2024 IEEE 63rd Conference on Decision and Control (CDC)
December 16-19, 2024. MiCo, Milan, Italy

Optimizing Queues with Deadlines under Infrequent Monitoring

Faraz Farahvash and Ao Tang

2024 IEEE 63rd Conference on Decision and Control (CDC) | 979-8-3503-1633-9/24/$31.00 ©2024 IEEE | DOI: 10.1109/CDC56724.2024.10885841

Abstract— **In this paper, we aim to improve the percentage of packets meeting their deadline in discrete-time M/M/1 queues with infrequent monitoring. More specifically, we look into policies that only monitor the system (and subsequently take actions) after a packet arrival. We model the system as an MDP and provide the optimal policy for some special cases. Furthermore, we introduce a heuristic algorithm called "AB-n" for general deadlines. Finally, we provide numerical results demonstrating the desirable performance of "AB-n" policies.**

I. Introduction

In recent years, there have been a lot of queuing systems where packets have a deadline to meet [1], [2], [3]. Such queuing systems are called "real-time queuing systems" in the literature [4]. In these systems, packets missing their deadline are discarded. Thus, using smart queuing policies to increase the percentage of packets meeting their deadline is needed.

The classical result proposed by [5] states that the "Earliest Deadline First (EDF)" policy minimizes the percentage of packets missing their deadline in $G/M/c + G$ queues. There have been numerous papers analyzing EDF. An approximation performance analysis is done by [6]. Also, [4] does a heavy traffic analysis for EDF queues. [7] argues that EDF is not necessarily optimal for wireless channels where queue failures can happen. Other related works include but are not limited to [8], [9], [10], [11], and [12].

A key assumption for the optimality of EDF policies is that a packet is dropped from the queue (or server) as soon as its deadline elapses. Enforcing that requires the policy to constantly monitor the system. This is not always feasible. In this paper, we will look into cases where the queuing policy can only monitor the system infrequently. Infrequent monitoring will result in a possibility of packets being served past their deadline. Thus, to prevent that, queue management policies will have to drop packets before their deadline. However, by dropping packets prematurely, we can lose packets that could have made their deadline. Analyzing this tradeoff is the center of this paper.

More specifically, we will analyze a discrete-time M/M/1 queue where the policy can only drop packets after a packet arrival event happens. We choose packet arrival as our monitoring trigger for three main reasons: (a) First of all, arrivals at the queue naturally incur a trigger in the system (no external clocks are needed). (b) It causes a relatively low frequency of monitoring. (c) It is not policy or model specific and can be extended more easily to general G/G/1 queues (in contrast to non-frequent versions of EDF where the monitoring occurs after a packet misses its deadline). We assume all the packets have the same hard deadline (D). We use two different approaches to this problem.

As a first approach, we write the optimization problem as a Markov Decision Process (MDP). We find the optimal policies for small deadlines ($D = 2, 3$). We also mention some observations and properties of the optimal policies. We observe that, in contrast to frequent modeling case, the decision of dropping a packet depends on the arrival rate, service rate, and the ages of packets present at the queue. This increases the complexity of the optimal policy considerably (as EDF only needs to keep track of the age of the packet at the head).

Noting that finding the optimal policy for general deadline using the MDP approach is inefficient, we will provide a heuristic policy called "$AB - n$". This policy reduces the computation complexity by only considering the first n packets in the queue and maximizing their chances of meeting their respective deadlines. We will show that this policy outperforms the previously proposed algorithms.

The rest of the paper is organized as follows. In section II, we formulate the problem and discuss the notion of extended states and infrequent monitoring. In section III, some previous results (namely EDF and DPGP) are mentioned. In section IV, we formulate the MDP for this problem. Section V presents the results for the optimal policy stemming from the MDP. The heuristic policy $AB - n$ is introduced in section VI. Section VII presents some experiments. Section VIII concludes the paper.

II. Problem Formulation

In this section, we formally define the problem. We will use the conventional discrete-time model for the M/M/1 queue described in [13]. The time is divided into time slots of unit duration, and we have the following rules:

- At each time slot, at most one packet arrival or service happens. Arrival and service can not occur at the same time slot.

This work was supported by National Science Foundation (NSF). Grant number: 2133403

Faraz Farahvash and Ao Tang are with the School of Electrical and Computer Engineering, Cornell University, Ithaca, NY 14853, USA. Emails: {ff227,atang}@cornell.edu

- Events in different time slots are independent.
- At each time slot, an arrival happens with probability λ. If the queue is not empty, a packet service happens with probability μ.

