Decentralized LLM Inference over Edge Networks with Energy Harvesting

Aria Khoshsirat, Giovanni Perin, and Michele Rossi

Department of Information Engineering (DEI)

University of Padova (Padova, Italy)

Emails: aria.khoshsirat@unipd.it, giovanni.perin.1@unipd.it, michele.rossi@unipd.it

Abstract—Large language models have significantly transformed multiple fields with their exceptional performance in natural language tasks, but their deployment in resource-constrained environments like edge networks presents an ongoing challenge. Decentralized techniques for inference have emerged, distributing the model blocks among multiple devices to improve flexibility and cost effectiveness. However, energy limitations remain a significant concern for edge devices. We propose a sustainable model for collaborative inference on interconnected, battery-powered edge devices with energy harvesting. A semi-Markov model is developed to describe the states of the devices, considering processing parameters and average green energy arrivals. This informs the design of scheduling algorithms that aim to minimize device downtimes and maximize network throughput. Through empirical evaluations and simulated runs, we validate the effectiveness of our approach, paving the way for energy-efficient decentralized inference over edge networks.

I. INTRODUCTION

The recent and rapid growth of large language models (LLMs) has transformed various fields by achieving remarkable performance in natural language understanding and generation tasks [1]. From language translation [2] to content recommendation systems [3], LLMs have become indispensable tools to tackle complex linguistic challenges. However, as the capabilities and complexities of these models keep increasing, so do the demands for computational resources, posing significant challenges for their deployment in real-world applications.

One of the key challenges in the implementation of LLMs lies in the efficient utilization of computational resources [4], particularly in resource-constrained environments such as edge networks. Edge networks, which consist of devices located close to data sources, provide benefits such as decreased latency, increased privacy, and improved reliability [5]. However, the constrained energy and memory capacity of edge devices pose significant challenges to the implementation of resource-intensive models such as LLMs. To address these challenges, decentralized techniques for LLM inference are gaining traction [6]–[8]. These techniques offer several advantages over traditional centralized approaches by distributing

the workload across multiple devices. Specifically, they provide greater flexibility and cost-effectiveness while alleviating the burden on individual devices by harnessing the computational power of multiple devices in a decentralized manner.

Despite the advantages offered by decentralized LLM inference, the energy constraints of edge devices remain a critical concern. The limited power available to edge devices requires innovative solutions to ensure sustainable operation, particularly in scenarios where reliable and continuous inference is needed. Energy harvesting techniques, which capture and convert ambient energy sources such as solar, kinetic, or thermal energy into electrical energy, show potential to supply energy for modern edge computing tasks [9], [10]. By integrating energy harvesting capabilities into edge devices, we can increase their energy reserves and extend their operational lifetime, thereby facilitating the deployment of LLMs in energy-constrained environments. In addition, the use of renewable energy sources aligns with the broader objectives of environmentally friendly computing practices, contributing to the reduction of the carbon footprint associated with computational tasks.

This study explores the decentralized LLM inference over edge computing networks. Our research aims to investigate the integration of energy harvesting devices with edge computers for sustainable and efficient LLM inference. The contributions of this paper to this emerging field of study are as follows:

- Empirical energy measurements of LLM transformer blocks are performed on a commercial edge device (Nvidia Jetson AGX Orin) while implementing a network simulation for collaborative inference.
- A semi-Markov model describing the battery state of the edge computer and future evolution based on processing parameters and green energy arrivals is developed.
- Scheduling approaches for the distributed inference task are designed using the developed model to minimize device downtimes and maximize job throughput.
- A new dynamic power mode for the operation of edge devices is introduced to optimize the balance between energy consumption and task completion time.

These efforts collectively contribute to the development of energy-efficient and scalable solutions for decentralized LLM inference over edge networks.
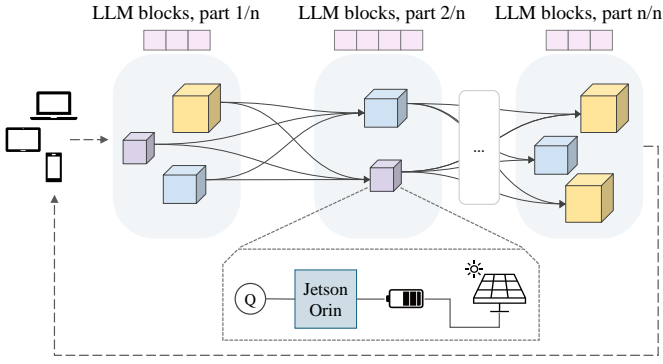
Fig. 1: **Decentralized inference network architecture.** Each cube represents a node comprising a battery-powered edge device linked to an energy source.

