not merely significant but total.

All the results so far rest on the analytical model and the discrete-event simulation, which share the same M/M/1 abstraction. It is worth checking the fog-side guarantee against a third evaluator that does not. The load sweep was therefore replicated in NS-3 over five independent seeds and aggregated as a mean with a 95% confidence interval. Across the range $\rho = 0.30$ to 0.95 the mean access-plus-queue latency stayed between 1.17 ms and 1.80 ms, and the deadline-meet ratio for critical alerts stayed between 99.52% and 99.82%, with confidence half-widths below 0.14 percentage points. Figure 6 overlays these packet-level measurements, shown as diamonds, on the discrete-event fog curve of Figure 3, shown as the solid line. The diamonds sit almost exactly on the line. This is a meaningful check, because NS-3 captures realistic IEEE 802.15.4 medium-access contention, packet fragmentation, and header overhead that the M/M/1 model abstracts away, yet it still produces deadline-meet ratios indistinguishable from unity even near saturation. The agreement confirms the model's prediction that the fog-side portion of the path is not what limits reliability. Taken together with Figure 3, it isolates the two wide-area crossings paid by the baseline as the only quantitative source of deadline violations.

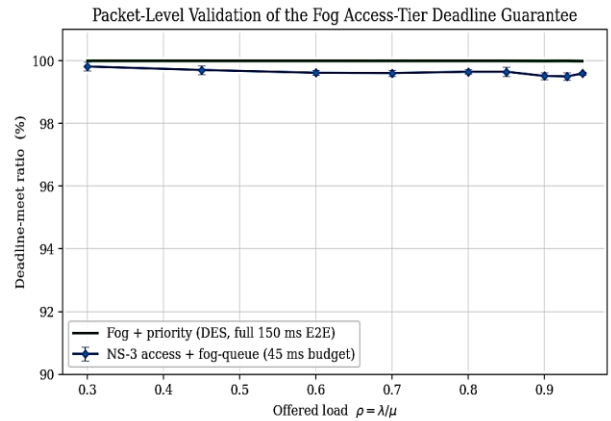


Figure 6, Packet-level validation of the fog access-tier deadline guarantee.

The results to this point compare the whole proposed system against the whole baseline, but that comparison mixes two independent design choices together: where the decision function sits, fog or cloud, and how the queue is ordered, FIFO or priority. To separate them, a factorial ablation was run over the four

combinations. FIFO is the classical shared queue [36]. Non-preemptive strict priority is the discipline analysed in [24] and standardised as the three-band PfifoFast queue of the Linux traffic-control layer [22]. All variants share the parameters of Table 3. The fog variants run the triage function locally and dispatch alerts without a wide-area crossing; the cloud variants decide at the cloud and pay two crossings per alert.

The theoretically deadline optimal EDF scheduler was added at both placements as a fifth and sixth point. EDF is preemptive and deadline-aware per packet, and it is the standard modern comparator in real time and IoMT scheduling work, where it and its variants are routinely used to benchmark priority based schemes for health-critical workloads.

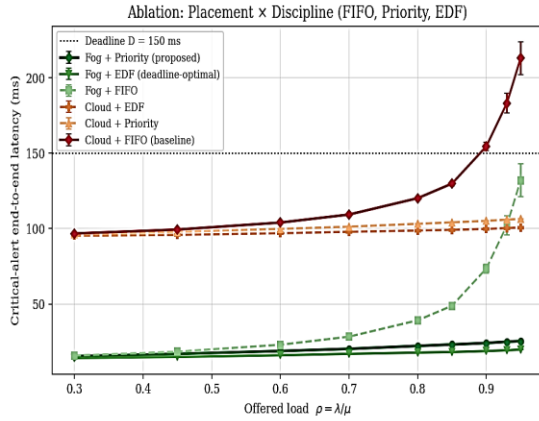


Figure 7, Mean critical-alert latency for six combinations of placement and discipline over 25 seeds.