Furthermore, all packets have the same hard deadline (D). Packets that do not meet the deadline are rendered pointless. We are trying to maximize the percentage of packets that meet their deadline.

At each time, the queue is recorded as $T = (T_1, ..., T_N)$ where T_i is the age of the i^{th} packet (i.e., the time elapsed since its arrival). We assume T_1 is the head of the queue and T_N is the tail. This is the same definition presented at [14] as extensive states.

Finally, unlike previous results (i.e., EDF), the queue is only monitored after a packet arrival (infrequent monitoring). In other words, the policy is only allowed to drop a packet (or packets) after a packet arrives at the queue. Packets are only dropped from the head of the queue.

III. Previous Results

In this section, we will present some previous results that will serve as points of comparison. More specifically, we will look into two specific policies (DPGP and EDF) and explain why these policies are not optimal in our model.

A. Earliest Deadline First (EDF)

In the real-time queuing literature, [5] provides the following definition for the Earliest Deadline First (or Shortest Time to Extinction (STE)) Policies and proves that EDF is the optimal policy for the discrete-time G/M/c+G queue.

Definition. *A policy is an **Earliest Deadline First** policy if (1) at any time, it always schedules the available packet closest to its deadline, (2) the servers are always busy as long as there are available packets (there are no forced idle times), and (3) the packets are discarded (removed from the queue and server) as soon as their deadline elapses.*

Remark. *Note that the third condition for EDF policies can not be enforced with infrequent monitoring (the policy is only allowed to drop at certain time slots).*

Here, we provide an intuitive example to illustrate the reason why EDF can be sub-optimal with infrequent monitoring.

Example. Let's assume we are looking into the queue at time slot t. Also, let the packet at the head of the queue (p_1) and the second packet (p_2) have k and $k + k'$ timeslots till their deadline, respectively. We are interested in the expected number of the packets making their deadline out of these two packets (we ignore the rest of the queue). We want to see whether dropping the packet at the head of the queue at time t can increase this probability.

First, we look into the constant monitoring case. By dropping the packet at the head of the queue at time t, the second packet will have $k + k'$ timeslots to be served. Thus, the expected number of packets served before their deadline is equal to $A = 1 - (1-\mu)^{k+k'}$. Now, by not dropping the packet, p_1 will have k timeslots to be served. Furthermore, p_2 will have at least k' timeslots to be served. (because of the constant monitoring p_1 is dropped after k timeslots if not served.) Hence, the expected number of packets being served is at least $B = 2 - (1-\mu)^k - (1-\mu)^{k'}$. It is easy to see that $\forall k, k' \in \mathbb{N}, \mu \in (0,1)$, $B \geq A$. Thus, dropping a packet before its deadline never increases the expectation (regardless of the values of t, k, k', μ).

Now, we go back into the infrequent monitoring case. Here, by dropping the packet at the head of the queue at time t, the desired expected value will be the same as the constant monitoring case (i.e., A). Finally, if we decide to not drop the packet at time t, p_1 will still have k timeslots to be served. But contrary to the frequent monitoring case, p_2 can have less than k' timeslots to be served (depending on the next arrival and departure times). Thus, there exist cases where dropping will increase the expectation. We will inspect these cases more thoroughly in future sections.

B. Drop Positive Gain Policy (DPGP)

[14] argues that when deciding to drop a packet, two factors should be considered:

1) The probability of the packet making the deadline
2) The impact of the packet being dropped on the probability of other packets meeting their respective deadlines.

Remark. *The probability of the i^{th} packet making the deadline is equal to $I_\mu(i, D - T_i - i + 1)$, Where I is the regularized incomplete beta function [15] (Section 6.6).*

To formally compute the trade-off of the two factors, the paper introduces the concept of gain as follows (The original definition is for the continuous time queue. We change the definition to fit the discrete model).

Definition. *The gain of dropping the i^{th} packet in state s is defined as:*

$$gain_i^s(\mu) = -I_\mu(i, D - T_i - i + 1) + \sum_{j=i+1}^{N} \left(I_\mu(j-1, D - T_j - j + 2) - I_\mu(j, D - T_j - j + 1) \right) \quad (1)$$

Using the gain function defined above and by proving that the gain function is maximized at the head of the queue, [14] introduces the Drop Positive Gain Policy as follows:

"Drop the packet at the head of the queue if and only if the $gain_1^s(\mu) > 0$."