II. PROBLEM STATEMENT AND CHARACTERISTICS

Building upon *Petals* [7], a framework for decentralized LLM inference across interconnected devices, we introduce novel considerations to accommodate battery-powered edge devices. Like *Petals*, our framework distributes the LLM layers between groups of devices, and, within a group, identical portions of the LLM layers are replicated on each device, enabling parallel inference. Fig. 1 shows an overview of the framework. When a new request from an end user arrives (referred to as a *job* in subsequent sections), a device from each group is designated to perform an LLM inference run. The chosen device within each successive group receives the output from the preceding group as its input and transmits the output generated by its LLM blocks to a device in the subsequent group.

We consider each device within the decentralized inference framework as a battery-powered edge device. The example edge device utilized in our experiments is the Jetson AGX Orin [11], a state-of-the-art commercial edge device from Nvidia. Orin, the latest addition to the Jetson series, stands out as the most advanced platform, featuring 12 CPU cores, a powerful GPU, and ample memory capacity. It can operate in four distinct power modes, specifically at 15, 30, 50, and 60 watts. These hardware specifications make it well suited to execute resource-intensive tasks on the edge [12] with high energy efficiency. These devices can be connected to renewable energy sources, such as solar panels, to mitigate energy constraints and enable sustained operation. The energy generated by solar panels can fluctuate over time, although it tends to remain relatively stable in short intervals. Therefore, we characterize the energy arrival rates as samples from a uniform distribution bounded by two values specific to each node, for every time step. This approach acknowledges the unique energy profiles of individual edge devices within the network, which may stem from variations in solar panel models or weather conditions at their respective locations.

Each edge device can accommodate a single job in its queue Q, representing a pending task awaiting processing. This limitation to one job in the queue is particularly relevant

for inference in LLMs and caching mechanisms, which often require significant memory resources. Considering the limited memory capacity of edge devices, restricting the queue to one job helps prevent memory overflow and promotes optimal resource utilization. Devices can enter a *downtime period* where they are put in *power-saving mode*. This occurs when a low battery threshold is hit until an upper threshold of safe battery level is recovered. This ensures that the devices do not deplete their battery entirely, shutting down as a result. This is particularly relevant considering that edge devices are typically employed for various tasks and should ideally remain operational without running out of energy.

Our primary objective is to optimize resource utilization across a heterogeneous edge network setup, considering variations in device capabilities such as energy consumption, processing time, and the rate of energy replenishment. This entails intelligently and efficiently selecting devices within each group for every inference run, aiming to minimize the downtime of these devices and maximize the overall network computing throughput. Furthermore, we investigate switching between different power modes implemented in edge devices, focusing on those applicable to the Jetson AGX Orin, considering changes in energy consumption and inference time. Through this comprehensive examination, our aim is to provide insight into the optimal utilization of power modes to improve both energy efficiency and the task completion rate in edge computing environments.

III. SYSTEM MODEL FOR EDGE DEVICES

As mentioned in the previous section, we consider each edge device to be connected to a renewable energy source, e.g., a small form factor solar panel. In addition, a local battery is also included to store the collected energy. The system operates according to discrete time slots with fixed duration δ (the reference time unit). New job arrivals obey a Bernoulli model, where a new job arrives in any time slot according to a fixed probability p. For the energy model, we assume that the renewable source generates $e \geq 0$ units of energy in a time slot according to a discrete mass distribution function (MDF) $f(e)$. Energy arrivals in different time slots are assumed to be statistically independent.

The operation of any edge node j is described through consecutive *stages* defined by the variable $m = 0, 1, 2, \ldots$. In each processing stage m there are several options: *i)* if the device is in the active state and the processing queue is non-empty, a new job is processed, *ii)* if the device is active but no jobs are available, it will be idling, and *iii)* if it is in the so-called power saving mode, the computation is momentarily suspended until the energy level in the battery will increase over a certain threshold. Stages have a duration that is a multiple of δ, namely, $\kappa\delta$, where $\kappa \geq 1$ is an integer depending on the power mode, through the (binary) power saving mode variable PM. The power saving state of the node corresponds to $PM = 0$, and the active states have $PM \geq 1$. The duration of stage m corresponds to the time taken to process the job at that stage, which directly descends from the chosen power

mode. The number of slots in stage m is referred to as κ_m. Due to statistical independence, the MDF of the energy inflow in stage m is obtained by convolving k_m times the MDF $f(e)$.

