A Comprehensive MDP-Based Approach to Model and Optimize Discontinuous Reception (DRX) in Cellular IoT Networks

Nicholas Accurso, Nicholas Mastronarde, *Senior Member, IEEE*, and Filippo Malandra, *Member, IEEE*

***Abstract*—Due to the exponential growth of endpoints in the Internet of Things (IoT), new protocols have been proposed to utilize cellular infrastructures, allowing a large amount of IoT devices to communicate through them. These novel protocols make up the Cellular IoT (C-IoT). In C-IoT, the energy efficiency of endpoints is essential in order to reduce both operational cost and required maintenance. One method of energy reduction is discontinuous reception (DRX). DRX allows a device's radio frequency (RF) circuitry to turn off for brief periods of time. While off, the device experiences a tradeoff between saving energy and an increase in expected latency, which can be tuned by how long the device spends asleep. In this article, we model DRX as a Markov decision process (MDP). This MDP is solved using a low-complexity "DRX-aware" value iteration algorithm, then verified through simulation and analytical analysis. Further, the energy-latency tradeoff is explored by varying the device's priority on either energy or latency in addition to varying the traffic intensity. Finally, a method of traffic estimation is applied, and the model's performance in an environment with time-varying traffic intensity is explored. This approach is compared with a reinforcement learning approach, showing that the traffic estimation approach is better suited to the problem of DRX optimization.**

***Index Terms*—Cellular Internet of Things (C-IoT), constrained devices, device management, discontinuous reception (DRX), efficient communications and networking, energy efficient devices, Markov decision processes (MDPs).**

I. Introduction

TO HELP account for the massive growth of the Internet of Things (IoT), Cellular IoT (C-IoT) networking protocols have been proposed. These C-IoT protocols allow IoT traffic to communicate using existing cellular infrastructures. Two popular novel C-IoT protocols are narrowband IoT (NB-IoT), introduced by 3GPP in 2016 [1], and long term evolution (LTE) Cat-M. These protocols allow user equipments (UEs) to communicate using a more narrow bandwidth when compared with legacy cellular protocols, such as LTE. This, in turn, allows more users to coexist in the same cell. While the bandwidth limitation imposed by NB-IoT and LTE Cat-M deter many applications, it is ideal for typical IoT applications, such as smart city scenarios, in which devices communicate infrequently and increased delays are tolerable to some extent.

Compared to its competitors, such as long range wide area network (LoRaWAN) and Sigfox, C-IoT protocols offer better network performance in many areas, such as throughput and latency. Also, C-IoT offers greater reliability. For instance, NB-IoT employs a coverage enhancement feature, which allows for reliable transmissions at a greater range thanks to repeated packet transmissions. In addition to improved network performance, the deployment of C-IoT is much simpler, since the cellular infrastructures that are used by C-IoT are already in place around the world. Thus, a C-IoT network can be deployed through a software upgrade rather than replacing the hardware itself [2].

Another important aspect of C-IoT protocols is reducing the energy consumption of UEs [3]. This can be done primarily in three ways: 1) improving the scheduling and routing of information through the network; 2) processing data using more energy efficient methods (e.g., the cloud computing); and 3) introducing sleep modes for nodes in the network. There are three direct consequences of improving energy efficiency in such networks: the amount of waste generated as a byproduct of the device's operation is reduced, maintenance of devices is decreased, and the cost of operation is reduced.

However, it is rarely the case that a reduction in energy consumption does not come at a cost. Two prime examples of this are discontinuous reception (DRX) and power save mode (PSM) which are two techniques introduced in LTE to extend the battery life of end devices. DRX and PSM allow devices to briefly turn off their radio frequency (RF) circuitry, which would otherwise consume considerable energy while on. At the same time, however, the device is not reachable by the network. If a packet is sent to the device while it is in this *off* state, significant delays can be incurred since any downlink (DL) traffic will need to be buffered at the base station (BS). Thus, DRX and PSM have an inherent energy-latency tradeoff. In essence, by tuning the various timers that facilitate DRX and PSM operation, we also tune this tradeoff. In addition to changing timers, this tradeoff is also affected by the traffic conditions in the network. In this article, we formulate the problem of DRX *off* duration optimization considering traffic conditions as a Markov decision process (MDP). An MDP was selected to model this problem because it allows the modeling of a time varying environment in which an agent

Manuscript received 15 April 2024; revised 3 June 2024 and 12 August 2024; accepted 20 August 2024. Date of publication 5 September 2024; date of current version 9 October 2024. This work was supported in part by the National Science Foundation under Grant 2105230. *(Corresponding author: Filippo Malandra.)*

The authors are with the Department of Electrical Engineering, University at Buffalo, Buffalo, NY 14260 USA (e-mail: naccurso@buffalo.edu; nmastron@buffalo.edu; filippom@buffalo.edu).

Digital Object Identifier 10.1109/JIOT.2024.3452487

makes decisions that will impact both immediate and future network performance.

The main contributions of our work are as follows.

1) The problem of DRX *off* duration selection is formulated as an MDP, under the assumption that the traffic arrival distribution is known. In this model, the DRX and PSM mechanisms are jointly modeled as a controlled discrete-time Markov chain.
2) A single parameter is introduced that can tune the energy-delay tradeoff of the system. The introduction of such a parameter allows us to closely examine the various possible operating points resulting in changes to the overall system. In our survey on energy efficiency in C-IoT networks [3], we found that this tradeoff has not yet been thoroughly studied.
3) The MDP is solved using value iteration for varying traffic intensities and varying energy/delay priorities. The value iteration algorithm was modified to exploit the limited number of possible next states given the current state, resulting in drastically reduced computational complexity.
4) An analytical model is formulated to compute the expected energy consumption and delay in each of the high-level modes of operation for DRX. These models are compared with our model.
5) These results are validated via simulation, where the cost of operation is calculated for varying DRX *off* durations. Further, the delay and energy costs are separated to allow us to observe the delay versus energy tradeoff curves.
6) A method is proposed to allow the optimal DRX *off* timer selection policy to be deployed in a network where the traffic intensity is either unknown or time varying. This method estimates the traffic load and employs the appropriate DRX *off* timer according to the MDP. This method is compared with an reinforcement learning (RL) approach, showing that a traffic estimation approach is more effective in the context of DRX optimization.

This work is an extension of our previous work [4]. One of the main novelties of this work is the formulation of an analytical model to compute the expected energy and delay in each high-level DRX state. Additionally, in this work we offer a much more thorough problem formulation with slight modifications to our model to improve computational complexity. Namely, we model the transition to PSM from DRX stochastically, eliminating the need to count the number of DRX cycles elapsed which decreases the size of the state space and thus decreases the computational complexity required to find the optimal solution. Further, the value iteration algorithm is modified in this work to significantly improve computational efficiency. Finally, several new results are presented, including a method in which the optimal DRX *off* timer can be efficiently deployed in an online environment.

The remainder of this article is organized as follows. In Section II, we discuss the related work. In Section III, we formulate the problem as an MDP and solve it using the value iteration. In Section IV, we present an analytical model for the expected delay and energy in each of the macro states. In Section V, we present and analyze our results. Finally, in Section VI, we conclude the work.

II. Related Work

There exist a number of diverse ways that a device can save energy in C-IoT. A comprehensive survey of such techniques is provided in [3]. One method is through the modification of scheduling, which can be done in the uplink direction [5], the DL direction [6], or in both the uplink and DL directions [7]. In this method, the ways in which time-frequency resources are utilized and how information is routed through the network is altered to minimize energy consumption. Another approach is to modify how, where, and when a computational task is executed in order to more efficiently use energy. There are three primary ways this can be achieved: by minimizing the transmitted data [8], through the use of approximate circuitry [9], or by employing task offloading methods [10]. A third approach to energy saving is to allow the activity of the RF circuitry in a node to be reduced or stopped entirely, i.e., enter a sleep mode. This can be done in the BS [11], [12], but this technique is more commonly applied to the UE.

One common approach that has been studied extensively in this regard is the use of duty cycling, especially in the context of wireless sensor networks (WSNs). Specifically, duty cycling is used to minimize the amount of idle time a device is monitoring the network. Since this idle time often wastes energy, employing duty cycling reduces the overall energy consumption of the device. A survey of duty cycling techniques employed in WSNs can be found in [13], where they highlight a critical need for accurate models to estimate the effect of duty cycling on data transmission latency. In this article, we provide accurate analytical models for latency and energy consumption in DRX, and provide approaches to characterize and optimize the achievable energy-latency tradeoff.

A common approach in studying and modeling duty cycling problems is through the use of MDPs. Trinh et al. [14] modeled duty cycling in a wireless multimedia sensor network as an MDP and solve it using RL. The goal of the model is to find the best times to be awake or asleep given energy consumption, latency, and throughput Quality of Service (QoS) constraints. Chan et al. [15] proposed an adaptive duty cycling scheme that takes into account a device's harvesting of ambient energy. This work models the problem as an MDP, modeling the probabilistic dynamics of the problem as a continuous time Markov chain. Overall, the objective is to minimize the energy consumption of the device subject to latency and packet loss constraints. In [16], the duty cycling of a solar energy harvesting WSN node is modeled. The action taken in the model is to increment/decrement the duty cycle (i.e., the ratio of awake to asleep time), and the objective is to maximize the amount of data transmitted from the node, which requires a level energy to be maintained through harvesting. Sandoval et al. [17] seek to find the optimal transmission policies, maximizing the number of reported events for LoRa and Sigfox technologies. In this problem, events that need to be transmitted over the network are generated stochastically, and the packet reception probability of events of varying priority is to be maximized. In [18], a duty-cycling optimization problem is formulated as an MDP and solved using RL. In this problem, the considered device has an active state during which data transmission may take place, and an inactive state during which the device saves energy. Overall, the authors employ Q-learning to optimize energy consumption–given as the ratio of time spent active–and throughput by altering the time spent active in each frame.

There are some commonalities between each of the above duty cycling works. First, each of these consider a duty cycle that consists of only an "*on*" and "*off*" state. This allows for a more simple modeling of the system, for example the energy efficiency can be simply defined as the ratio of time spent on to the time spent off. Our work parts from the existing literature in this sense by considering a duty cycling dynamic which contains more than these two states. We model DRX as having four unique macro sleep states (radio resource control (RRC) connected, DRX *on*, DRX *off*, and PSM), each with their own potentially nondeterministic amount of energy consumed and varying effects on the latency of packets. Also, many works in this area do not consider the continuous tradeoff that exists between latency and energy efficiency in duty cycling problems ([14] is an exception, though only considers an *on* and *off* state). In our work, we model this tradeoff exactly and thoroughly study the interplay between energy efficiency, latency, and the sleep duration in the DRX mechanism.

There are two primary sleep modes for UEs in C-IoT: 1) DRX and 2) PSM. Sultania et al. [19] introduced both of these mechanisms, provide an analytical model for each, and evaluate the performance of both mechanisms through their implementation in network simulator 3 (NS3) using the NB-IoT protocol.

In [20], the DRX mechanism is evaluated through a cross-layer analytical model with traffic distributed according to a Poisson process. Results show that the introduction of the DRX mechanism yields a considerable improvement (up to three times) in the energy efficiency of the device. Further, results show that, for given DRX timers, there is a certain traffic load at which the energy efficiency improvement of the mechanism is optimum. This illustrates the importance of choosing DRX timers according to traffic load to achieve the best energy efficiency and delay results.