Remark. *gain is a myopic concept and ignores future system evolution (packet arrival, service, or drop). Thus, DPGP is not the optimal policy.*

Now that we have established that the previously proposed policies are not optimal, we will provide two different approaches to finding better policies. First, in section IV, we will try to find the optimal policy by formulating the queue as an MDP. Next, in section VI, using the intuition gained from the example above, we propose a heuristic policy.

IV. Queue as an MDP

Markov Decision Processes (MDPs) are used for making a sequence of decisions in situations where the outcomes are uncertain. This framework can fit the queue optimization problem that we are analyzing in this paper. The MDP we use for this problem is an infinite-horizon MDP where we are trying to maximize the average reward per stage.

A. General definition

An MDP is defined as $\mathcal{M} = (S, \mathcal{A}, P, r)$ where, S is the state space. $\mathcal{A}$ is the action space. It maps a state to admissible actions for that state. P is the transition function. $P : S \times \mathcal{A} \to \Delta(S)$ where $\Delta(S)$ is the space of probability distributions over S. In simple words, P explains the transition probability from a state and an action taken when in that state, to a new state. Finally, r is the reward function. $r : S \times \mathcal{A} \to \mathbf{R}$. $r(s, a)$ is the stage reward associated with taking action a in state s. We will try to optimize the average reward per stage starting from state s (using policy π), which is defined as:

$$J^{\pi}(s) = \lim_{N\to\infty} \frac{1}{N} \mathbf{E}\left[\sum_{t=0}^{N-1} r(s_t, a_t) | \pi, s_0 = s\right]$$

In the next part, we will formulate the problem of maximizing the expected percentage of packets meeting the deadline as an MDP.

B. MDP Design

To derive the MDP for this problem, we will define each of $(S, \mathcal{A}, P, r)$. We also note that state transitions will happen after a packet arrives, is served, or is dropped.

1) S: The states are the same as T (extensive states) with two additional binary bits (b_a, b_r) which are defined as below:

$$b_a = \begin{cases} 1, & \text{If we are allowed to drop packets} \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

And:

$$b_r = \begin{cases} 1, & \text{If } b_a = 0 \text{ and the previously served} \\ & \text{packet made the deadline} \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

In other words, b_a accounts for whether or not a packet service was the reason for the previous state transition (this will help us define the action space). b_r shows that if the previous state transition was caused by a packet service, whether the packet made the deadline or not (this will help us define the reward function). Note that without defining b_r, we need to record the previous state to realize whether the previous packet made its respective deadline. Thus, although the addition of these two binary bits are not necessary, they greatly simplify the definition of the action space and the reward function.

To conclude, state s can be defined as:

$$s = ((T_1^s, ..., T_n^s), b_a^s, b_r^s)$$

2) $\mathcal{A}$: To define the action space, we will use b_a. We have two possibilities:

- $b_a^s = 0$: If $b_a^s = 0$, a packet service was the reason for the previous state transition. Thus, we are not allowed to drop a packet and $\mathcal{A}(s) = \{\bar{d}\}$ Where $\bar{d}$ means not dropping a packet.
- $b_a^s = 1$: If $b_a^s = 1$, a packet service was not the reason for the previous state transition. Thus, we are allowed to drop a packet and: $\mathcal{A}(s) = \{\bar{d}, d\}$ Where $\bar{d}$ means not dropping a packet, and d means dropping the packet at the head.

3) P: We will first provide the following lemma.

Lemma 1. *If the queue is not empty, the interval between two consecutive state transitions comes from a geometric distribution with parameter $\lambda + \mu$ and the probability of an arrival triggering the state transition is $\frac{\lambda}{\lambda+\mu}$.*

Proof: If the queue is not empty, the probability of no arrival and no service (failure) in a slot is $1 - \mu - \lambda$. Thus, the probability distribution of the first success (arrival or packet service) is $geom(\lambda + \mu)$. The probability of the success being from arrival is $\frac{\lambda}{\lambda+\mu}$.