At edge node j, the energy E_m^j that is available in its battery at the beginning of stage m is expressed as a multiple of a reference energy unit, with $E_m^j = 0, 1, \ldots, E_{\max}$, where $E_{\max}$ represents the maximum number of energy units that can be stored in the battery. At the beginning of stage $m+1$, the battery level E_{m+1}^j is updated as

$$E_{m+1}^j = \max\left(\min\left(E_m^j + \Delta IE_m - CE(PM), E_{\max}\right), 0\right) \quad (1)$$

where E_m^j is the energy in the battery at the beginning of stage m, ΔIE_m is the energy collected from the local energy source during stage m, and $CE(PM) \leq E_m^j$ denotes the energy consumed by node j for computation in stage m.

Each edge node j stores the jobs that are to be processed in a local queue, and this queue at stage m is denoted by Q_m^j. In what follows, we refer to the generic edge node j omitting the index for the sake of clarity. The state of the edge node in stage m is described by the tuple $S_m = (Q_m, E_m, \gamma_m)$. This state contains the queue state $Q_m \in \{0,1\}$, the discrete energy level E_m and a binary variable $\gamma_m \in \{0,1\}$ that indicates whether the node is in "power saving" or "active" mode. Together, these variables form the state space of a semi-Markov chain [13] describing the edge node evolution and play a key role in describing the system behavior and transitions at any specific time step m. This state representation enables us to analyze the statistical evolution of the edge node over time.

At any stage m, a power mode (PM) is set for each edge node, in the power saving mode, it holds $\gamma_m = PM = 0$ and in any state with $\gamma_m = 0$ new jobs are rejected. The node is put into power saving if E_m becomes smaller than a (user-defined) threshold E_{th} and it remains in power saving state until the energy E_m increases beyond a second threshold $E'_{\text{th}} > E_{\text{th}}$. When this occurs, γ_m is set back to $\gamma_m = 1$ to indicate that the node is back to "active". This enforces a hysteresis on the battery level to prevent unwanted oscillations in the node power saving behavior. In all active states ($\gamma_m = 1$) the power mode $PM \geq 1$ is deterministically set (see Section V) depending on the energy level E_m: this reflects the fact that different configurations (processing power and speed) can be used depending on the available energy. In the active states ($\gamma_m = 1$), new job arrivals are accepted as follows: we consider that a new job arrives during stage m if a new arrival occurs in at least one of the time slots within this stage. This event probability is $p_m = 1 - (1-p)^{\kappa_m}$, where the stage duration κ_m is deterministically obtained from PM.

The transition probabilities between the edge node states, $S_m = (Q_m, E_m, \gamma_m) \rightarrow S_{m+1} = (Q_{m+1}, E_{m+1}, \gamma_{m+1})$ are computed as follows:

- If $Q_m = Q_{m+1} = 0$, there are no jobs at the node, neither pending nor being computed. Therefore, all the feasible transitions must have $E_{m+1} \geq E_m$. Moreover, we discriminate between two cases:

1) If $\gamma_m = 1$, the allowed transitions are to state $S_{m+1} = (0, E_{m+1}, 1)$, with $E_{m+1} \geq E_m$ and transition probability $P(\Delta IE_m = E_{m+1} - E_m) \times (1 - p_m)$, where $P(\cdot)$ indicates a probability.
2) If $\gamma_m = 0$ the allowed transitions are to state $S_{m+1} = (0, E_{m+1}, \gamma_{m+1})$, where $E_{m+1} \geq E_m$, with probability $P(\Delta IE_m = E_{m+1} - E_m)$, where $\gamma_{m+1} = 0$ if $E_{m+1} \leq E'_{\text{th}}$ and $\gamma_{m+1} = 1$ otherwise (hysteresis). The arrival probability in this case is irrelevant as new arrivals are rejected when $\gamma_m = 0$.