Numerous attempts have been made to study and optimize DRX. Moradi et al. [21] expanded the typical 3-state semi-Markov model of DRX to a five-state semi-Markov model to study DRX in Device to Device (D2D) communications. In the typical three-state model, the device can be either active, in short DRX cycles, or in long DRX cycles. Moradi et al. introduced two additional states, i.e., discovery states for both the short and long cycle states. In the discovery states, UEs monitor the physical sidelink discovery channel (PSDCH) where other devices can send discovery messages. In the case where a discovery message is received, the devices can then establish a link. These additional states are added between the short/long cycles and the active state, where the device monitors the channel for discovery messages. After this, the device continues DRX operation as normal, checking for the existence of DL data and waking up/sleeping accordingly. This model, while useful, only offers the modeling of DRX using two possible DRX *off* timers: 1) short and 2) long. Our work, in contrast, offers a method in which the DRX *off* timers can be selected from a finite set of possible timers.

Moradi et al. [22] considered DRX in a video streaming environment where the channel capacity is changing over time. They utilize a channel prediction method in order to minimize the energy consumption of the UE while simultaneously preventing significant receiver buffer underflows, which would indicate a significant incurred delay. The authors present their results as separate plots of energy versus video bitrate and number of buffer underflows versus video bitrate, but they do not closely examine the relationship between energy and the number of buffer underflows.

Zhou et al. [23] proposed an actor–critic algorithm to improve the latency-energy tradeoff that exists in DRX. The authors consider a modified DRX mechanism consisting of four states: 1) continuous reception; 2) *on* duration of DRX cycle; 3) *off* duration of DRX cycle; and 4) RRC Idle. The algorithm learns over time through the modification of the timers that facilitate state transitions (e.g., *on* duration of DRX cycle). The authors show how their algorithm performs in terms of both energy and delay compared to simpler DRX implementations; however, they do not provide a detailed study on the tradeoff between energy and delay that is inherent to DRX. In contrast to this work, we formulate the DRX problem in such a way where we can tune the energy-delay tradeoff to more closely examine the complex relationship between the DRX *off* timer, energy, and delay.

Koc et al. [24] modeled DRX in an LTE-A network according to a semi-Markov process. In their model, they consider two types of traffic: 1) active traffic, where there are frequent packet calls and 2) background traffic, where there are less frequent transmissions (e.g., updates from a weather app). The authors propose a weighted sum approach in which they attempt to minimize energy consumption while also minimizing delay. In deployment, the authors propose two modes of operation – one for each type of traffic. During active traffic, all weight is placed on delay, thus the objective is to minimize delay subject to an energy constraint. During background traffic, all weight is placed on energy, thus the objective is to minimize energy subject to a delay constraint. While the approach of this work is similar to ours in the sense that the problem is formulated as the minimization of a weighted sum of energy and delay, this work only considers the two extreme cases where all priority is placed on either energy or delay. In contrast to this, our work more closely examines the energy-delay tradeoff by varying the priority on energy/delay over a range of possible values. This allows us to achieve a more complete understanding of the complex relationship between the DRX *off* timer, energy, and delay.

In our work, we formulate the DRX problem as a multiobjective optimization where the objective functions are delay and energy. There are a number of approaches used in the literature to solve such a problem. One of the most popular methods is linear scalarization, applied in [25] and [26]. Linear scalarization is the process of translating the vector of

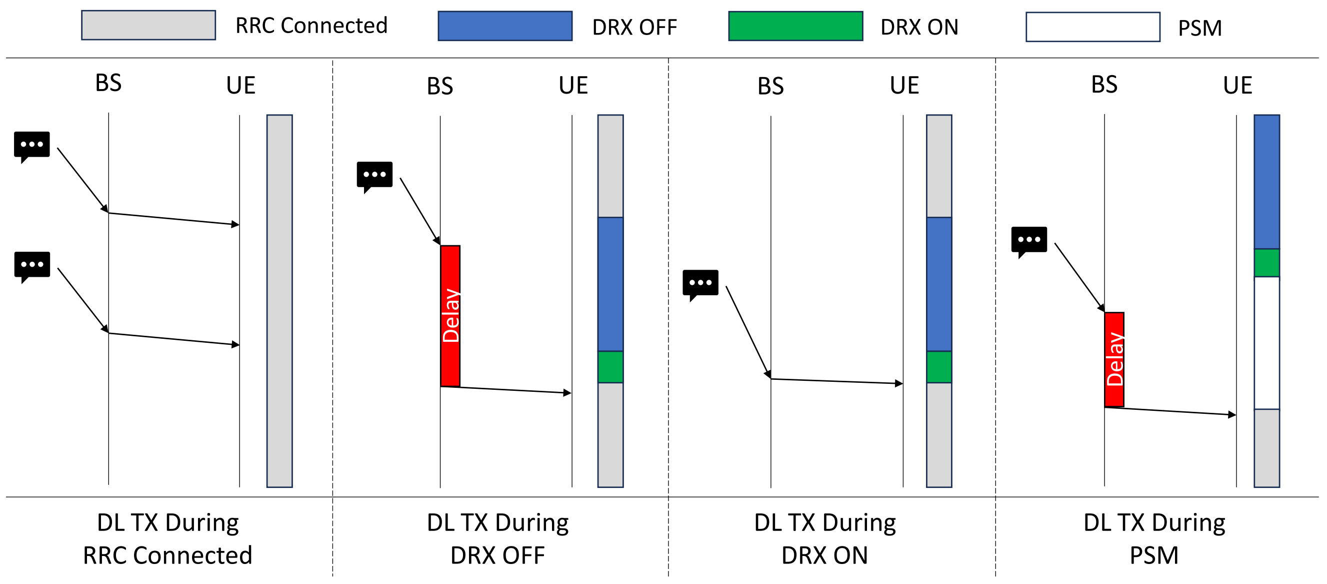

Fig. 1. Timing diagram showing data transmission and delays in each DRX macro state.

objective functions to a scalar value through a weighted sum of the vector elements. Scalarization opens the doors to more straightforward, well known solutions, such as value iteration.

In contrast to the previous work, this work formulates the DRX mechanism in full as a single discrete MDP. In doing so, we are able to directly solve the MDP through dynamic programming. In much of the available literature, the energy-delay tradeoff in DRX is examined through setting a delay constraint and attempting to minimize energy consumption subject to that constraint. However, in this work, we define a single continuous variable which can be used to tune this tradeoff in either direction, i.e., varying emphasis can be placed on either energy or delay. In doing so, this allows for a much wider range of operating points. To the best of our knowledge, this inherent tradeoff between energy and delay in DRX has not been thoroughly studied. Finally, we propose a practical deployment of our results in C-IoT networks where the traffic intensity is either unknown or time varying.

III. Problem Formulation

In what follows, we first provide a high-level overview of the DRX mechanism. Next, we describe the general structure and objectives of MDPs. Then, the remainder of the section is dedicated to rigorously defining how we model DRX as an MDP. That is, we define the state space, the action space, the transition probability function (TPF), and the cost function. Finally, we introduce the method we use to solve the MDP – the DRX-aware value iteration algorithm. A list of the notation used throughout is given in Table I.

A. DRX Overview

In many instances, devices in IoT networks can go extended periods of time without the existence of DL traffic. In such instances, devices would waste considerable energy by unnecessarily monitoring DL control channels continuously.

TABLE I
List of Notation

S_{RRC}	RRC Connected State
S_{ON}	DRX *On* State
S_{OFF}	DRX *off* State
S_{PSM}	PSM State
T_{OFF}	DRX *off* Timer
T^*_{OFF}	Optimal DRX *off* Timer
T^{max}_{OFF}	Maximum possible DRX *off* Timer
p	Packet Arrival Probability
$P(s', a, s)$	Transition Probability Function
$C(s, a)$	Cost Function
$Q(s, a)$	Action-Value Function
$V(s)$	Value Function
T_{RRC}	RRC Inactivity Timer
T_{ON}	DRX On Duration
T_{PSM}	PSM Timer Duration

DRX offers a solution to this problem of wasted energy. A system model diagram of the DRX mechanism is given in Fig. 1, where the various modes of operation are illustrated and transmission timings are shown. Additionally, a timing diagram of the DRX mechanism is illustrated in Fig. 2. In DRX, if the device has gone a certain period of time without having received a packet, it will enter DRX cycles. Each of these DRX cycles consists of an *off* and *on* period. When the device is *off*, it will minimize the activity of its RF circuitry to not waste energy monitoring channels. During this period, the device is saving energy, but it is unable to receive DL packets. During the *on* period, the device will consume energy to wake up and check the radio control channel, to see if there are any incoming DL packets. If there are none, the device will go back into the *off* mode, and these cycles will continue. However, if there are any packets, the device will wake up fully, and exit these DRX cycles. A list of all possible timer values that facilitate these transitions can be found in [27]. While there are also many other timers listed, such as the RRC release timer, DRX *on* timer, etc., this work only focuses on the DRX *off* timer.

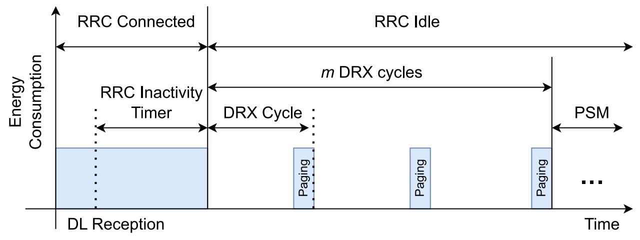

Fig. 2. DRX timing diagram.

PSM is an additional sleep mode, which allows the device to sleep for much longer periods. PSM is triggered by the device going through m consecutive DRX cycles without any DL packets. In PSM, the device saves energy by turning off its RF circuitry for an extended period of time, but is unreachable by the network. Eventually, the device will wake up from PSM and go back to regular operation.

Fig. 2 illustrates the operation of DRX according to when packets arrive. When a DL packet arrives in the RRC connected state, there is no added delay due to sleeping, since the packet can be immediately forwarded from the BS to the UE. When a packet arrives in DRX off, this packet must be queued at the BS until the UE exits DRX *off*, observes the packet in DRX on, and transitions to RRC connected. The added delay due to this is indicated in red. When a packet arrives in DRX on, the UE can immediately transition to RRC connected and receive the packet. Finally, in PSM, the packet needs to be queued for the full duration of PSM until the UE enters RRC connected again, incurring an additional delay.

B. Markov Decision Processes

To model the DRX mechanism, an MDP is introduced. An MDP is used to model an agent making decisions in a stochastic environment in which immediate decisions impact the current and future costs. MDPs are a common choice to model and analyze complex systems [23], [28], [29]. We will consider a discrete-time MDP with uniform time steps Δt. In each time step, the agent first observes the current state $s \in \mathcal{S}$. The agent then takes action $a \in \mathcal{A}(s)$ accordingly, where $\mathcal{A}(s)$ denotes the set of available actions in state s. Finally, the environment stochastically transitions to state $s' \in \mathcal{S}$. The probabilities of transitions between states are defined by the following TPF P:

$$P\big(s', a, s\big) = \Pr\big[s'|s, a\big], s, s' \in \mathcal{S}, a \in \mathcal{A}(s). \tag{1}$$

The fourth component of an MDP is the cost function $C(s, a)$. This cost function measures how "expensive" the action a was in state s. The fifth and final part of an MDP is the discount factor $\gamma \in [0, 1)$. γ defines how much the model cares about future costs. When γ is zero, all the weight is placed on immediate cost while as γ approaches one, more emphasis is placed on anticipated future costs. Details about the states, actions, and TPF in the proposed MDP are provided in the sections below.