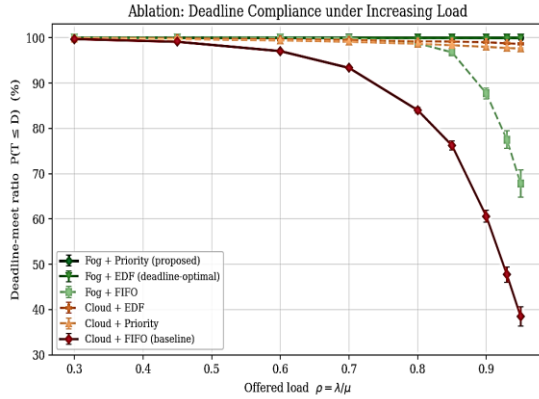


Figure 8, Deadline-meet ratio against offered load for the same six variants.

Figures 7 and 8 plot mean latency and deadline compliance for all six variants across the load sweep, and Table 5 lists the values at the reference load. Reading the two figures together, the six curves fall into a clear pattern: the two fog variants sit low and flat, the two cloud variants sit high, and within each pair the priority and EDF disciplines track each other closely while FIFO lags.

Table 5, Ablation of placement and discipline at offered load 0.90.

Variant	Mean latency (ms)	Deadline met (%)	Change vs baseline
Cloud + FIFO (baseline)	154.5	60.6	reference
Cloud + Priority	105.2	98.0	−31.9%
Cloud + EDF	99.8	99.0	−35.4%
Fog + FIFO	73.5	87.7	−52.4%
Fog + Priority (proposed)	24.2	100.0	−84.3%
Fog + EDF	18.9	100.0	−87.8%

Three points come out of this. First, moving the decision function from cloud to fog cuts mean latency by 52% at $\rho = 0.90$ even with no change to the queue discipline, because the alert avoids two wide-area crossings. Fog with FIFO still degrades under load, though. The critical class shares one queue with routine traffic, so its compliance falls from 100% at light load to 87.7% at 0.90 and 67.8% at 0.95. Second, applying priority at the cloud cuts mean latency by 32% and lifts compliance from 60.6% to 98.0%, which shows the discipline is worth having in its own right even when the wide-area penalty cannot be avoided. Third, the combination does what neither part does alone. It cuts mean latency by 84% and holds essentially perfect compliance across the whole range. Fog placement removes the wide-area penalty that caps Cloud with Priority near 105 ms, and the priority discipline removes the shared-queue congestion that limits Fog with FIFO under load. The two effects complement each other rather than duplicate each other, and both are needed for the reported guarantee.

The EDF comparison sharpens the claim. On deadline compliance, which is the criterion that matters for a hard-deadline alerting system, Fog with EDF and Fog with Priority cannot be told apart. Both meet 100% of the deadline across the range. On mean latency EDF is slightly lower, 18.9 ms against 24.2 ms at $\rho = 0.90$, a gap of about 5 ms that comes from opportunistically preempting routine service. That gap does not translate into a reliability difference here, because both schedulers already sit well inside the deadline. The proposed design therefore adopts non-preemptive strict priority not because it beats EDF on latency. It does not, and no scheduler can beat EDF on that criterion under a compatible workload. It adopts the mechanism because it matches EDF's reliability using only the three-band PfifoFast primitive that ships with Linux and is available in most SDN dataplanes, with no per-packet deadline tags, no preemption support, and no deadline-sorted data structures. That is a deliberate trade of about 5 ms of average headroom for immediate deployability on ordinary hardware. Meanwhile Cloud with EDF, at

99.8 ms and 99.0%, is still capped from below by the two mandatory crossings and improves on Cloud with Priority by only about 5 ms. No queue discipline, however sophisticated, can close a gap created by placement.

One question remains. In all the results so far the class ratio is fixed at the dataset value of 52.55% critical, but a real deployment will not be so steady. The mix can shift over a ward round or a change of shift, as a population grows more acute or as more subjects enter a high-risk state. To establish how each system responds, a transient experiment was run in which the total offered load is held at 0.90 while the critical fraction is ramped linearly from 10% to 60% across ten equal observation bins. Each bin at each seed simulates 4 000 arrivals, results are averaged over 25 seeds with 95% confidence intervals, and the analytical predictions are overlaid as a consistency check.