- If $\gamma_m = 0$ and $Q_m = 1$ the pending job is not computed and new job arrivals are ignored. The transition to the next state $(1, E_{m+1}, \gamma_{m+1})$ has probability $P(\Delta IE_m = E_{m+1} - E_m)$. Moreover, $\gamma_{m+1} = 0$ if $E_{m+1} \leq E'_{\text{th}}$ and $\gamma_{m+1} = 1$ otherwise.
- If $\gamma_m = 1$, $Q_m = 0$ and $Q_{m+1} = 1$, the job queue is empty and a new job arrives in stage m. Again, all the feasible transitions towards stage $m+1$ have $E_{m+1} \geq E_m$, since there is no job being processed in stage m. The transition probability to state $S_{m+1} = (1, E_{m+1}, 1)$ is $P(\Delta IE_m = E_{m+1}^j - E_m^j) \times p_m$.
- If $\gamma_m = 1$, $Q_m = 1$ and $Q_{m+1} = 0$, the node in stage m will be processing the pending job in the queue and no new jobs arrive at the node during stage m. The transition probability towards state S_{m+1} in this case is $P(\Delta IE_m = E_{m+1} - E_m + CE(PM)) \times (1 - p_m)$.
  The node will transition to state $S_{m+1} = (0, E_{m+1}, \gamma_{m+1})$, where $\gamma_{m+1} = 0$ if $E_{m+1} < E_{\text{th}}$ and $\gamma_{m+1} = 1$ otherwise.
- If $\gamma_m = 1$, $Q_m = Q_{m+1} = 1$, the node will be processing an accepted job in stage m, and in the same stage a new job will also arrive. The transition probability to state $S_{m+1} = (Q_{m+1}, E_{m+1}, \gamma_{m+1})$ in this case is $P(\Delta IE_m = E_{m+1} - E_m + CE(PM)) \times p_m$, where $\gamma_{m+1} = 0$ if $E_{m+1} < E_{\text{th}}$ and $\gamma_{m+1} = 1$ otherwise.

Let $T_S = \kappa \delta$ be the dwell time of a state S. The dwell time of those states where $Q_m = 0$ corresponds to a single time slot ($\kappa = 1$). The dwell time of those states where $\gamma_m = 1$ (i.e., $PM \geq 1$) and $Q_m = 1$ corresponds to the computational time κ_m for a single job, which depends on the power mode PM of that state. The dwell time of a state for which $PM = \gamma_m = 0$ (power saving) also corresponds to a single time slot: in this case, new jobs are rejected, and the node keeps idling until the energy increases above E'_{th}.

Given this semi-Markov model, several key performance measures can be computed. For instance, the average energy level of a node in a time slot is calculated as

$$\bar{E} = \frac{\sum_S \pi_S E_S}{\sum_S \pi_S T_S}, \quad (2)$$

where the summations extend over all states, E_s represents the energy level in state s, and π_s is the stationary distribution of the embedded Markov chain. Another metric of interest is the total probability that, in any time slot, the energy of a device

will be less than a chosen threshold E_{lim}, computed as

$$\xi_{lim} = \frac{\sum_{S \text{ s.t. } E \leq E_{lim}} \pi_S T_S}{\sum_S \pi_S T_S}. \qquad (3)$$

By establishing a suitable threshold, such as the battery level that triggers the power-saving mode, this risk metric can be minimized, thereby minimizing device downtime. Note that the stationary distribution only needs to be computed once unless the network parameters change, making the algorithm cost-effective.

IV. DISTRIBUTED CONTROL OF IN-NETWORK PROCESSING

We devise three scheduling algorithms of increasing complexity, selecting the job assignment probability distribution in different ways. Specifically, the proposed policies are:

- **Uniform.** The target edge device is chosen by sampling a uniform random variable among the devices that are currently available for processing.
- **Long-term.** The scheduling probability distribution is selected to minimize the risk of the battery level falling below a given threshold, using the developed semi-Markov model. This policy yields a static optimal-on-average solution over an infinite horizon.
- **Adaptive.** The long-term-obtained processing input rates are further refined in an adaptive way considering the current (instantaneous) energy state. The idea is that a part of the input probability is moved from devices with low energy to devices with higher energy availability.