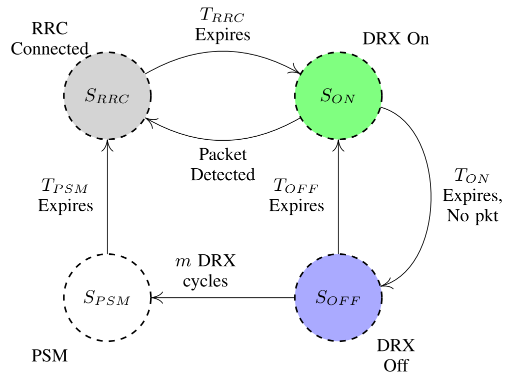

Fig. 3. DRX state transitions overview, showing the events that trigger state transitions.

Overall, in an MDP, we look to minimize the infinite horizon discounted sum of costs, specifically

$$\min_{\pi} \sum_{t=0}^{\infty} \gamma^t C(s_t, \pi(s_t)) \tag{2}$$

where $\pi : \mathcal{S} \rightarrow \mathcal{A}$ denotes the decision policy, which maps states to actions.

In the remainder of this section, we describe how DRX is modeled by mapping the DRX mechanism onto each of these four MDP components. Namely, in Section III-C1 we define the state space, in Section III-C2 we define the actions the agent can take, in Section III-C3 we provide the TPF, in Section III-C4 we introduce our problem's cost function, and finally in Section III-C5 we describe our low-complexity "DRX-aware" value iteration algorithm that will be used to solve the MDP.

C. DRX System Model

In this section, we describe how we model DRX as an MDP through the four elements of an MDP: 1) the state space; 2) the action space; 3) TPF; and 4) the cost function. We then introduce the method we use to solve the MDP – DRX-aware value iteration algorithm.

1) States: Similar to the model in [23], our base model of DRX will consist of four primary states, i.e., RRC Connected (S_{RRC}), DRX on (S_{ON}), DRX off (S_{OFF}), and PSM (S_{PSM}), as illustrated in Fig. 3 where each of these four states are color coded. In S_{RRC}, the device is fully awake and can transmit or receive packets at any time. In this state, the device is sacrificing its energy for a potential improvement in network performance, specifically latency, as any packets that arrive in this state will be transmitted immediately and therefore have zero delay. If no packets are received in the timer duration T_{RRC}, the state will transition to the DRX *on* state, denoted by S_{ON}, and will begin a series of DRX cycles. However, if a packet is received in S_{RRC} at any time, T_{RRC} is reset.

The second state, S_{ON}, is illustrated in the top right of Fig. 3. In this state, the device is in the awake part of its DRX cycles, and is able to receive a packet at any time during this state. Similar to the RRC Connected state, in this state the device is sacrificing energy for a potential improvement in network performance. If a packet is received during S_{ON}, the

TABLE II
DRX STATE ARRAY FOR S_{RRC}

	Timer State					
	$5\Delta t$	$4\Delta t$	$3\Delta t$	$2\Delta t$	$1\Delta t$	$0\Delta t$
S_{RRC} states:	$S_{RRC}, 5, 0$	$S_{RRC}, 4, 0$	$S_{RRC}, 3, 0$	$S_{RRC}, 2, 0$	$S_{RRC}, 1, 0$	$S_{RRC}, 0, 0$

state transitions back to S_{RRC}. However, if no packet has been received, the device will transition to PSM if m DRX cycles have elapsed, and transition to S_{OFF} (DRX *off*) otherwise.

The third state, S_{OFF}, is illustrated in the bottom right of Fig. 3. In this state, the device is in its *off* period of the DRX cycles. The device is consuming a reduced amount of energy, but it cannot be reached by the network, so any DL packet that arrives in this state will have an added delay. In this state, the device will stay asleep until the timer T_{OFF} expires, in which case it will transition back to S_{ON}.

The final state, S_{PSM}, is illustrated in the bottom left of Fig. 3. In this state, the device sleeps for a long period of time. Similar to S_{OFF}, in PSM, the device is saving energy and hoping that its network performance does not suffer. After the timer T_{PSM} expires, the device wakes back up and transition to state S_{RRC}, where the entire process can begin again.

We define $\mathcal{S}_m$ to be the set of all possible "macro" states, i.e., $\mathcal{S}_m = \{S_{\text{RRC}}, S_{\text{ON}}, S_{\text{OFF}}, S_{\text{PSM}}\}$. Each of these macro states is composed of a number of substates, as shown in Table II. To define these substates, a couple of additional variables must be considered. The first addition is a timer state that will help facilitate the transitions between these main states. The set of possible timer values $t \in \mathcal{T}$ depends on the current DRX state as follows:

$$t \in \begin{cases} [0, T_{\text{RRC}}], & \text{if } s_m = S_{\text{RRC}} \\ [0, T_{\text{ON}}], & \text{if } s_m = S_{\text{ON}} \\ \left[0, T_{\text{OFF}}^{\max}\right], & \text{if } s_m = S_{\text{OFF}} \\ [0, T_{\text{PSM}}], & \text{if } s_m = S_{\text{PSM}} \end{cases} \tag{3}$$

where $T_{\text{OFF}}^{\max}$ is the largest possible DRX *off* timer. It is worth noting that all timer values specified can be only integers. We will slightly abuse notation and let the closed interval $[x, y]$ denote the integers $x, x+1, \ldots, y$ and the open interval (x, y) denote the integers $x+1, x+2, \ldots, y-1$.

The second addition is a Boolean packet indicator state, indicating the existence of a packet, i.e., this indicator will be 1 if there is a packet waiting and 0 otherwise. Note that, this indicator can only be 1 in states where immediate reception of the packet is not possible (S_{OFF} and S_{PSM}); hence

$$\mathcal{S}_{pkt} \in \begin{cases} \{0, 1\}, & \text{if } s_m \in \{S_{\text{OFF}}, S_{\text{PSM}}\} \\ \{0\}, & \text{if } s_m \in \{S_{\text{RRC}}, S_{\text{ON}}\}. \end{cases} \tag{4}$$

It is possible that multiple packets arrive during a single *off* duration. In this case, all packets would need to be buffered at the BS until the device wakes. In practice, this is quite unlikely as devices operating with DRX tend to communicate infrequently. Currently, as to not significantly expand the state space unnecessarily, our model assumes that at most one packet will be waiting in the BS at any given time.

The resulting state space $\mathcal{S}$ is then defined as a subset of the Cartesian product of the macro state, timer state, and packet indicator state

$$\mathcal{S} \subset \mathcal{S}_m \times \mathcal{T} \times \mathcal{S}_{pkt}. \tag{5}$$

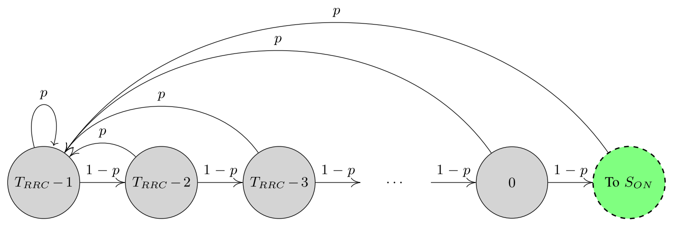

Fig. 4. RRC connected state transitions.

It is important to note that not all elements resulting from this Cartesian product are actually possible. For instance, assuming $T_{\text{ON}} < T_{\text{RRC}}$, then the state $s = (S_{\text{ON}}, T_{\text{RRC}} - 1, 0)$ is a state within this Cartesian product, but is not reachable. An example of a possible sequence of the resulting reachable states is given in Table II. The sequence begins in S_{RRC} timer state 5. In each subsequent time step, the timer step is decremented by 1.

After all these considerations, the size of the state space is given as follows:

$$|\mathcal{S}| = T_{\text{RRC}} + T_{\text{ON}} + 2\left(T_{\text{OFF}}^{\max} + T_{\text{PSM}}\right). \tag{6}$$

2) Actions: The action considered in this model is the length of time the device spends in the *off* period of its DRX cycles, i.e., T_{OFF}. We define this action space $\mathcal{A}(s)$ to be a discrete set of predetermined timer values whose entries depend on the current state. A discrete action space is selected because 3GPP standards [27] define a discrete and finite set of possible DRX timers. Recall that this action is only taken immediately prior to switching to S_{OFF}. Thus, the action space only contains possible selections at this specific state, i.e., $s = (S_{\text{ON}}, 0, 0)$. For all other states, $\mathcal{A}(s)$ is the empty set

$$\mathcal{A}(s) \in \begin{cases} \{a_1, a_2, a_3, \ldots, T_{\text{OFF}}^{\max}\}, & \text{if } s = (S_{\text{ON}}, 0, 0) \\ \emptyset, & \text{otherwise} \end{cases} \tag{7}$$

where a_i is a valid timer selection and $a_i < a_{i+1}$. It should be noted that in all states where the action space is the empty set, our MDP is reduced to a Markov reward process [30].

3) Transition Probability Function: Now that the state space and the transitions between states have been modeled, all that is needed before we arrive at the TPF is a model of the incoming traffic. To this end, we use a Bernoulli-distributed traffic model [31], [32]. In each time step, there is a probability p of there being an incoming packet. This distribution keeps the model simplest, as the probability of a packet arrival in a given time slot does not vary with time. This results in a TPF that also does not vary with time.

With this traffic distribution defined, the TPF can be constructed. The high level view is illustrated through the macro state transition diagram in Fig. 3. Note that, in these state transition diagrams, states with a dashed border describe a general macro state, while a solid border indicates a specific state.

Next, we will go through the transition probabilities within each of these high level states, beginning with S_{RRC}, illustrated in Fig. 4. The system is initialized in this state, with a timer

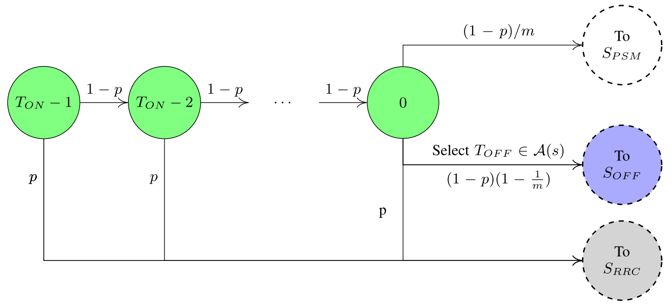

Fig. 5. DRX *on* state transition probabilities.

state of $T_{\mathrm{RRC}}-1$. With each time step, the timer state is decremented with the probability that no packet arrives $1-p$, and gets reset to timer state T_{RRC} with the probability of a packet arrival p. In essence, if the system is in timer state t, it will move to timer state $t-1$ with probability $1-p$, and will move to timer state T_{RRC} with probability p. Once the timer state reaches 0, it will instead transition to timer state T_{ON} of DRX *on* with probability $1-p$. Formally, we have

$$P\big((S_{\mathrm{RRC}}, t-1, 0), a, (S_{\mathrm{RRC}}, t, 0)\big) = \begin{cases} 1-p, & \text{if } t \in [1, T_{\mathrm{RRC}}) \\ 0, & \text{if } t = 0 \end{cases} \tag{8}$$

$$P\big((S_{\mathrm{RRC}}, T_{\mathrm{RRC}}-1, 0), a, (S_{\mathrm{RRC}}, t, 0)\big) = p \quad t \in [0, T_{\mathrm{RRC}}) \tag{9}$$

$$P\big((S_{\mathrm{ON}}, T_{\mathrm{ON}}-1, 0), a, (S_{\mathrm{RRC}}, 0, 0)\big) = 1-p. \tag{10}$$