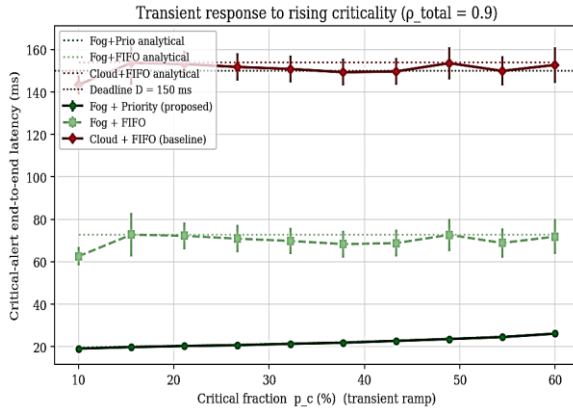


Figure 9, Transient response of mean critical-alert latency as the critical fraction rises from 10% to 60% at fixed total load 0.90. Dotted lines are analytical predictions; solid and dashed lines are the discrete-event simulation with 95% confidence bands.

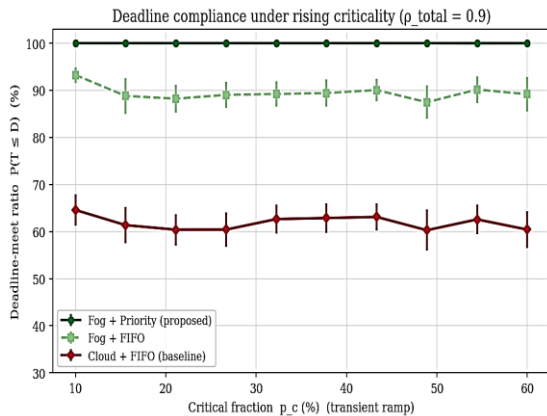


Figure 10, Deadline-meet ratio for the same transient sweep.

Figures 9 and 10 show the outcome, and three findings follow. First, the proposed system delivers 100% of critical alerts within the deadline across the entire ramp, from 10% critical to 60%. Its mean

latency in Figure 9, the lower green curve, rises only modestly, from 19.1 ms to 26.1 ms. That gentle rise is predicted quantitatively by Eq. (9), whose dotted analytical overlay sits essentially on the simulated markers for the whole ramp. The rise happens because the critical-class wait depends on the critical load, which grows from 0.09 to 0.54 as the mix shifts. Even at the far end that load stays well below one, so the system remains on the flat part of the reliability curve, and Figure 10 shows its deadline line holding at 100% throughout. The safeguard is bounded by the critical load where $\rho_1 = p_c \times \rho$, not by the share alone. At the fixed load of 0.90, ρ_1 reaches only 0.54 at 60% critical and at most 0.90 even if every reading were critical, so the deadline is never threatened here. The break-even lies in the total load, not the mix: only as ρ_1 approaches one does the critical wait grow without bound.

Second, the two FIFO variants do not adapt. Fog with FIFO holds between 87.5% and 93.3% compliance and 63 to 73 ms mean latency across the whole ramp. Cloud with FIFO holds between 60.3% and 64.6% compliance and 143 to 154 ms. In both cases the numbers barely move with the critical fraction. This is not a defect in the measurement. It follows directly from the discipline. A single shared queue does not distinguish class labels, so its per-class deadline-meet ratio is set by the total offered load, which is fixed throughout the ramp. FIFO therefore cannot respond to a workload in which more of the traffic becomes urgent. A critical packet arriving during a burst of criticality is treated exactly like any other packet at the same total load. The steady-state ablation could not reveal that, by construction.

Third, the transient result strengthens the main claim rather than merely repeating it. Under a fixed mix the proposed system cuts mean latency by 84% and holds full compliance at 0.90. The transient shows that this compliance is not an artefact of the assumed 52 to 48 split. It survives a sixfold swing in the critical fraction. Among the three systems, only the proposed one has reliability that depends on the critical load rather than the total load, and it is therefore the only one that stays deadline compliant when the traffic mix drifts within the studied range.