Algorithm description. To start with, we define the shorthand notation PM_i if $PM = i$ and with $PM(d)$ we mean the power mode of device d. Algorithm 1 summarizes the system operation and the above scheduling policies. At the beginning of each time slot, the energy states of the devices are updated according to Eq. (1) (line 4), and the device d chooses its power mode $PM(d)$ according to a pre-defined lookup table (line 5, see Section V). Then, the scheduling probabilities are set for each layer ℓ according to the adopted scheduling procedure (line 8), i.e., uniform, long-term or adaptive. After setting the scheduling probabilities, decentralized inference of the input can be performed. Since the stationary distribution is computed offline, scheduling subroutines can be executed fast, making the algorithm suitable for real-time applications.
Static long-term optimal scheduling. The long-term optimal job arrival probability is determined by analyzing the energy consumption of the edge devices under a known energy arrival distribution. Specifically, the maximum acceptable input rate q_{lim}^{energy} relative to the maximum tolerable risk ξ_{lim} (user-defined) can be retrieved via a root-finding algorithm, such as Brent's method [14]. The method is fed with the function given by Eq. (3), where the stationary distribution π_S depends on the variable q (job input rate). The obtained input rate q_{lim}^{energy} only considers the energy risk minimization, lacking

---

Algorithm 1 Split inference of LLMs

1: **input:** device limit input rates vector q_{lim}^{ℓ} for every layer ℓ.
2: **repeat**
3:     **for each** device d **do**
4:         Energy update (Eq. (1))   ▷ Update the energy state
5:         $PM(d) \leftarrow \texttt{choose}(PM_0, \ldots, PM_M)$ ▷ Select PM
6:     **end for**
7:     **for each** layer ℓ **do**
8:         $q^{\ell} \leftarrow \texttt{scheduling}(q_{lim}^{\ell})$ ▷ Set input probs.
9:     **end for**
10:     $\texttt{inference}(\text{input})$ ▷ Run the system
11: **until** The system is working.
12: **procedure** UNIFORM(x)
13:     $x \leftarrow 1/\texttt{length}(x)$
14:     **return** x
15: **end procedure**
16: **procedure** LONG_TERM(x)
17:     $x \leftarrow x/\sum_i x_i$
18:     **return** x
19: **end procedure**
20: **procedure** ADAPTIVE(x)
21:     $x \leftarrow \texttt{long\_term}(x)$
22:     **for each** device i^{ℓ} **do**
23:         **if** device i^{ℓ} having power mode PM_1 **then**
24:             $z \leftarrow \alpha/N_{\ell}$
25:             $x_i \leftarrow x_i - (1-z)x_i$ ▷ decrease input rate
26:         **end if**
27:         $x \leftarrow x/\sum_i x_i$ ▷ normalize to get a valid MDF
28:     **end for**
29: **end procedure**

---

the inclusion of time constraints due to processing. Let $\bar{\kappa}$ be the expected number of slots for processing a task, it holds

$$\bar{\kappa} = \frac{\sum_{S \text{ s.t. } (Q=1, \gamma=1)} \pi_S \kappa_S}{\sum_{S \text{ s.t. } (Q=1, \gamma=1)} \pi_S}. \qquad (4)$$

When the power mode is fixed, the processing slots are also fixed, i.e., $\bar{\kappa} = \kappa_S = \kappa^{PM}$. However, it is also possible to dynamically tune the power mode depending on the state S, and this will yield diversified values of $\bar{\kappa}$.

Since the input rate is the inverse of the delay, we take

$$q_{lim} = \min\{q_{lim}^{\text{energy}}, 1/\bar{\kappa}\} \qquad (5)$$

to also consider the processing delay. The value q_{lim} represents the maximum job arrival rate a single edge device can tolerate with the given energy harvesting distribution. For any layer ℓ, the scheduling probability of device $i \in \ell$ is (line 17)

$$r_i = \frac{q_{lim,i}}{\sum_{j \in \ell} q_{lim,j}}. \qquad (6)$$

This scheduling probability is used to select a device from the available ones within each group to execute an inference run.
Adaptive scheduling. For the adaptive policy, we propose a heuristic that adaptively tunes the input rate MDF. At each time step, the procedure starts by setting the input MDF to the static long-term solution (line 21). Then, the input rate of those devices in the critical power mode PM_1 is decreased by a quantity proportional to α/N_{ℓ} (line 25), where N_{ℓ} is the number of devices in layer ℓ and $\alpha \in [0, N_{\ell}]$ is a tunable parameter. Choosing the value $\alpha = |\{u \mid PM(u) = 1\}|$, i.e., the number of devices in the power mode PM_1, allows

assigning to devices $v \mid PM(v) > 1$ a rate proportional to their cardinality in the considered layer. The obtained vector is finally re-normalized to obtain a valid MDF (line 27).