These stochastic timer transitions occur similarly in the state S_{ON}, as illustrated in Fig. 5, i.e., the timer is decremented with probability $1-p$. The only difference is that with the probability of a packet arrival p the state will transition back to timer state $T_{\mathrm{RRC}}-1$ in the macro state S_{RRC}. Recall that at the end of the DRX *on* period, it will transition to S_{PSM} if m DRX cycles have elapsed, and will move to S_{OFF} otherwise. If we were to model this exactly, each timer state in DRX cycles would have to be replicated m times, expanding the state space significantly. In general, the computational complexity of value iteration is given as $\mathcal{O}(|\mathcal{S}|^2|\mathcal{A}|)$,[1] quadratic in the size of the state space. Replicating the DRX states for each unique cycle would increase the number of states by $(m-1)(T_{\mathrm{ON}}+2T_{\mathrm{OFF}})$ and therefore increase the complexity overall. To avoid this significant increase in computational complexity, the transition to PSM is modeled stochastically. When the system reaches timer 0 of S_{ON}, assuming there is no packet arrival, it will transition to S_{PSM} with probability $1/m$ and will transition to S_{OFF} with probability $1-1/m$. Thus, the transition to PSM is approximated as being geometrically distributed with an expectation of m trials. The transition probabilities in S_{ON} are given by

$$P\big((S_{\mathrm{ON}}, t-1, 0), a, (S_{\mathrm{ON}}, t, 0)\big) = \begin{cases} 1-p, & \text{if } t \in [1, T_{\mathrm{ON}}) \\ 0, & \text{if } t = 0, \end{cases} \tag{11}$$

$$P\big((S_{\mathrm{RRC}}, T_{\mathrm{RRC}}-1, 0), a, (S_{\mathrm{ON}}, t, 0)\big) = p \quad t \in [0, T_{\mathrm{ON}}) \tag{12}$$

[1]In our model, this complexity is actually slightly less since an action is only taken in one state.

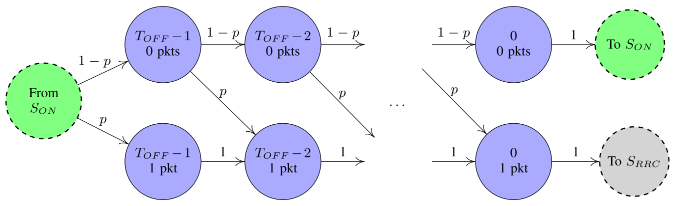

Fig. 6. DRX *off* state transition probabilities.

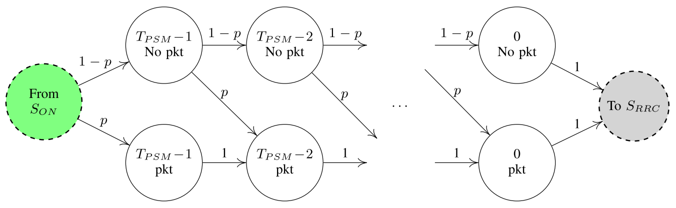

Fig. 7. PSM state transition probabilities.

$$P\big((S_{\mathrm{OFF}}, a-1, 0), a, (S_{\mathrm{ON}}, 0, 0)\big) = \frac{(1-p)(m-1)}{m} \tag{13}$$

$$P\big((S_{\mathrm{PSM}}, T_{\mathrm{PSM}}-1, 0), a, (S_{\mathrm{ON}}, 0, 0)\big) = \frac{1-p}{m}. \tag{14}$$

Further, this final timer state of DRX *on* is the state at which T_{OFF} will be selected. To model this, when the system does transition to S_{OFF}, it will transition to the corresponding timer value, as illustrated in Fig. 6. That is, if a DRX *off* timer T_{OFF} is selected, it will transition to timer state $T_{\mathrm{OFF}}-1$ in S_{OFF}. From there, the timer state will be decremented each time step until the timer state reaches zero, at which point it will transition back to S_{ON} if there is no packet and to S_{RRC} otherwise.[2] In each time step of S_{OFF}, if there is not yet a buffered packet, there is a probability p of a packet arrival, thus setting the packet indicator state to 1 in the next time step. If the packet indicator is already 1, then the timer simply decrements in the following time step. The transition probabilities in S_{OFF} are given by

$$P\big((S_{\mathrm{OFF}}, t-1, 0), a, (S_{\mathrm{OFF}}, t, 0)\big) = \begin{cases} 1-p, & \text{if } t \in [1, T_{\mathrm{OFF}}^{\max}) \\ 0, & \text{if } t = 0, \end{cases} \tag{15}$$

$$P\big((S_{\mathrm{OFF}}, t-1, 1), a, (S_{\mathrm{OFF}}, t, 0)\big) = \begin{cases} p, & \text{if } t \in [1, T_{\mathrm{OFF}}^{\max}) \\ 0, & \text{if } t = 0, \end{cases} \tag{16}$$

$$P\big((S_{\mathrm{OFF}}, t-1, 1), a, (S_{\mathrm{OFF}}, t, 1)\big) = \begin{cases} 1, & \text{if } t \in [1, T_{\mathrm{OFF}}^{\max}) \\ 0, & \text{if } t = 0, \end{cases} \tag{17}$$

$$P\big((S_{\mathrm{ON}}, T_{\mathrm{ON}}-1, 0), a, (S_{\mathrm{OFF}}, 0, 0)\big) = 1 \tag{18}$$

$$P\big((S_{\mathrm{RRC}}, T_{\mathrm{RRC}}-1, 0), a, (S_{\mathrm{OFF}}, 0, 1)\big) = 1. \tag{19}$$

The PSM state transitions are illustrated in Fig. 7. In S_{PSM}, the timer state is decremented with each time step until timer state zero is reached, at which point the state is transitioned back to S_{RRC}. The transition probabilities in S_{PSM} are given by

[2]In practice, there would need to be a brief transition to S_{ON} regardless of the existence of a packet. However, it can be assumed without much loss that if there is a packet, there can be a direct transition from S_{OFF} to S_{RRC}.

$$P\big((S_{\text{PSM}}, t-1, 0), a, (S_{\text{PSM}}, t, 0)\big) = \begin{cases} 1-p, & \text{if } t \in [1, T_{\text{PSM}}) \\ 0, & \text{if } t = 0, \end{cases} \tag{20}$$

$$P\big((S_{\text{PSM}}, t-1, 1), a, (S_{\text{PSM}}, t, 0)\big) = \begin{cases} p, & \text{if } t \in [1, T_{\text{PSM}}) \\ 0, & \text{if } t = 0, \end{cases} \tag{21}$$

$$P\big((S_{\text{PSM}}, t-1, 1), a, (S_{\text{PSM}}, t, 1)\big) = \begin{cases} 1, & \text{if } t \in [1, T_{\text{PSM}}) \\ 0, & \text{if } t = 0, \end{cases} \tag{22}$$

$$P\big((S_{\text{RRC}}, T_{\text{RRC}}-1, 0), (S_{\text{PSM}}, 0, x)\big) = 1. \tag{23}$$

4) Immediate Cost: The immediate cost $C(s)$ incurred in state s is defined by considering a weighted sum of delay and energy costs as follows:

$$C(s) = D(s) + \lambda E(s) \tag{24}$$

where $D(s)$ is the delay cost in state s, $E(s)$ is the energy cost in state s, and λ is a coefficient that adjusts the weight placed on energy as opposed to delay. For example, when $\lambda = 0$ the UE places all priority on reducing latency no matter the cost in terms of energy. In practice, the value of λ will be adjusted according to the application using DRX. While an application, such as smart metering may be able to use higher values of λ to achieve greater energy savings, applications, such as health monitoring systems may wish to employ a lower value of λ to decrease the risk of significant delays. In terms of the physical interpretation of delay and energy, the considered delay for a given packet is computed as the number of time steps the packet has been waiting at the BS buffer while the UE is asleep. In essence, $D(s) = 1$ every time step that the packet is waiting at the BS buffer. For the energy consumption, we assume a certain amount of energy ϵ_0 is consumed per time step in S_{RRC} and S_{ON}, no energy is consumed in S_{PSM}, and a fraction $\alpha \in [0, 1]$ of ϵ_0 is consumed in S_{OFF}. It is important to note that the scale of energy and delay relative to each other is not important since we consider a weighted sum of each metric, where the weight can adjust the importance of one metric versus the other.

The values of $D(s)$, $E(s)$, and $C(s)$ for each state are given in Table III. In S_{RRC} and S_{ON}, the delay cost is always 0 and the energy cost is always ϵ_0. This is because in these states, the UE is consuming maximum energy to stay awake and minimize delay. In S_{OFF}, the energy cost is always

$$E(s) = \alpha\epsilon_0 \tag{25}$$

where ϵ_0 is the energy consumed in S_{RRC} and α is the fraction of the energy ϵ_0 consumed in S_{OFF}. The delay cost in this state is 0 when there is no packet waiting and 1 when there is a packet waiting. Similarly, in S_{PSM}, there is no cost associated with energy loss, and a delay cost of 0 when there is no packet waiting and 1 when there is a packet waiting.

Note that, the immediate cost function defined in (24) does not depend on the action a, so it will simply be denoted as $C(s)$. Specifically, the calculation of this function amounts to a lookup in Table III based on the current state. While the action does not affect the immediate cost, it does affect the probabilities of visiting certain states in the future. Since the cost function depends on the state, the action indirectly affects future observed costs. More specifically, in our problem, the selection of T_{OFF} does not immediately influence the cost. Instead, it influences how long the device will sleep in the future, and thus influences the expected future costs.

It should be noted that this problem formulation can be achieved through two distinct methods. The first method is to first construct a constrained MDP, which then can be translated to an unconstrained MDP using the Lagrangian multiplier method found in [33]. The second method involves constructing the problem as a multiobjective optimization, where the objective functions are delay and energy. Then, through linear scalarization, a single objective optimization is reached.

5) DRX-Aware Value Iteration Algorithm: Now, all the necessary components of the MDP have been constructed and the DRX model is complete. Next, the optimal actions need to be found. To do this, value iteration is employed. Value iteration is described in [30].

The value iteration algorithm given in Algorithm 1 takes as an input the MDP, i.e., $\mathcal{S}$, $\mathcal{A}$, $P(s', a, s)$, $C(s)$, and γ. As an output, the algorithm provides two functions: 1) the action-value function $Q(s, a)$, which tells us how good or bad it is to take action a in state s and then follow the optimal policy π^* thereafter and 2) the value function $V(s)$, which tells us how good or bad being in state s is assuming the optimal policy π^* is followed. The final output is the optimal policy π^*. This

TABLE III
IMMEDIATE COST TABLE

State	Packet Waiting $D(s)$	Packet Waiting $E(s)$	Packet Waiting $C(s)$	No Packet Waiting $D(s)$	No Packet Waiting $E(s)$	No Packet Waiting $C(s)$
S_{RRC}	0	ϵ_0	$\lambda\epsilon_0$	0	ϵ_0	$\lambda\epsilon_0$
S_{ON}	0	ϵ_0	$\lambda\epsilon_0$	0	ϵ_0	$\lambda\epsilon_0$
S_{OFF}	1	$\alpha\epsilon_0$	$1+\alpha\lambda\epsilon_0$	0	$\alpha\epsilon_0$	$\alpha\lambda\epsilon_0$
S_{PSM}	1	0	1	0	0	0

Algorithm 1 DRX-Aware Value Iteration Algorithm

```
Input: $S, A, P, C, \gamma$
Output: $Q, V, \pi$
    Initialization:
1:  $\gamma, \delta$
2:  $Q(s, a), V(s)$ arbitrarily
3:  $V_{old}$ s.t. $\max_s|V(s) - V_{old}(s)| > \delta$
4:  while $\max_s|V(s) - V_{old}(s)| > \delta$ do
5:      for $s \in \mathcal{S}$ do
6:          Define $\mathcal{S}' = \{s' \in \mathcal{S} | P(s', a, s) > 0\}$
7:          for $a \in \mathcal{A}(s)$ do
8:              $Q(s, a) \leftarrow C(s) + \gamma \sum_{s' \in \mathcal{S}'} P(s', a, s)V(s')$
9:          end for
10:         $V_{old}(s) \leftarrow V(s)$
11:         $V(s) \leftarrow \min_a Q(s, a)$
12:     end for
13: end while
14: return $Q, V, \pi$
```

optimal policy is simply the action with the lowest associated value in the state $s = (S_{\mathrm{ON}}, 0, 0)$.