Taken together, the results in the studied regime support a simple reading. Two mechanisms are at work here, and they work for different reasons. One moves the time-critical decision to the fog tier. The other protects alert traffic with a non-preemptive priority queue. In combination they cut critical-alert latency by 84.0% at an offered load of 0.90 under the parameters of Table 3. Full deadline compliance is held at the same time, in conditions where the cloud-only baseline fails to deliver most alerts on time. The ablation shows that neither part is sufficient by itself. Fog placement without priority degrades under load, down to 67.8% compliance at $\rho = 0.95$, and priority confined to the cloud is capped from below by the two mandatory wide-area crossings. The structural reason

is visible in the model. The baseline pays two fixed crossings plus a shared-queue wait that diverges with total load. The proposed system pays neither.

The boundary of the claim is marked by the EDF comparison. A slightly lower mean latency is obtained under EDF, by about 5 ms at the reference load, at the cost of packet-level preemption and per-packet deadline tagging. Its advantage over non-preemptive priority would be expected to widen in workloads with several tight-deadline classes, or with different deadlines per class. Neither case is covered by the two-class model used here. Within the single tight-deadline-class regime studied, the deployable discipline is enough. It matches EDF's reliability and remains implementable on standard Linux and SDN queueing primitives. The 95.3% cut in backhaul traffic in this regime is consistent with bandwidth savings reported for recent fog-edge health architectures. Non preemption has one cost, which is head-of-line blocking. A critical alert may sit behind a routine packet that is already being transmitted. But that wait can never exceed a single packet time, and in return the stock queue needs no deadline-sorted structures and barely any CPU or memory which is exactly what a constrained gateway wants.

Earlier fog-based health monitoring studies have largely reported detection accuracy while leaving the network as a black box. Here the wireless link, the queue discipline, and the alert deadline are modelled together on real traffic. The contributions of placement and discipline are then separated by an explicit ablation, tested against both a simple and a deadline-optimal reference. What emerges is the dimension along which fog placement most directly helps time-critical alert delivery under the modelled conditions.

How far these results should be read is limited by several scope conditions. The triage function is a light rule-based classifier used to generate the workload for a communication study. Its perfect agreement with the dataset labels reflects how those labels were built, and it is not a claim about clinical diagnostic quality. A production system would replace it with a validated clinical detector, whose runtime cost would not change the queueing behaviour analysed here provided it stays inside the fog-node service budget. A deeper model would raise T_{proc} , but even a heavy detector on a modern gateway runs in a few to tens of milliseconds still below a single ~ 40 ms wide-area crossing. The conclusion holds as long as inference fits the fog-node budget; only if it overran that budget would processing rival the network terms.

The traffic model is a Poisson process with two classes, which is the standard tractable assumption for aggregate wearable traffic but not the only one. Real populations show bursty, correlated, and periodic arrivals that a Markov-modulated Poisson process, an ON and OFF model, or a self-similar model would represent more faithfully. Eq. (1) applies to the pooled traffic of many independent wearables, not to one

patient. By Khinchin's superposition result, such a pool stays close to Poisson no matter how each source behaves. And when several patients deteriorate together, what rises is the critical share, not the total load. Section 5 tests this directly: as the critical fraction climbs from 10% to 60% at a fixed load of 0.90, compliance holds at 100%, because the critical class wait tracks the critical load rather than the total. The queueing analysis assumes exponential service and a single server, chosen for tractability; extending it to general service times or several cores would generalise the results.

The tested scale is twenty wearables on one gateway. That represents a single ward, or a small to medium cohort. It is not a hospital-wide deployment with hundreds of patients, several gateways, mobility, and gateway failures. The evidence is entirely simulation-based across the analytical, discrete-event, and packet-level evaluators, which agree within 1.5% at steady state and track one another under the transient workload. Operating-system effects are not exercised by simulation. Interrupt jitter and long-term wireless channel drift both fall outside it. The NS-3 study also covers only the fog access and queue portion of the path, not an end-to-end packet-level pipeline including the wide-area leg. Energy appears only indirectly, as a cut in gateway-initiated wide-area transmissions, and was not measured on hardware. Security overheads for authentication, encryption, secure routing, and intrusion detection are outside this evaluation; the extra latency they introduce would reduce the budget available to the communication pipeline. On deployability, the design is intended to run on ordinary infrastructure with no custom hardware or protocol work. The priority discipline is the default three-band queue of the Linux traffic control layer, available in-kernel on any standard distribution and configurable with a single command. Classifying critical against routine flows by socket priority, as done in the NS-3 script, uses the standard POSIX socket option and is honoured by the traffic-control layer without modification. The triage function is a handful of comparisons per packet. On a gateway-class processor around 1.5 GHz that is well under a microsecond, five orders of magnitude below the millisecond scale queue waits modelled here. The packet level results agree with the analytical and discrete-event predictions. All three assume the same stock queue semantics. That agreement is a first indication that the modelled behaviour matches the real implementation.