Remark. *If the queue is empty, the interval between two consecutive state transitions comes from a geometric distribution with parameter λ.*

To describe the transition function, we condition it on the action taken:

- $a = d$: The next state is deterministic and it will be $s' = ((T_2^s, \dots, T_n^s), 1, 0)$ with probability 1.
- $a = \bar{d}$: Here the next state is stochastic. We have:
  - n=0: Then the next state will be $s' = ((0), 1, 0)$ with probability 1.
  - $n > 0$: Then using lemma 1, we get that with probability $\frac{\lambda}{\lambda+\mu}$:

    $$s' = ((T_1^s + K, ..., T_n^s + K, 0), 1, 0)$$

    where $K \sim geom(\lambda + \mu)$.
    Otherwise, with probability $\frac{\mu}{\lambda+\mu}$:

    $$s' = ((T_2^s + K, ..., T_n^s + K), 0, b_r^{s'})$$

    where $K \sim geom(\lambda + \mu)$ and:

    $$b_r^{s'} = \begin{cases} 1, & \text{If } K \le D - T_1^s \\ 0, & \text{Otherwise} \end{cases}$$

4) r: The reward function is pretty simple and has the form of $r(s,a) = 2b_r^s$.

In the next part, we will provide reasoning on the equivalence of the MDP defined above and the optimization problem itself.

C. On the equivalence of the MDP and original problem

Note that we are optimizing:

$$J^\pi(s) = \lim_{N\to\infty} \frac{1}{N}\mathbf{E}\left[\sum_{t=0}^{N-1} r(s_t,a_t)|\pi, s_0 = s\right]$$

Now, first, note that each packet causes two state transitions (once with arrival and once with departure). Thus, asymptotically speaking, if we let the number of packets till state transition number N be Y_N, we have:

$$\lim_{N\to\infty} \frac{Y_N}{N} = \frac{1}{2}$$

and thus, we can rewrite $J^\pi(s)$:

$$J^\pi(s) = \lim_{N\to\infty} \frac{1}{2Y_N}\mathbf{E}\left[\sum_{t=0}^{N-1} 2b_r^{s_t}|\pi, s_0 = s\right]$$

$\sum_{t=0}^{N-1} b_r^{s_t}$ is equal to the number of packets meeting the deadline till N. Thus:

$$\lim_{N\to\infty} \frac{1}{Y_N}\mathbf{E}\left[\sum_{t=0}^{N-1} b_r^{s_t}|\pi, s_0 = s\right] = \mathbf{P}(\text{meeting the deadline})$$

Thus, an optimal policy for the MDP problem maximizes the expected percentage of packets meeting the deadline. Furthermore, the reward incurred by a policy on the MDP is the same as the probability of a packet meeting the deadline if policy π is implemented for the original problem. Thus, to the extent of our interest, these two problems are equivalent.

V. MDP OPTIMIZATION

In this section, we present results on the optimal policies for the MDPs presented in previous sections.

As the number of states is infinite (or with some considerations mentioned in the next part, exponential), it is not efficient to solve this MDP using methods such as policy iteration or value iteration. But, we will provide the optimal policy for some special cases (i.e., $D = 2, 3$)

A. The case with D=2

Here, we would define the optimal policy for the special case where the deadline is equal to 2. Any optimal policy would drop the packets that have missed the deadline (If they are allowed to drop packets). Furthermore, the queue under any optimal policy will never have a length of more than 3. Finally, if $T_i \geq D$, we will say $T_i = D$ since we only care that the packet has missed the deadline. Also, for simplicity define $\alpha = \lambda + \mu$.

Using the above considerations, the queue for D=2 will have 9 possible states. We will describe the state transitions here:

1) $((),0,0)$: The only possible action is not dropping, and the next state will be $((0),1,0)$ with probability 1.
2) $((),0,1)$: The only possible action is not dropping, and the next state will be $((0),1,0)$ with probability 1.
3) $((0),1,0)$: The only possible action is not dropping (as if we drop the packet here, we will circulate between state 1 and this state forever, and the reward would be zero.). We have:

$$((0),1,0) \xrightarrow{\bar{d}} \begin{cases} ((),0,0), & \text{wp } \mu\frac{(1-\alpha)^2}{\alpha} \\ ((),0,1), & \text{wp } \mu(2-\alpha) \\ ((1,0),1,0), & \text{wp } \lambda \\ ((2,0),1,0), & \text{wp } \lambda\frac{(1-\alpha)}{\alpha} \end{cases}$$