After the two output functions are initialized to arbitrary values, the value iteration algorithm consists of two steps that are repeated for all possible states until an exit condition is met. In the first step in line 7 of Algorithm 1, a form of the Bellman's equation is used to update the action-value function for every possible action a, i.e.,

$$Q(s,a) \leftarrow C(s) + \gamma \sum_{s' \in \mathcal{S}} P\big(s', a, s\big) V\big(s'\big). \tag{26}$$

This equation consists of the summation of two parts. The first part is simply the immediate cost from the MDP model. The second part is a measure of expected *future* costs. This part is multiplied by a discount factor, $\gamma \in [0, 1)$, which quantifies how much the algorithm should care about the future.

In the second step of the algorithm given in line 9 of Algorithm 1, the value function is updated based on the current best action to take in each state, i.e.,

$$V(s) = \min_{a \in \mathcal{A}(s)} Q(s,a). \tag{27}$$

These two steps are repeated until the value function is relatively static for all states. This is checked after step 2 using the old and new value functions and a threshold δ, i.e.,

$$\text{stop if} \max_s |V_{\mathrm{new}}(s) - V_{\mathrm{old}}(s)| < \delta. \tag{28}$$

In the case of this problem, an action is only taken in the final timer state of DRX *on*, $s^* = (S_{\mathrm{ON}}, 0, 0)$. So, we only need to look at the optimal action in this state to determine the optimal timer: i.e.,

$$T^*_{\mathrm{OFF}} = \operatorname{argmin}_a Q\big(s^*, a\big). \tag{29}$$

Our improvement to this algorithm comes in line 6 of Algorithm 1, where we define $\mathcal{S}'$ to be the set of next states for which there is a nonzero transition probability from state s. In the case of our MDP, there are only at most two possible next states from each state, so $|\mathcal{S}'| \leq 2$. In the traditional value iteration, updating the value function consists of a summation over the expected value of all possible next states. By limiting this summation to only the set $\mathcal{S}'$ instead of $\mathcal{S}$, we significantly limit the computation required to update each state in each iteration.

IV. Analytical Model

In this section, we develop an analytical model for the expected delay and energy in each of the macro states, given by D_{sm} and E_{sm}, respectively, where $sm \in \{RRC, ON, OFF, PSM\}$.

In S_{RRC} and S_{ON}, there is no delay, thus

$$\mathbb{E}[D_{\mathrm{RRC}}] = \mathbb{E}[D_{\mathrm{ON}}] = 0. \tag{30}$$

In S_{OFF} and S_{PSM}, a normalized delay of 1 is incurred each time step during which a packet is waiting at the BS. We define a random variable Y to be the number of time steps t it takes for the first packet to arrive at the BS during either S_{OFF} or S_{PSM}. Specifically, Y is a geometrically distributed random variable with success probability p. Further, a success at each time instance t has an associated delay. For instance, if a packet arrives in the first time step, then $Y = 1$ and the delay of this packet is equal to T_{OFF}. For any time t, the delay is given as

$$D_{\mathrm{OFF}} = \begin{cases} T_{\mathrm{OFF}} - Y + 1, & \text{if } 1 \leq Y \leq T_{\mathrm{OFF}} \\ 0, & \text{if } Y > T_{\mathrm{OFF}}. \end{cases} \tag{31}$$

The expectation of this delay is

$$\mathbb{E}[D_{\mathrm{OFF}}] = \sum_{t=1}^{T_{\mathrm{OFF}}} (T_{\mathrm{OFF}} - t + 1) P(Y = t) = \sum_{t=1}^{T_{\mathrm{OFF}}} P(Y \leq t). \tag{32}$$

Similarly, the expected delay for S_{PSM} is

$$\mathbb{E}[D_{\mathrm{PSM}}] = \sum_{t=1}^{T_{\mathrm{PSM}}} P(Y \leq t). \tag{33}$$

In S_{RRC}, the overall energy consumed over the duration of the state depends on how long the device spends in the RRC connected state. We define the random variable X to represent the amount of time spent in S_{RRC}. X is defined as a piecewise function as follows. If $X < T_{\mathrm{RRC}}$, then we can be certain the transition has not occurred yet. At $X = T_{\mathrm{RRC}}$, then the probability of transition is the probability no packets have arrived in the duration T_{RRC}. Finally, if $X > T_{\mathrm{RRC}}$, then it is necessary that the final $T_{\mathrm{RRC}} + 1$ time steps before transition are exactly a packet arrival followed by T_{RRC} consecutive time steps with no packet arrival. It is also necessary that the transition has not occurred at a prior time. Thus, the probability of X taking on a value i is defined as follows:

$$P(X = i) = \begin{cases} 0, & \text{if } i < T_{\mathrm{RRC}} \\ (1-p)^{T_{\mathrm{RRC}}}, & \text{if } i = T_{\mathrm{RRC}} \\ p(1-p)^{T_{\mathrm{RRC}}}\Big[1 - \sum_{z=1}^{i - T_{\mathrm{RRC}} - 1} P(X = z)\Big], & \text{if } i > T_{\mathrm{RRC}}. \end{cases} \tag{34}$$

Now that we have derived a random variable describing the time spent in S_{RRC}, to get the expected energy consumption in the state we need only to multiply the expectation of this random variable by the energy consumed in each time step of S_{RRC}. We should note that this expectation is calculated numerically by summing over a sufficiently large range of i as follows:

$$\mathbb{E}[E_{\mathrm{RRC}}] = \epsilon_0 \mathbb{E}[X] = \epsilon_0 \sum_{i=0}^{\infty} i P(X = i). \tag{35}$$

For each of the other macro states, this calculation is simpler as the duration of the macro state is deterministic. For each of these macro states, the expected energy consumption is simply the duration of the macro state multiplied by the energy consumed per time step in that macro state. The expected energy consumption in these states is thus

$$\mathbb{E}[E_{\mathrm{ON}}] = T_{\mathrm{ON}} \epsilon_0 \tag{36}$$

$$\mathbb{E}[E_{\mathrm{OFF}}] = T_{\mathrm{OFF}} \alpha \epsilon_0 \tag{37}$$

$$\mathbb{E}[E_{\mathrm{PSM}}] = 0. \tag{38}$$

TABLE IV
LIST OF SIMULATION PARAMETERS

Parameter	Value
T_{RRC}	100 ms
T_{ON}	30 ms
T_{OFF}	{10, 20, ... 300} ms
T_{PSM}	150 ms
Number of DRX Cycles m	3
Number of States	103
Number of Actions	30
Δt	10 ms
Energy consumed in S_{RRC}, S_{ON} ϵ_0	1
α	0.1
p	0.05
λ	0.6

With these analytical models for the expected delay and energy in each macro state formulated, these models can be compared to our MDP by calculating the expected cost, specifically

$$\mathbb{E}[C_{sm}] = \mathbb{E}[D_{sm}] + \lambda\mathbb{E}[E_{sm}]. \tag{39}$$

We should note that this analytical model is only capable of analyzing the individual parts of the problem. In order to derive a complete analytical solution to the problem of optimizing the DRX sleep duration, it is also crucial to know the amount of time spent in each macro state relative to one another. The only way this information can be gathered is through steady-state analysis of the MDP.

V. RESULTS

In this section, we will first present, discuss, and analyze results obtained from solving our MDP with value iteration. Then, we present our simulation setup. Next, we present simulation results in order to validate our model. Further, we discuss two additional simulations: one to more closely examine the energy-delay tradeoff and another to compare stochastic and deterministic PSM transitions. Finally, we discuss a method to apply these results in an environment where traffic conditions are unknown, and present simulation results using this method.

A. System Parameters

The list of parameters used in all results unless specified otherwise is given in Table IV. The list of all possible values for such timers is given in [27]. The values we have chosen correspond to realistic system values based on this standard. The range of values used for T_{OFF} should be wide enough to be sure to contain the optimal timer and fine enough to accurately find the optimal timer. In this model, 30 values of T_{OFF} were used in a range from 10 to 300 ms at 10 ms intervals. It should be noted that this range can be extended arbitrarily toward infinity, since if there is no traffic in the network, it would be optimal for this timer to be infinite. However, in the interest of practicality and for the range of traffic used in this work, the timer range of 10 to 300 ms is adequate.

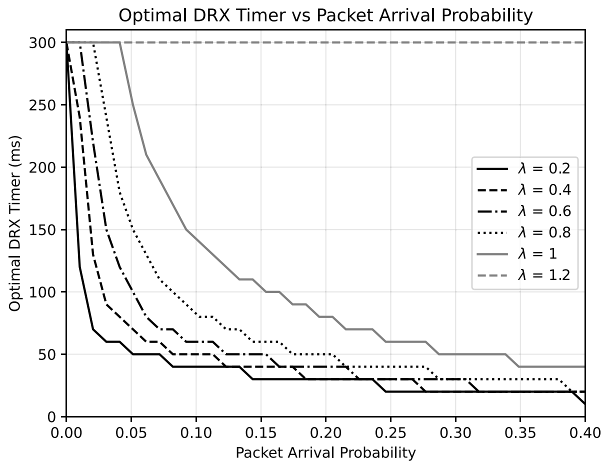

Fig. 8. Optimal DRX *off* timer versus p for different values of the tradeoff parameter λ.

For the base energy consumption considered in an *on* mode ϵ_0, we have selected this to be a value of 1 for simplicity. It is important to note that while we normalize this parameter, it can be selected as any arbitrary value based on the UE hardware itself. This is possible because the selected value is weighted according to the value of λ in the cost function, so if a different energy consumption is considered, λ can just be scaled accordingly. Finally, for each of the remaining parameters that dictate DRX operation, we simply selected moderate values based on the list of all possible values given in [27].

B. Value Iteration Results

In the first test, we varied the probability of a packet arrival in each time step p. With each p, we performed the value iteration algorithm and recorded the optimal DRX *off* duration T^*_{OFF}. This was done using varying values of λ. The data obtained is plotted in Fig. 8. Note that, the constant time step interval Δt was set to be equal to 10 ms, which we have determined is an adequate time granularity for the DRX system without adversely affecting the computational complexity of the model.

These results show exactly what was to be expected. For very low traffic rates (very small p), T^*_{OFF} becomes very large, trending toward the maximum allowed T_{OFF} at $p = 0$. This was expected because at very low traffic rates, the device can be in a sleep mode more often without risking too much network performance degradation. The opposite is also true: as p increases, the optimal DRX *off* timer becomes shorter. In this case, the system realizes that the probability of missing a packet when sleeping increases with increasing traffic rate, so it decides to stay awake more often.