VI. CONCLUSION

Alatency aware, priority queued fog communication framework has been developed and evaluated for delivering time-critical health alerts in IoT healthcare systems. Two changes define it. The time-critical decision is moved from the cloud to a fog gateway. Alert traffic is then protected by a non-preemptive strict-priority discipline, built from the

standard three-band PfifoFast queue of the Linux traffic-control layer. A single analytical model underpins the design. It combines a log-distance wireless access link with a Cobham non-preemptive priority queue at the gateway, yielding closed-form expressions for critical-class end-to-end latency and deadline-meet ratio. Evaluation was carried out on three mutually independent evaluators: the analytical model, a 25-seed discrete-event simulation, and a packet-level NS-3 simulation of the IEEE 802.15.4 access tier using the same queue. Traffic came from a real vital-signs dataset of 200 020 records, whose measured risk balance set the critical and routine mix.

The three evaluators agree within 1.5% at every steady-state operating point. Four findings follow. First, at an offered load of 0.90 mean critical-alert latency is cut by 84.0%, from 151.7 ms to 24.2 ms, and deadline compliance is lifted from 60.6% to 100% against a cloud-only FIFO baseline. A paired Wilcoxon test and very large effect sizes confirm the difference. Second, the factorial ablation shows that neither mechanism suffices alone. Fog with FIFO reaches only 87.7% compliance at the same load, while priority confined to the cloud stays near 100 ms because of the two mandatory wide-area crossings. Third, the proposed discipline matches the deadline-optimal Earliest-Deadline First scheduler on deadline compliance, both holding 100% at the reference load. The cost is about 5 ms of extra mean latency. What is bought in exchange is immediate deployability on stock infrastructure, with no per-packet deadline tagging and no preemption. Fourth, a transient test ramps the critical share from 10% to 60% at fixed total load. Full compliance is held throughout by the proposed system. The FIFO variants, by contrast, deliver alerts at rates set by total load rather than by the traffic mix. Backhaul traffic falls by 95.3% at the same time.

Within the scope stated above, these findings support a practical conclusion. Fog placement combined with a stock non-preemptive priority queue is sufficient for time-critical health-alert delivery over an IoT healthcare network, and it is attractive because it deploys so easily. One caveat should be restated. The contribution is a communication and evaluation framework, not a new clinical detector. The triage function only generates the workload, and the reported gains concern latency and reliability rather than diagnosis.

Four extensions follow naturally. The first is a physical testbed. A hardware in the loop study on a Linux gateway carrying real wearable traffic would strengthen the deployability case, since the traffic-control primitives used in the packet level study transfer directly to that setting. The second is a richer traffic model. Moving beyond independent Poisson arrivals and two classes, towards correlated bursts, non-Poisson inter-arrivals, and several deadline classes, should expose regimes in which the advantage

of Earliest-Deadline First over strict priority grows. Third, the analytical framework can be extended to non-terrestrial segments. A satellite backhaul for rural or maritime monitoring is one example, and there the wide-area delay behaves very differently, so the case for deciding at the fog becomes stronger still. Fourth, validated clinical deterioration detectors could be integrated into the gateway triage function. Early warning scores and models trained on longitudinal data are the obvious candidates. That is the most direct route to a clinically validated deployment.

VII. DATA AVAILABILITY STATEMENT

The dataset used in this study is from the Kaggle data source, "Human Vital Sign Dataset" URL: <https://www.kaggle.com/datasets/nasirayub2/human-vital-sign-dataset> (April- 2026).