V. EXPERIMENTAL RESULTS AND ANALYSIS

As a case study, we implement an LLM block with 100 encoder layers and 100 decoder layers, each employing 100 attention heads. We conduct inference on the AGX Orin device for an input with size (64 x 16 x 512), measuring both energy consumption and processing time across various power modes. Accurate power readings are obtained using the method described in [15]. We observe that for power modes of 15 W, 30 W, 50 W, and 60 W, the average processing times and energy consumption are approximately (300 s, 26 kJ), (200 s, 22 kJ), (205 s, 23.5 kJ), and (100 s, 23 kJ), respectively. The 50 W power mode exhibits about the same processing time as the 30 W mode, yet it surpasses the 30 W mode in terms of energy consumption. Therefore, due to its inefficiency in this task, we opt not to utilize it. Based on these empirical findings, in the collaborative inference simulation, a battery with a capacity of 100 kJ is chosen for each node and the time step is $\delta = 100\ s$. Every node is associated with a uniform distribution representing energy arrivals, with a *distinct mean*. The network topology consists of three groups of devices with three nodes each. For simplicity, each group hosts the same LLM block mentioned above. Communication times between devices are disregarded, and devices within each group are assumed to be fully connected to those in the next one.

We evaluate the effectiveness of different power modes in facilitating decentralized LLM inference. Four power modes are compared: conventional 15 W, 30 W, and 60 W modes, along with a "dynamic" power mode adapting to the device's current energy level. For the dynamic power mode, through manual exploration, we identified battery thresholds of 40% and 60% as optimal for transitioning from power mode 15 W to 30 W and from 30 W to 60 W, respectively. Fig. 2a compares the number of completed jobs and the average battery level in a simulation of length 100δ for a single Orin device. The 15 W power mode, corresponding to $PM = 1$, allows the device to save energy (89% of battery) while having the lowest throughput, with only 31 completed jobs, as the mode is characterized by $\kappa = 3$. Fixing the power mode to the highest possible, i.e., $PM = 3, \kappa = 1$, with 60 W, gives the dual result: we have the highest job throughput, with 58 completed jobs, but the battery is low (16%) and the risk of entering the power saving mode is high. Using $PM = 2, \kappa = 2$ at 30 W is the best tradeoff considering fixed strategies, with 42% of battery and 45 completed jobs. However, using the proposed dynamic power mode, with $PM \in \{1, 2, 3\}$ depending on battery level, job throughput increases (47 completed jobs) while having an 18% average battery improvement concerning the 30 W fixed power mode.

The maximum input rate q_{lim} is computed by applying Brent's method (see Section IV). Specifically, we calculate the risk ξ_{lim} that the battery drops below the threshold $E_{th} = 10\%$, triggering the activation of the power saving mode. Our
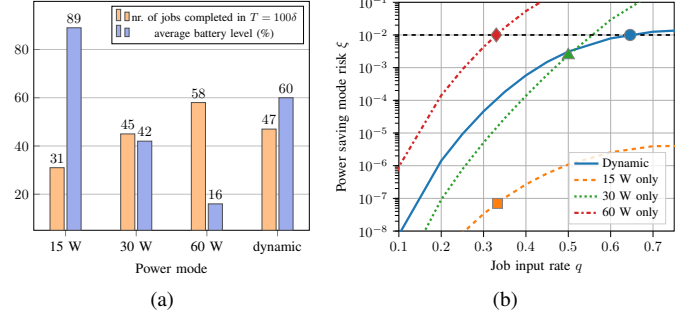

Fig. 2: **Power modes study.** In (a), we observe the effects on job throughput and energy savings for both fixed and dynamic power modes. (b) illustrates the maximum job arrival rate while keeping the risk of entering the power saving mode under a user-defined threshold $\xi_{lim} = 0.01$.

objective is to find the highest input rate q_{lim} to keep the risk less than the probability threshold $\xi_{lim} = 0.01$. The results of this experiment are shown in Fig. 2b. The lowest risk is attained by using 15 W, while the highest risk is by using the maximum power of 60 W. For the 15 W and 30 W strategies, the threshold ξ_{lim} (denoted by the horizontal dashed line) cannot be reached, as processing time constraints dominate in this case. In fact, the q_{lim}s (from Eq. (5)) are $1/3$ (the orange square) and $1/2$ (the green triangle), respectively. For the 60 W power mode, there are no time constraints, as it entirely processes a job in a single slot of duration δ. The energy risk threshold is reached at $q_{lim} \approx 0.33$ (red diamond). For the dynamic power mode, the energy threshold is hit at $q_{lim} \approx 0.64$ (blue circle). This is also well approximated by $1/\bar{\kappa}$, i.e., the average maximum rate that can be tolerated for a stable input queue. According to these results, using a dynamic power mode allows enduring a higher input rate, while controlling the downtime risk below ξ_{lim}. In other words, the dynamic power mode effectively manages energy resources while optimizing the throughput for decentralized inference over edge networks.