Further, the effect of λ can be seen by comparing the different curves in Fig. 8. For lower values of λ corresponding to cases where less priority is placed on saving energy, T^*_{OFF} is very small, while when λ is increased, T^*_{OFF} is also increased.

Similar conclusions can be reached in Fig. 9, which shows the direct relationship between λ and T^*_{OFF}. Again, it can be seen that, with increasing λ, T^*_{OFF} increases exponentially.

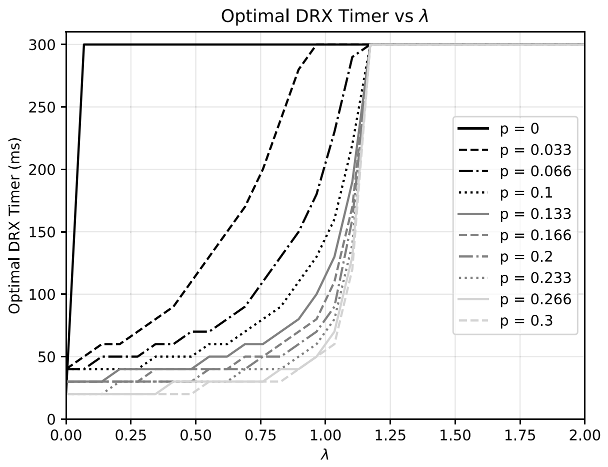

Fig. 9. Optimal DRX timer versus λ for different values of the packet arrival probability p.

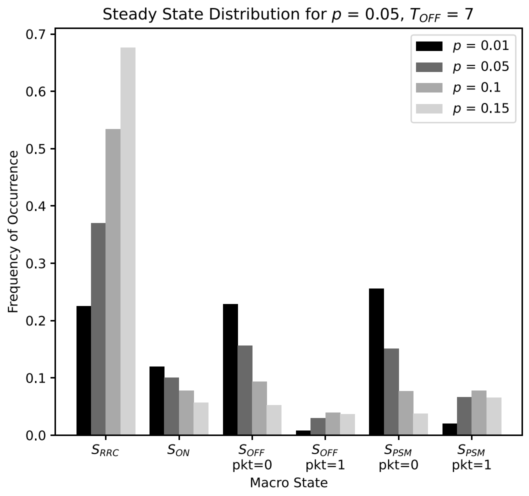

Fig. 10. Steady state distribution for different values of p and $T_{\text{OFF}} = 70$ ms.

Also, it can be seen by comparing the different curves corresponding to different values of p that lower values of p lead to greater values of T^*_{OFF}, which is consistent with Fig. 8. Further, it is clear from this figure that there is a λ at which $T^*_{\text{OFF}} = T^{\max}_{\text{OFF}}$, i.e., the optimal DRX *off* timer is the maximum allowed timer, regardless of the incoming traffic rate. This point occurs when it becomes more costly to be awake than to remain asleep, even when there is a buffered packet accumulating delay. This occurs when the cost of being awake is greater than the cost of sleeping with a packet waiting

$$\lambda\epsilon_0 \geq 1 + \lambda\alpha\epsilon_0 \tag{40}$$

where $\alpha\epsilon_0$ is the amount of energy consumed in S_{OFF} expressed as a fraction of the power ϵ_0 consumed in S_{RRC}. After solving for λ, we arrive at

$$\lambda \geq \frac{1}{\epsilon_0(1-\alpha)}. \tag{41}$$

C. Simulation Setup

In order to test our model, a simulation scenario was set up using Python. In this scenario, we consider a discrete time simulation in which there exists one BS and one UE that is using the DRX mechanism and employing a policy $\pi(s)$. In each discrete time step of the simulation, the BS first observes the current state consisting of the macro state, the current timer values, and whether or not there is a DL packet. Next, the cost is calculated from this observed state. Then, the BS will take action $a \in \mathcal{A}$ if the current state is the final timer state of S_{ON}. Finally, the state transitions stochastically based on the current state and the existence of a DL packet arrival. It is important to note that in the following simulations, stochastic transitions to PSM were employed unless specified otherwise.

To compare our simulations results with the results from our model, we calculate the expected cost of our model as

$$C = \sum_{s\in\mathcal{S}} \mu_s C(s) \tag{42}$$

where μ_s is the stationary distribution of the TPF P. More specifically, μ_s is calculated as

$$P\mu_s = \mu_s. \tag{43}$$

This stationary distribution can provide insight in the form of telling us how frequently each state is being visited. An example of this distribution for varying traffic is given in Fig. 10. In this example, the DRX *off* duration is set statically at 70 ms. Here, we can see that as p increases, more time is spent in S_{RRC} because each time a packet is received, the device will transition back to S_{RRC}. Conversely, as p decreases, more time is spent in S_{OFF} and S_{PSM} since lower traffic rates allow for larger expected interpacket arrival times leading to more time spent in off modes.

Finally, one additional consideration must be made prior to comparing simulation results with our model. In the model, we use a discount factor γ to calculate the infinite horizon discounted sum of costs, while this process is not done in the simulation. Simply averaging the observed simulated cost would therefore introduce a mismatch. To overcome this, the first-visit Monte Carlo method given in Algorithm 2 is used [28]. First, a simulation of n time steps was conducted, and the state visited at each time step $s(t)$ was recorded. Through this entire simulation, the action a is fixed. After the simulation, the discounted future costs of the first visit of each state was calculated. This process is shown in lines 8 through 13 of Algorithm 2. First, the first visit of state s is located, and the time at which this occurred is marked as time t. Next, for all times after t until the end of the simulation $t' \leq n$, the value of state s is updated according to the following equation:

$$V(s) \leftarrow V(s) + \gamma^{t'-t} C\big(s(t')\big) \tag{44}$$

where γ is the discount factor and $C(s(t'))$ is the cost of the state visited at time t'. Note that, after this value is computed, it will need to be normalized by a factor of $1-\gamma$ so we can directly compare values for different discount factors. After repeating this for all states, $V(s)$ is returned. This process was repeated for all $a \in \mathcal{A}$.

Algorithm 2 Value Function Approximation Algorithm

```
Input: $S, A, \gamma, C, \pi$
Output: $V$
    Initialization:
 1: $V(s)$ arbitrarily
 2: $t = 0$
 3: $s_0 = (S_{RRC}, T_{RRC}, 0)$
 4: while $t < n$ do
 5:     $a_t = \pi(s_t)$
 6:     $s(t) = s_t$
 7:     $s_t \leftarrow s_{t+1}$
 8: end while
 9: for $s \in \mathcal{S}$ do
10:     $t = \arg\min_t(s(t) == s)$
11:     for $t \leq t' \leq n$ do
12:         $V(s) += \gamma^{t'-t} C(s(t'))$
13:     end for
14: end for
15: return $(1-\gamma)V(s)$
```

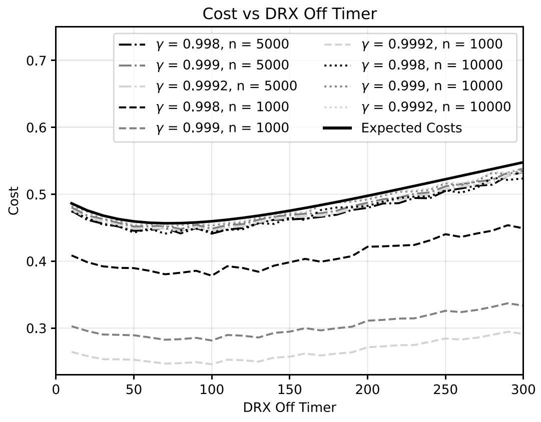

Fig. 11. Average observed cost versus policy for various γ and n.

D. Model Validation

The results of this simulation are provided in Fig. 11. Here, the value function approximation algorithm was conducted for various discount factors γ and simulation times n. It is clear from this figure that, when the simulation time is increased, the average cost tends toward the expected cost. It can also be seen that in general, when γ is increased, the average cost again tends toward the expected cost. From these two observations, it can be concluded that in an ideal simulation environment, i.e., γ infinitely close to 1 and $n = \infty$, the observed cost in the simulation would match the expected cost exactly, verifying the integrity of the model.

Additionally, we compare our value iteration results with that of the analytical model and the simulation. For each of the macro states, the average cost in each macro state was computed for the three methodologies used in this work – analytical, through the steady state distribution of the MDP, and via simulation. The resulting costs were plotted in Fig. 12, where a 95% confidence interval is shown for the simulation. Results show a high degree of similarity between all three approaches.

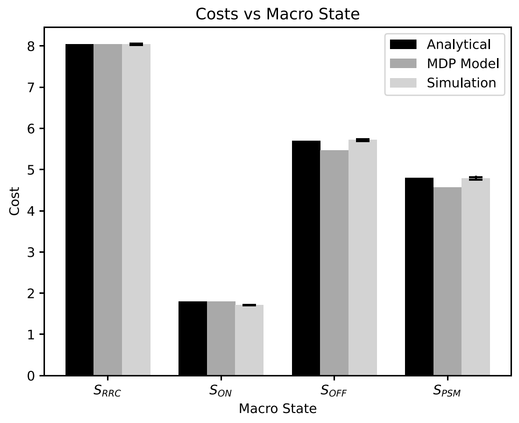

Fig. 12. Cost in each macro state for the analytical model, MDP, and simulation.

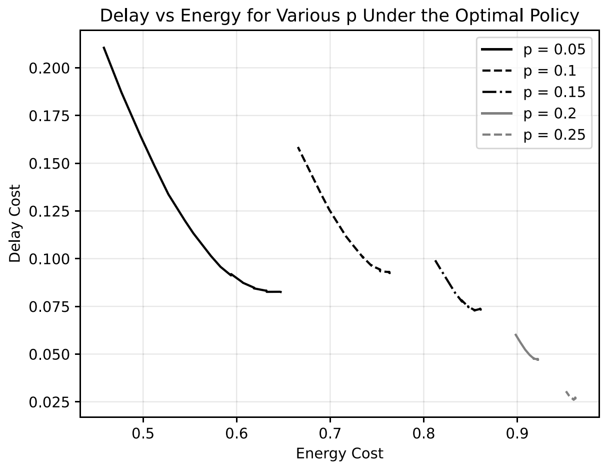

Fig. 13. Energy versus delay cost for varying values of λ between 0 and 1.

E. Energy-Delay Tradeoff

An additional discrete-time simulation was conducted to further explore the tradeoff between energy and delay. The results of this simulation are shown in Fig. 13, where each curve is a different value of p and the points making up each curve result from changes in λ. For each value of p, λ was varied between 0 and 1. For each pair of p, λ, the optimal timer T^*_{OFF} was gathered from the value iteration results. Using this timer, a simulation of 10000 time steps (i.e., 100 s) was conducted and the average energy costs and delay costs are recorded. As λ increases, more emphasis is placed on saving energy, thus T^*_{OFF} is increased. This reduces the energy cost while increasing the delay cost. When p is increased, the energy cost increases while the delay cost increases. This occurs because the device becomes less likely to enter an *off* state, resulting in more energy consumption in S_{RRC} and less opportunity to be asleep during packet arrivals.