REFERENCES

- [1] K. S. Awaisi, S. Hussain, M. Ahmed, A. A. Khan, and G. Ahmed, "Leveraging IoT and Fog Computing in Healthcare Systems," *IEEE Internet Things Mag.*, vol. 3, no. 2, pp. 52–56, Jun. 2020, doi: 10.1109/IOTM.0001.1900096.
- [2] Z. Ashfaq *et al.*, "A review of enabling technologies for Internet of Medical Things (IoMT) Ecosystem," *Ain Shams Eng. J.*, vol. 13, no. 4, p. 101660, Jun. 2022, doi: 10.1016/j.asej.2021.101660.
- [3] U. Islam, M. N. Alatawi, A. Alqazzaz, S. Alamro, B. Shah, and F. Moreira, "A hybrid fog-edge computing architecture for real-time health monitoring in IoMT systems with optimized latency and threat resilience," *Sci. Rep.*, vol. 15, no. 1, p. 25655, Jul. 2025, doi: 10.1038/s41598-025-09696-3.
- [4] R. Mahmud, F. L. Koch, and R. Buyya, "Cloud-Fog Interoperability in IoT-enabled Healthcare Solutions," in *Proceedings of the 19th International Conference on Distributed Computing and Networking*, Varanasi India: ACM, Jan. 2018, pp. 1–10. doi: 10.1145/3154273.3154347.
- [5] Z. A. Jaaz, M. Ezanee Bin Rusli, N. A. Rahmat, and M. K. A. Al-Adilee, "Latency optimization approaches for healthcare Internet of Things and fog computing: A comprehensive review," *J. Intell. Syst.*, vol. 34, no. 1, p. 20240543, Aug. 2025, doi: 10.1515/jisys-2024-0543.
- [6] A. Kumari, S. Tanwar, S. Tyagi, and N. Kumar, "Fog computing for Healthcare 4.0 environment: Opportunities and challenges," *Comput. Electr. Eng.*, vol. 72, pp. 1–13, Nov. 2018, doi: 10.1016/j.compeleceng.2018.08.015.
- [7] F. A. Kraemer, A. E. Braten, N. Tamkittikhun, and D. Palma, "Fog Computing in Healthcare—A Review and Discussion," *IEEE Access*, vol. 5, pp. 9206–9222, 2017, doi: 10.1109/ACCESS.2017.2704100.
- [8] P. Verma and S. K. Sood, "Fog Assisted-IoT Enabled Patient Health Monitoring in Smart Homes," *IEEE Internet Things J.*, vol. 5, no. 3, pp. 1789–1796, Jun. 2018, doi: 10.1109/JIOT.2018.2803201.
- [9] B. Farahani, F. Firouzi, V. Chang, M. Badaroglu, N. Constant, and K. Mankodiya, "Towards fog-driven IoT eHealth: Promises and challenges of IoT in medicine and healthcare," *Future Gener. Comput. Syst.*, vol. 78, pp. 659–676, Jan. 2018, doi: 10.1016/j.future.2017.04.036.
- [10] H. Atlam, R. Walters, and G. Wills, "Fog Computing and the Internet of Things: A Review," *Big Data Cogn. Comput.*, vol. 2, no. 2, p. 10, Apr. 2018, doi: 10.3390/bdcc2020010.
- [11] A. H. Sodhro, S. Pirbhulal, and V. H. C. De Albuquerque, "Artificial Intelligence-Driven Mechanism for Edge Computing-Based Industrial Applications," *IEEE Trans. Ind.*