For subsequent experiments, we use the dynamic mode as the power strategy and investigate the scheduling policies outlined in Section IV through simulated inference runs of the system. We repeat system runs for 1,000 iterations per experiment and depict the mean and standard deviation in the forthcoming plots. We explore how the fraction of devices in power saving mode changes concerning the average energy arrival (Fig. 3a) and the job arrival probability (Fig. 3b). As can be seen, blindly applying a uniform distribution among available devices for the job assignment ends up in having a sub-optimal solution in terms of downtime probability. This is because the uniform distribution does not consider the differences in energy arrival for the scheduled devices.

In contrast, long-term selects the scheduling probabilities set by considering the average optimal solution according to the semi-Markov model developed, so that the scheduling distribution is unbalanced towards the richest devices for energy income. This allows long-term to outperform the naif
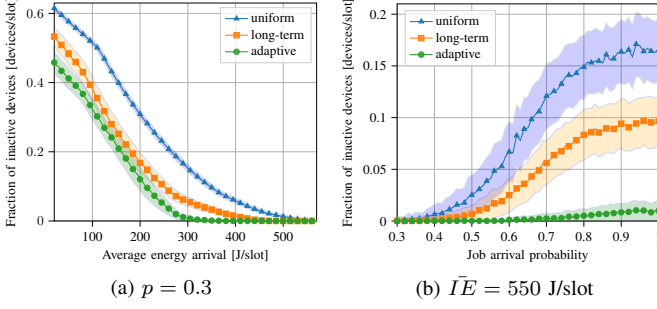
(a) $p = 0.3$

(b) $\bar{IE} = 550$ J/slot

Fig. 3: **Fraction of inactive devices.** Share of devices in power saving mode due to a lack of harvested energy *vs.* average energy arrivals (a) and job arrival probability (b).

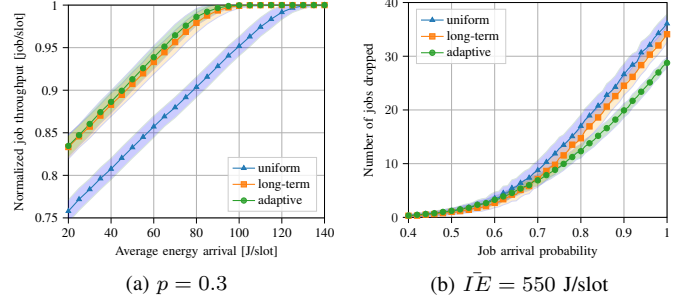(a) $p = 0.3$

(b) $\bar{IE} = 550$ J/slot

Fig. 4: **Edge network processing capacity.** The processing capacity of the proposed decentralized inference system is evaluated as normalized job throughput or fraction of dropped jobs *vs.* energy arrivals (a) and job arrivals (b).

uniform distribution: the downtime risk is reduced to 15% when varying the energy arrivals and is consistently halved when varying the job arrival rate. Using the adaptive method, a reduction of up to 10% in the downtime probability is observed compared to long-term when varying both energy and job arrival rates. Notably, with the chosen energy settings, even when one job arrives at each time slot (i.e., the processing is always active), adaptive can keep the downtime probability as low as about 1%.

In Fig. 4a we show the normalized job throughput (i.e., the number of jobs processed by the system concerning the total input jobs) with respect to the average energy arrivals. The scheduling methods based on our semi-Markov model outperform the uniform scheduling: They increase the job throughput by about 10% when the energy arrival is relatively low. Compared to long-term, using the adaptive scheduling policy allows the system to further gain some processing capacity on average (about 2%). Fig. 4b, instead, shows the dual metric, i.e., the number of jobs dropped by the system, when varying the job arrival probability. The three strategies provide similar results up to the input rate $p \approx 0.65$, where they enter a linear behavior with a similar slope. However, using uniform scheduling, we enter this region faster than the proposed methods, which have slower elbows (transition regions). On average, long-term and adaptive drop about 3 and 7 jobs less than uniform, respectively.