F. Deterministic Versus Stochastic PSM Transitions

Finally, one more simulation was conducted using deterministic transitions to PSM. Recall that in our model and in the previous simulations, we employ stochastic transitions to PSM, i.e., at the end of each DRX cycle, there is a probability $1/m$ of transitioning to PSM. The comparison between these

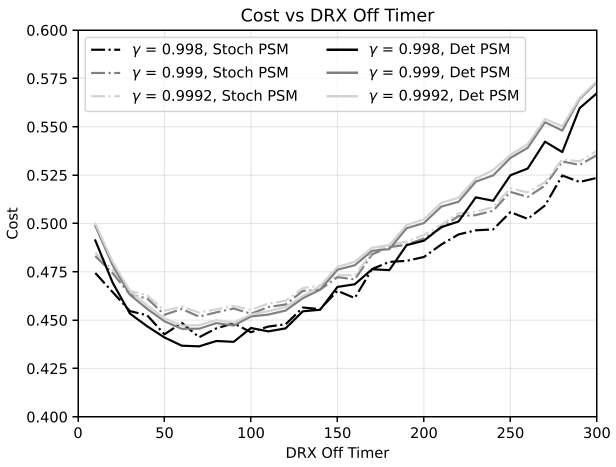

Fig. 14. Stochastic versus deterministic PSM transitions for different discount factors. $n = 10\,000$.

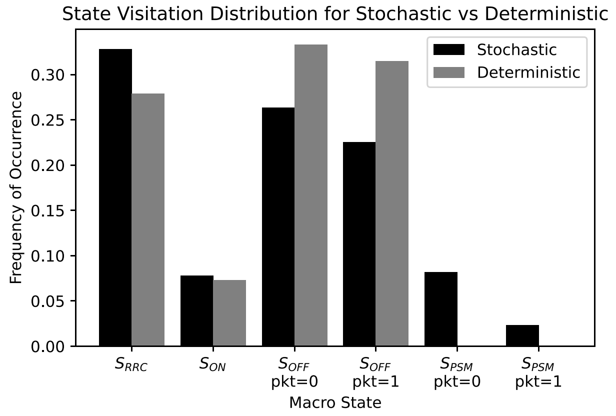

Fig. 15. State visitation distribution for stochastic and deterministic PSM transitions ($n = 10\,000$ and $T_{\text{OFF}} = 300$ ms).

two approaches is illustrated in Fig. 14. It can be seen in this plot that for the majority of DRX *off* timers ($T_{\text{OFF}} < 200$), both approaches match closely. However, with greater DRX off durations, these two approaches begin to diverge, with a deterministic PSM transition incurring a slightly greater cost. This can be explained by taking a deeper look at how these two approaches affect the state distribution in the simulation. This state distribution was plotted for both approaches in Fig. 15. In both of these simulations, T_{OFF} was set to its maximum value of 300 ms. In this plot we can see that in the deterministic approach, almost no time is spent in S_{PSM}. This makes sense because in order to enter S_{PSM} deterministically, it must go through S_{RRC} and $m = 3$ consecutive DRX cycles without an incoming packet. With long DRX cycles, this becomes very unlikely. The stochastic approach, however, spends approximately 11% of the time in S_{PSM}. This is also logical since after going T_{RRC} consecutive time slots without an incoming packet, there is a $1/m = 1/3$ probability of immediately entering S_{PSM}. Since less energy is consumed in S_{PSM} compared to S_{OFF}, there is an overall lower cost for the stochastic simulation as T_{OFF} increases. Despite this difference, it is important to note that these two approaches lead to minimum costs at similar DRX *off* timers. In this

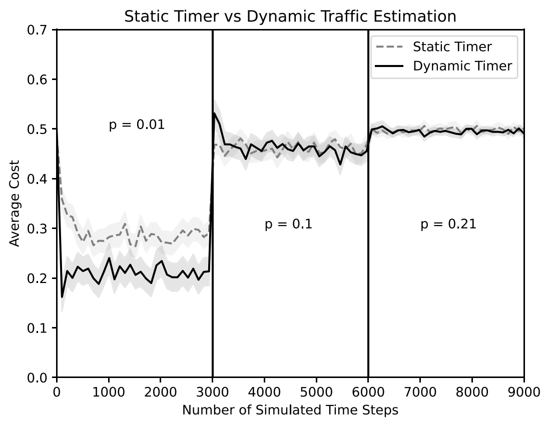

Fig. 16. Average observed cost under a static timer versus a dynamic timer, determined by estimating p.

example, both sets of curves have a minimum at $T_{\text{OFF}} = 70$ ms. It is because of this similarity that this approximation can be used, since both approaches will return the same optimal *off* timer.

G. Traffic Estimation

In addition to the case where the traffic distribution is known a priori, we have also analyzed the case where the traffic intensity is unknown and the probability p to generate a packet needs to be estimated. Prior to its implementation, a set of optimal timers is precomputed using the value iteration for a range of traffic conditions p in the range $[0, 0.4]$ with a granularity of 0.01. This scenario consists of two UEs, one of which is the oracle which knows the value of p while the other one does not. The UE that does not know the value of p assumes that the traffic arrivals are Bernoulli in each time slot and estimates the traffic rate p by averaging from the previous time slots by means of the following equation:

$$p_{\text{est}} = a/t \tag{45}$$

where a is the number of packets that have arrived during the time duration t. This estimation is done each time the device is about to enter DRX *off*, when the DRX *off* timer is set. From this estimated p value, the device looks up the precomputed optimal timer for these traffic conditions and implements it.

Fig. 16 illustrates the performance of two different methods of timer selection. In the blue curve, T_{OFF} is set initially to five time steps (50 ms), which is the optimal timer for $p = 0.1$. This value does not change through the entire simulation. The red curve, however, estimates the incoming traffic rate and selects T_{OFF} accordingly.

For the first 3000 time steps, p is set to 0.01. In this case, the static timer (blue) of 50 ms is far from optimal. The dynamic timer (red) on the other hand is able to learn the incoming traffic rate and adjust T_{OFF} accordingly, leading to a lower cost than the static timer. From time steps 3000 to 6000, p is set to 0.1. This is the optimal traffic intensity for the static timer, so it performs well. For the dynamic timer, a small spike in cost is observed initially before settling on the optimal timer,

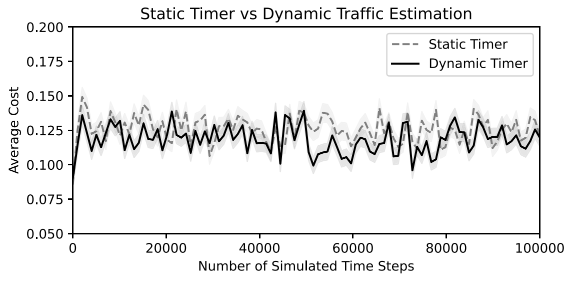

Fig. 17. Average observed cost for traffic estimation method versus optimal static timer—elderly sensor devices scenario.

leading to a cost equal to that of the static timer. This initial spike can be explained by the time it takes for the dynamic timer to learn the incoming traffic rate. In the final 3000 time steps, the performance of both timer selection methods are again equal despite the static timer being suboptimal. This is because with greater values of p, it becomes increasingly unlikely that DRX cycles are entered at all. With almost all time being spent in S_{RRC} and T_{OFF} is nearly irrelevant, and will not affect the observed cost.

An additional experiment was conducted using a more realistic traffic model. For this experiment, we use IEEE's "elderly sensor devices" traffic model [34], where the interarrival times of traffic are distributed according to a Poisson process with a mean of 60 s. The results of this experiment are shown in Fig. 17, where the value of λ used is 0.2. Here, we compare our traffic estimation method (dynamic timer) with a static timer that is aware of the traffic distribution and sets its timer optimally. It is clear that both implementations arrive at the same operating point, which illustrates that our model is transferable to more realistic settings.

H. Computational Complexity Analysis

In this section, we analyze the computational complexity of our improved value iteration algorithm and compare it with the computational complexity of traditional value iteration. All simulation is run on a Windows 11 operating system with an AMD Ryzen 7 3700X 8-core processor at 3.6 GHz with 16 GB RAM.

Both algorithms are computed for a single pair $(p, \lambda) = (0.05, 0.6)$ and the convergence threshold $\delta = 0.001$. For each algorithm, the number of total states is adjusted by increasing or decreasing the timer values. Both value iteration algorithms are conducted on the modified state space and the amount of time required for convergence is recorded. The results for the improved and traditional value iteration algorithms are shown in Fig. 18, where our improved algorithm is also magnified.

The first clear observation that can be made is that with our improved value iteration algorithm, the computational complexity is reduced by approximately two orders of magnitude when compared with the traditional algorithm. Second, it is clear that the improved algorithm exhibits a linear relationship between complexity and the size of the state space while the traditional algorithm's complexity is proportional to the size of the state space squared. The computational complexity of

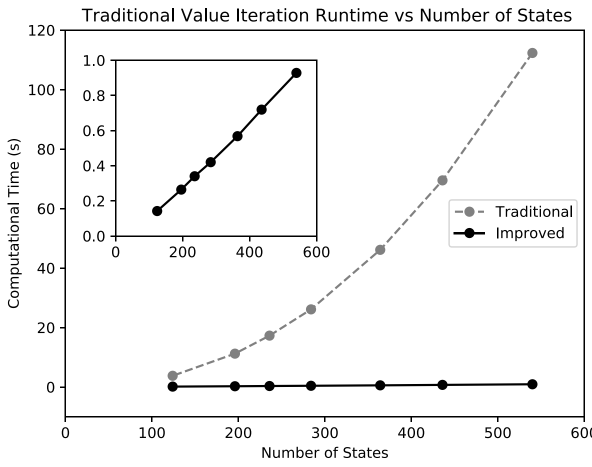

Fig. 18. Algorithm runtime as a function of the number of states ($p = 0.05$, $\lambda = 0.6$, and $\delta = 0.001$).

traditional value iteration is given as $\mathcal{O}(|\mathcal{S}|^2|\mathcal{A}|)$, where $|\mathcal{S}|$ and $|\mathcal{A}|$ are the size of the state and action space, respectively. With the improved algorithm applied to our problem, this is reduced to $\mathcal{O}(|\mathcal{S}|)$, since the number of possible next states from a given state is constant, meaning it does not depend on the total number of states. Overall, the improved algorithm shows a significant improvement in computational time while also demonstrating better scalability in terms of the size of the state space when compared with the traditional algorithm.

I. Comparison to Q-Learning

We compare our solution with an RL approach, an approach also taken in [23]. Note that, in [23], they apply an actor-critic algorithm due to the continuous nature of their problem formulation. Rather than comparing our results directly to actor–critic, we compare them to a Q-learning approach, which is a more appropriate methodology for our discrete model.

A simulation of 10 000 time steps was conducted for the static values $p = 0.05$ and $\lambda = 0.6$. Both Q-learning and our proposed traffic estimation approach were used. The simulation was conducted 100 times for each, and the cumulative average cost for both were recorded and plotted in Fig. 19 along with a 95% confidence interval. It is clear by the end of the simulation that on average, the value iteration solution will provide a lower cost. This is largely due to Q-learning not having nearly enough learning opportunities. The action taken in the DRX model is the selection of the DRX *off* timer, which does not happen frequently. This in turn results in infrequent meaningful learning updates, and thus a greater observed cost. Our traffic estimation approach, on the other hand, quickly estimates the incoming traffic intensity and applies the optimal timer accordingly, allowing the cost to settle at a lower value.