- Inform.*, vol. 15, no. 7, pp. 4235–4243, Jul. 2019, doi: 10.1109/TII.2019.2902878.
- [12] A. A. Mutlag, M. K. Abd Ghani, N. Arunkumar, M. A. Mohammed, and O. Mohd, “Enabling technologies for fog computing in healthcare IoT systems,” *Future Gener. Comput. Syst.*, vol. 90, pp. 62–78, Jan. 2019, doi: 10.1016/j.future.2018.07.049.
- [13] M. Songhorabadi, M. Rahimi, A. MoghadamFarid, and M. Haghi Kashani, “Fog computing approaches in IoT-enabled smart cities,” *J. Netw. Comput. Appl.*, vol. 211, p. 103557, Feb. 2023, doi: 10.1016/j.jnca.2022.103557.
- [14] S. F. Ahmed, Md. S. B. Alam, S. Afrin, S. J. Rafa, N. Rafa, and A. H. Gandomi, “Insights into Internet of Medical Things (IoMT): Data fusion, security issues and potential solutions,” *Inf. Fusion*, vol. 102, p. 102060, Feb. 2024, doi: 10.1016/j.inffus.2023.102060.
- [15] A. Elhadad, F. Alanazi, A. I. Taloba, and A. Abozeid, “Fog Computing Service in the Healthcare Monitoring System for Managing the Real-Time Notification,” *J. Healthc. Eng.*, vol. 2022, pp. 1–11, Mar. 2022, doi: 10.1155/2022/5337733.
- [16] S. Rani, A. Kataria, S. Kumar, and P. Tiwari, “Federated learning for secure IoMT-applications in smart healthcare systems: A comprehensive review,” *Knowl.-Based Syst.*, vol. 274, p. 110658, Aug. 2023, doi: 10.1016/j.knsys.2023.110658.
- [17] K. Alatoun, K. Matrouk, M. A. Mohammed, J. Nedoma, R. Martinek, and P. Zmij, “A Novel Low-Latency and Energy-Efficient Task Scheduling Framework for Internet of Medical Things in an Edge Fog Cloud System,” *Sensors*, vol. 22, no. 14, p. 5327, Jul. 2022, doi: 10.3390/s22145327.
- [18] R. Mahmud, K. Ramamohanarao, and R. Buyya, “Latency-Aware Application Module Management for Fog Computing Environments,” *ACM Trans. Internet Technol.*, vol. 19, no. 1, pp. 1–21, Feb. 2019, doi: 10.1145/3186592.
- [19] S. Tuli, R. Mahmud, S. Tuli, and R. Buyya, “FogBus: A Blockchain-based Lightweight Framework for Edge and Fog Computing,” *J. Syst. Softw.*, vol. 154, pp. 22–36, Aug. 2019, doi: 10.1016/j.jss.2019.04.050.
- [20] M. Aazam, S. Zeadally, and K. A. Harras, “Offloading in fog computing for IoT: Review, enabling technologies, and research opportunities,” *Future Gener. Comput. Syst.*, vol. 87, pp. 278–289, Oct. 2018, doi: 10.1016/j.future.2018.04.057.
- [21] C. L. Liu and J. W. Layland, “Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment,” *J. ACM*, vol. 20, no. 1, pp. 46–61, Jan. 1973, doi: 10.1145/321738.321743.
- [22] W. Almesberger, “Linux traffic control-implementation overview,” 1998.
- [23] M. Harchol-Balder, *Performance Modeling and Design of Computer Systems: Queueing Theory in Action*, 1st ed. Cambridge University Press, 2013. doi: 10.1017/CBO9781139226424.
- [24] A. Cobham, “Priority Assignment in Waiting Line Problems,” *J. Oper. Res. Soc. Am.*, vol. 2, no. 1, pp. 70–76, Feb. 1954, doi: 10.1287/opre.2.1.70.
- [25] G. F. Riley and T. R. Henderson, “The ns-3 Network Simulator,” in *Modeling and Tools for Network Simulation*, K. Wehrle, M. Günes, and J. Gross, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 15–34. doi: 10.1007/978-3-642-12331-3_2.
- [26] J. Misić, S. Shafi, and V. B. Misić, “Performance of a beacon enabled IEEE 802.15.4 cluster with downlink and uplink traffic,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 17, no. 4, pp. 361–376, Apr. 2006, doi: 10.1109/TPDS.2006.54.
- [27] A. M. Alwakeel, “Adaptive edge-fog healthcare networks: a novel framework for emergency response management,” *J. Cloud Comput.*, vol. 14, no. 1, p. 48, Sep. 2025, doi: 10.1186/s13677-025-00784-3.