VI. CONCLUDING REMARKS

This work presented a semi-Markov model for edge computers equipped with energy harvesting. The model was tested with a decentralized LLM inference task, considering time and energy constraints, and using simple scheduling algorithms. The experiments were performed using parameters from real energy and time measurements of a Jetson Orin. Using the proposed semi-Markov model and the derived scheduling policies, the processing performance has been improved while effectively reducing the shutdown risk. The method presented in this paper can be adapted to similar scenarios with different parameter settings, allowing its application across various contexts and conditions.

REFERENCES

[1] W. X. Zhao, K. Zhou, J. Li, T. Tang, X. Wang, Y. Hou, Y. Min, B. Zhang, J. Zhang, Z. Dong *et al.*, "A survey of large language models," *arXiv preprint arXiv:2303.18223*, 2023.

[2] Y. Moslem, R. Haque, J. D. Kelleher, and A. Way, "Adaptive machine translation with large language models," *arXiv preprint arXiv:2301.13294*, 2023.

[3] Z. Zhao, W. Fan, J. Li, Y. Liu, X. Mei, Y. Wang, Z. Wen, F. Wang, X. Zhao, J. Tang, and Q. Li, "Recommender systems in the era of large language models (LLMs)," *IEEE Transactions on Knowledge and Data Engineering*, pp. 1–20, 2024.

[4] M. Xu, W. Yin, D. Cai, R. Yi, D. Xu, Q. Wang, B. Wu, Y. Zhao, C. Yang, S. Wang *et al.*, "A survey of resource-efficient LLM and multimodal foundation models," *arXiv preprint arXiv:2401.08092*, 2024.

[5] M. S. Murshed, C. Murphy, D. Hou, N. Khan, G. Ananthanarayanan, and F. Hussain, "Machine learning at the network edge: A survey," *ACM Computing Surveys (CSUR)*, vol. 54, no. 8, pp. 1–37, 2021.

[6] B. Yuan, Y. He, J. Davis, T. Zhang, T. Dao, B. Chen, P. S. Liang, C. Re, and C. Zhang, "Decentralized training of foundation models in heterogeneous environments," *Advances in Neural Information Processing Systems*, vol. 35, pp. 25 464–25 477, 2022.

[7] A. Borzunov, M. Ryabinin, A. Chumachenko, D. Baranchuk, T. Dettmers, Y. Belkada, P. Samygin, and C. A. Raffel, "Distributed inference and fine-tuning of large language models over the Internet," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[8] Z. Tang, Y. Wang, X. He, L. Zhang, X. Pan, Q. Wang, R. Zeng, K. Zhao, S. Shi, B. He *et al.*, "FusionAI: Decentralized training and deploying LLMs with massive consumer-level GPUs," *arXiv preprint arXiv:2309.01172*, 2023.

[9] W. Li, T. Yang, F. C. Delicato, P. F. Pires, Z. Tari, S. U. Khan, and A. Y. Zomaya, "On enabling sustainable edge computing with renewable energy resources," *IEEE Communications Magazine*, vol. 56, no. 5, pp. 94–101, 2018.

[10] G. Perin, M. Berno, T. Erseghe, and M. Rossi, "Towards sustainable edge computing through renewable energy resources and online, distributed and predictive scheduling," *IEEE Transactions on Network and Service Management*, vol. 19, no. 1, pp. 306–321, 2022.

[11] "Nvidia Jetson Orin for next-gen robotics." [Online]. Available: https://www.nvidia.com/en-gb/autonomous-machines/embedded-systems/jetson-orin/

[12] A. Archet, N. Gac, F. Orieux, and N. Ventroux, "Embedded AI performances of Nvidia's Jetson Orin SoC series," in *17ème Colloque National du GDR SOC2*, 2023.

[13] F. Grabski, *Semi-Markov Processes: Applications in System Reliability and Maintenance*. Elsevier Science, 2014. [Online]. Available: https://books.google.co.uk/books?id=Y2hzAwAAQBAJ

[14] R. P. Brent, *Algorithms for Minimization without Derivatives*, 1st ed. Englewood Cliffs, New Jersey: Prentice-Hall, 1973.

[15] N. Shalavi, A. Khoshsirat, M. Stellini, A. Zanella, and M. Rossi, "Accurate calibration of power measurements from internal power sensors on NVIDIA Jetson devices," in *2023 IEEE International Conference on Edge Computing and Communications (EDGE)*, 2023, pp. 166–170.