Table V shows the cumulative average observed cost over a simulation of 10 000 time steps for both Q-learning and traffic estimation methods for various values of p and λ. Additionally, the percent improvement of traffic estimation relative to Q-learning is reported. In some cases, traffic estimation is shown to outperform Q-learning by up to 38%. In

TABLE V
COMPARING TRAFFIC ESTIMATION AND Q-LEARNING—AVERAGE COST OBSERVED OVER 10 000 TIME STEPS FOR VARYING p, λ

	$\lambda = 0.2$			$\lambda = 0.6$			$\lambda = 1$		
p	Traffic Estimation	Q-learning	% Improvement	Traffic Estimation	Q-learning	% Improvement	Traffic Estimation	Q-learning	% Improvement
0.01	0.129	0.13	1.07%	0.225	0.294	23.34%	0.36	0.459	21.75%
0.05	0.197	0.269	26.84%	0.448	0.488	8.08%	0.673	0.679	0.9%
0.1	**0.223**	**0.363**	**38.44%**	0.541	0.619	12.6%	0.861	0.867	0.65%

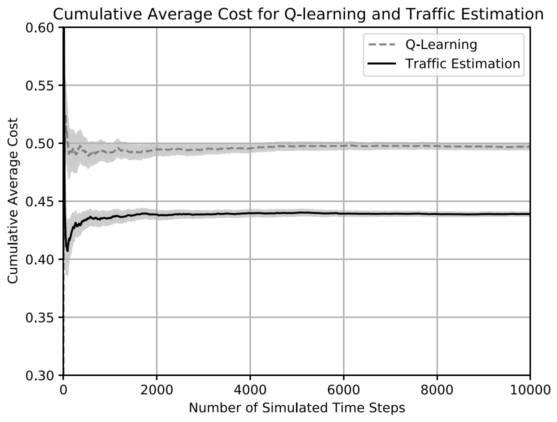

Fig. 19. Cumulative average observed cost for both Q-learning and traffic estimation approaches ($p = 0.05$ and $\lambda = 0.6$).

other cases, the improvement is minimal, but traffic estimation outperforms Q-learning in all cases.

VI. CONCLUSION

In this work, the energy-latency tradeoff inherent to DRX was closely examined. First, the problem of optimizing DRX sleep duration was formulated as a MDP. In this MDP, the action taken is the selection of a DRX *off* duration from a discrete set of possible timers. The state of the device evolved according to a discrete Markov chain realistically simulating DRX operation. A single parameter λ was introduced to facilitate the tradeoff between energy and latency in the cost function of the MDP. This MDP was solved using an improved version of value iteration.

The results of value iteration were analyzed by examining the effects of λ and the incoming traffic intensity p on the optimal timer selection. These results were verified through a simulation during which all possible DRX *off* timers were selected and the average cost was observed. As predicted by the value iteration results, there exists a timer at which the observed cost is at a minimum.

Additionally, a method was proposed to deploy our results. By observing the traffic intensity, the DRX *off* timer can be selected via a precomputed list of optimal timers. Very often in practice, a single DRX *off* timer will be selected, which is suboptimal for traffic rates higher or lower than expected. With our traffic estimation approach, we showed an improvement in observed cost in comparison with this traditional approach in environments where the traffic is either unknown or time-varying. Finally, the performance of this method was compared to an RL approach, which showed our traffic estimation approach outperforms the RL approach.

In the future, we are planning two possible extensions of this work. First, we are planning to include more timers in addition to the DRX *off* timer in the action space. While the DRX *off* timer most directly affects the energy-latency tradeoff, the other DRX related timers, such as the RRC connectivity timer and PSM timer can also affect this tradeoff. Additionally, we plan on expanding the scenario to consider multiple UEs in a single heterogeneous network. In such a scenario, the UEs may place different priority on energy or delay depending on their application. By adding a constraint to this scenario, such as an energy budget for the entire network, the UEs would need to work together to meet this constraint while jointly minimizing their delay.

REFERENCES

[1] “Cellular system support for ultra-low complexity and low throughput Internet of Things (CIoT),” 3GPP, Sophia Antipolis, France, Rep. 45.820, Aug. 2016.

[2] Y. D. Beyene et al., “NB-IoT technology overview and experience from cloud-RAN implementation,” *IEEE Wireless Commun.*, vol. 24, no. 3, pp. 26–32, Jun. 2017.

[3] N. Accurso, N. Mastronarde, and F. Malandra, “Exploring tradeoffs between energy consumption and network performance in cellular-IoT: A survey,” in *Proc. IEEE Glob. Commun. Conf. (GLOBECOM)*, 2021, pp. 1–6.

[4] N. Accurso, N. Mastronarde, and F. Malandra, “Modelling and optimization of DRX in cellular IoT networks: An MDP approach,” in *Proc. IEEE Int. Conf. Commun. (ICC)*, 2023, pp. 6169–6174. [Online]. Available: https://doi.org/10.36227/techrxiv.21992096.v1

[5] P.-Y. Liu, K.-R. Wu, J.-M. Liang, J.-J. Chen, and Y.-C. Tseng, “Energy-efficient uplink scheduling for ultra-reliable communications in NB-IoT networks,” in *Proc. IEEE 29th Annu. Int. Symp. Perso., Indoor Mobile Radio Commun. (PIMRC)*, 2018, pp. 1–5.

[6] N. H. Bui, C. Pham, K. K. Nguyen, and M. Cheriet, “Energy efficient scheduling for networked IoT device software update,” in *Proc. 15th Int. Conf. Netw. Service Manage. (CNSM)*, 2019, pp. 1–5.

[7] A. S. Hampiholi and B. P. V. Kumar, “Efficient routing protocol in IoT using modified genetic algorithm and its comparison with existing protocols,” in *Proc. 3rd Int. Conf. Circuits, Control, Commun. Comput. (I4C)*, 2018, pp. 1–5.

[8] B. R. Stojkoska and Z. Nikolovski, “Data compression for energy efficient IoT solutions,” in *Proc. 25th Telecommun. Forum*, 2017, pp. 1–4.

[9] M. Osta, A. Ibrahim, H. Chible, and M. Valle, “Inexact arithmetic circuits for energy efficient IoT sensors data processing,” in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, 2018, pp. 1–4.

[10] X. Liu, X. Xu, Y. Yuan, X. Zhang, and W. Dou, “Energy-efficient computation offloading with privacy preservation for edge computing-enabled 5G networks,” in *Proc. Int. Conf. Internet Things (iThings) IEEE Green Comput. Commun. (GreenCom) IEEE Cyber, Phys. Soc. Comput. (CPSCom) IEEE Smart Data (SmartData)*, 2019, pp. 176–181.

[11] M. S. A. Shuvo, A. R. Munna, T. Adhikary, and M. A. Razzaque, “An energy-efficient scheduling of heterogeneous network cells in 5G,” in *Proc. Int. Conf. Sustain. Technol. Ind. (STI)*, 2019, pp. 1–6.

[12] J. Kim, H.-W. Lee, and S. Chong, “Traffic-aware energy-saving base station sleeping and clustering in cooperative networks,” *IEEE Trans. Wireless Commun*, vol. 17, no. 2, pp. 1173–1186, Feb. 2018.

[13] R. C. Carrano, D. Passos, L. C. S. Magalhaes, and C. V. N. Albuquerque, “Survey and taxonomy of duty cycling mechanisms in wireless sensor networks,” *IEEE Commun. Surveys Tuts.*, vol. 16, no. 1, pp. 181–194, 1st Quart., 2014.

[14] B.-N. Trinh, L. Murphy, and G.-M. Muntean, "A reinforcement learning-based duty cycle adjustment technique in wireless multimedia sensor networks," *IEEE Access*, vol. 8, pp. 58774–58787, 2020.

[15] W. H. R. Chan et al., "Adaptive duty cycling in sensor networks with energy harvesting using continuous-time Markov chain and fluid models," *IEEE J. Sel. Areas Commun.*, vol. 33, no. 12, pp. 2687–2700, Dec. 2015.

[16] K. Charoenchaiprakit, W. Piyarat, and K. Woradit, "Optimal data transfer of SEH-WSN node via MDP based on duty cycle and battery energy," *IEEE Access*, vol. 9, pp. 82947–82965, 2021.

[17] R. M. Sandoval, A.-J. Garcia-Sanchez, J. Garcia-Haro, and T. M. Chen, "Optimal policy derivation for transmission duty-cycle constrained LPWAN," *IEEE Internet Things J.*, vol. 5, no. 4, pp. 3114–3125, Aug. 2018.

[18] Z. Liu and I. Elhanany, "RL-MAC: A reinforcement learning based MAC protocol for wireless sensor networks," *Int. J. Sens. Netw.*, vol. 1, nos. 3–4, pp. 117–124, 2006.

[19] A. K. Sultania, P. Zand, C. Blondia, and J. Famaey, "Energy modeling and evaluation of NB-IoT with PSM and eDRX," in *Proc. IEEE Globecom Workshops (GC Wkshps)*, 2018, pp. 1–7.

[20] B. Cai, Y. Chen, and I. Darwazeh, "Analyzing energy efficiency for IoT devices with DRX capability and poisson arrivals," in *Proc. 26th Int. Conf. Telecommun. (ICT)*, 2019, pp. 254–259.

[21] F. Moradi, E. Fitzgerald, and B. Landfeldt, "Modeling DRX for D2D communication," *IEEE Internet Things J.*, vol. 8, no. 4, pp. 2574–2584, Feb. 2021.

[22] F. Moradi, E. Fitzgerald, M. Pióro, and B. Landfeldt, "Flexible DRX optimization for LTE and 5G," *IEEE Trans. Veh. Technol.*, vol. 69, no. 1, pp. 607–621, Jan. 2020.

[23] J. Zhou, G. Feng, T. P. Yum, M. Yan, and S. Qin, "Online learning-based discontinuous reception (DRX) for machine-type communications," *IEEE IoT J.*, vol. 6, no. 3, pp. 5550–5561, Jun. 2019.

[24] A. T. Koc, S. C. Jha, R. Vannithamby, and M. Torlak, "Device power saving and latency optimization in LTE-a networks through DRX configuration," *IEEE Trans. Wireless Commun.*, vol. 13, no. 5, pp. 2614–2625, May 2014.

[25] T. A. Le, T. Van Chien, M. R. Nakhai, and T. Le-Ngoc, "Pareto-optimal pilot design for cellular massive MIMO systems," *IEEE Trans. Veh. Technol.*, vol. 69, no. 11, pp. 13206–13215, Nov. 2020.

[26] C. Leone, M. Longo, L. M. Fernández-Ramírez, and P. García-Triviño, "Multi-objective optimization of PV and energy storage systems for ultra-fast charging stations," *IEEE Access*, vol. 10, pp. 14208–14224, 2022.

[27] *5G; NR; Radio Resource Control (RRC); Protocol Specification; (Release 15), Version 15.4.0*, 3GPP Standard TS 38.331, Apr. 2019.

[28] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 2018.

[29] X. Xiong, K. Zheng, L. Lei, and L. Hou, "Resource allocation based on deep reinforcement learning in IoT edge computing," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 6, pp. 1133–1146, Jun. 2020.

[30] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Hoboken, NJ, USA: Wiley, 2014.

[31] N. Apthorpe, D. Huang, D. Reisman, A. Narayanan, and N. Feamster, "Keeping the smart home private with smart(er) IoT traffic shaping," in *Proc. Privacy Enhancing Technol.*, 2019, pp. 128–148.

[32] N. Akar and O. Dogan, "Discrete-time queueing model of age of information with multiple information sources," *IEEE Internet Things J.*, vol. 8, no. 19, pp. 14531–14542, Oct. 2021.

[33] E. Altman, *Constrained Markov Decision Processes: Stochastic Modeling*. Oxfordshire, U.K.: Routledge, 1999.

[34] "IEEE 802.16m evaluation methodology document (EMD)." 2008. [Online]. Available: http://ieee802.org/16/tgm/docs/80216m-08_004r2.pdf