- [28] S. Iftikhar *et al.*, “HunterPlus: AI based energy-efficient task scheduling for cloud–fog computing environments,” *Internet Things*, vol. 21, p. 100667, Apr. 2023, doi: 10.1016/j.iot.2022.100667.
- [29] A. A. Mutlag *et al.*, “MAFC: Multi-Agent Fog Computing Model for Healthcare Critical Tasks Management,” *Sensors*, vol. 20, no. 7, p. 1853, Mar. 2020, doi: 10.3390/s20071853.
- [30] H. Sahu, S. Choudhari, and S. Chakole, “The Use of Blockchain Technology in Public Health: Lessons Learned,” *Cureus*, Jun. 2024, doi: 10.7759/cureus.63198.
- [31] Z. Sharif, L. Tang Jung, M. Ayaz, M. Yahya, and S. Pitafi, “Priority-based task scheduling and resource allocation in edge computing for health monitoring system,” *J. King Saud Univ. - Comput. Inf. Sci.*, vol. 35, no. 2, pp. 544–559, Feb. 2023, doi: 10.1016/j.jksuci.2023.01.001.
- [32] S. Azizi, M. Shojafar, J. Abawajy, and R. Buyya, “Deadline-aware and energy-efficient IoT task scheduling in fog computing systems: A semi-greedy approach,” *J. Netw. Comput. Appl.*, vol. 201, p. 103333, May 2022, doi: 10.1016/j.jnca.2022.103333.
- [33] Z. Arefian, M. R. Khayyambashi, and N. Movahhedinia, “Delay reduction in MTC using SDN based offloading in Fog computing,” *PLOS ONE*, vol. 18, no. 5, p. e0286483, May 2023, doi: 10.1371/journal.pone.0286483.
- [34] A. Lakhan *et al.*, “Smart-Contract Aware Ethereum and Client-Fog-Cloud Healthcare System,” *Sensors*, vol. 21, no. 12, p. 4093, Jun. 2021, doi: 10.3390/s21124093.
- [35] S. Bhushan and M. Ma, “Priority-Queue based Dynamic Scaling for Efficient Resource Allocation in Fog Computing,” 2021, *arXiv*. doi: 10.48550/ARXIV.2107.08300.
- [36] Wushow Chou, “Queueing Systems, Volume II: Computer Applications - Leonard Kleinrock,” *IEEE Trans. Commun.*, vol. 25, no. 1, pp. 180–180, Jan. 1977, doi: 10.1109/TCOM.1977.1093723.
- [37] S. Shukla, M. F. Hassan, M. K. Khan, L. T. Jung, and A. Awang, “An analytical model to minimize the latency in healthcare internet-of-things in fog computing environment,” *PLOS ONE*, vol. 14, no. 11, p. e0224934, Nov. 2019, doi: 10.1371/journal.pone.0224934.
- [38] M. Cerchecchi, F. Luti, A. Mecocci, S. Parrino, G. Peruzzi, and A. Pozzebon, “A Low Power IoT Sensor Node Architecture for Waste Management Within Smart Cities Context,” *Sensors*, vol. 18, no. 4, p. 1282, Apr. 2018, doi: 10.3390/s18041282.
- [39] S. El Kafhali and K. Salah, “Efficient and dynamic scaling of fog nodes for IoT devices,” *J. Supercomput.*, vol. 73, no. 12, pp. 5261–5284, Dec. 2017, doi: 10.1007/s11227-017-2083-x.
- [40] D. Van Anh and V.-H. Nguyen, “Leveraging Priority Queueing in IoT-Edge-Fog-Cloud Infrastructures for Efficient Healthcare Monitoring,” *IEEE Access*, vol. 13, pp. 80461–80477, 2025, doi: 10.1109/ACCESS.2025.3565679.
- [41] S. A. AlQahtani, “An Evaluation of e-Health Service Performance through the Integration of 5G IoT, Fog, and Cloud Computing,” *Sensors*, vol. 23, no. 11, p. 5006, May 2023, doi: 10.3390/s23115006.
- [42] A. K. Jacob, G. M. Kishore, and L. Jacob, “Lifetime and latency analysis of IEEE 802.15.6 WBAN with interrupted sleep mechanism,” *Sādhanā*, vol. 42, no. 6, pp. 865–878, Jun. 2017, doi: 10.1007/s12046-017-0648-2.
- [43] M. S. Akbar, Z. Hussain, M. Sheng, and R. Shankaran, “Wireless Body Area Sensor Networks: Survey of MAC and Routing Protocols for Patient Monitoring under IEEE 802.15.4 and IEEE 802.15.6,” *Sensors*, vol. 22, no. 21, p. 8279, Oct. 2022, doi: 10.3390/s22218279.