4) $((1),0,1)$: The only possible action is not dropping, and the next state will be:

$$((1),0,1) \xrightarrow{\bar{d}} \begin{cases} ((),0,0), & \text{wp } \mu\frac{(1-\alpha)}{\alpha} \\ ((),0,1), & \text{wp } \mu \\ ((2,0),1,0), & \text{wp } \frac{\lambda}{\alpha} \end{cases}$$

5) $((2),0,0)$: The only possible action is not dropping, and the next state will be:

$$((2),0,0) \xrightarrow{\bar{d}} \begin{cases} ((),0,0), & \text{wp } \frac{\mu}{\alpha} \\ ((2,0),1,0), & \text{wp } \frac{\lambda}{\alpha} \end{cases}$$

6) $((1,0),1,0)$: This is the most important state, and we have two possible actions:
   - Drop packet 1: The next state will be $((0),1,0)$ with probability 1.
   - Don't drop: The next will be:

$$((1,0),1,0) \xrightarrow{\bar{d}} \begin{cases} ((1),0,1), & \text{wp } \mu \\ ((2),0,0), & \text{wp } \mu\frac{1-\alpha}{\alpha} \\ ((2,1,0),1,0), & \text{wp } \lambda \\ ((2,2,0),1,0), & \text{wp } \frac{\lambda(1-\alpha)}{\alpha} \end{cases}$$

7) $((2,0),1,0)$: Any optimal policy would drop packet 1 as it has missed its deadline. Thus, the next state will be $((0),1,0)$.
8) $((2,1,0),1,0)$: Any optimal policy would drop packet 1 as it has missed its deadline. Thus, the next state will be $((1,0),1,0)$.
9) $((2,2,0),1,0)$: Any optimal policy would drop packet 1 as it has missed its deadline. Thus, the next state will be $((2,0),1,0)$.

Thus, any policy has to decide which action to take in state 6. Let's see what decision DPGP makes. $gain_1^s(\mu)$ is equal to:

$$gain_1^s(\mu) = (\mu + \mu(1-\mu)) - \mu^2 - \mu = \mu - 2\mu^2$$

Thus, DPGP will be:

$$a(s_6) = \begin{cases} d, & \text{if } \mu < 0.5 \\ \bar{d}, & \text{if } \mu \geq 0.5 \end{cases} \tag{4}$$

To compute the optimal policy, let π^d be the stationary distribution of the Markov chain induced by policy d on our MDP. Now, by definition of the reward ($r = 2b_r$), the percentage of packets meeting the deadline will be $2(\pi_2^d + \pi_4^d)$. Thus, given λ, μ, the optimal policy maximizes $2(\pi_2^d + \pi_4^d)$ (Call that $AR(\lambda, \mu)$).

Depending on μ and λ, the optimal policy has one of these two forms. Either $a(s_6) = d$ and we drop at state 6 or $a(s_6) = \bar{d}$ and we keep the packet.

We will compute the rewards of each policy and compute the optimal policy.

1) $a(s_6) = d$: If we decide to drop from the head in state 6, the Markov chain will have the structure shown in Fig. 1.
The transition matrix would have the format (deleting the states that we will never enter):

$$P = \begin{bmatrix} 1: & 0 & 0 & 1 & 0 & 0 \\ 2: & 0 & 0 & 1 & 0 & 0 \\ 3: & \mu\frac{(1-\alpha)^2}{\alpha} & \mu(2-\alpha) & 0 & \lambda & \lambda\frac{1-\alpha}{\alpha} \\ 6: & 0 & 0 & 1 & 0 & 0 \\ 7: & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

To compute the stationary distribution, we must have $\pi P = \pi$. We have:

$$\pi_1 + \pi_2 + \pi_3 + \pi_6 + \pi_7 = 1 \tag{5}$$
$$\pi_3 = \pi_1 + \pi_2 + \pi_6 + \pi_7 \tag{6}$$
$$\pi_2 = \mu(2-\alpha)\pi_3 \tag{7}$$

Combining equation 5 and 6, we get that $\pi_3 = 0.5$ and plugging it in equation 7, we get:

$$AR_1(\lambda, \mu) = 2(\pi_2^d + \pi_4^d) = \mu(2-\alpha) \tag{8}$$

2) $a(s_6) = \bar{d}$: If we decide not to drop from the head in state 6, the Markov chain will have the following transition matrix [1]. Without going into further detail, we present the stationary distribution for important states:

$$\pi_2 = \frac{\mu}{2}[(2-\alpha)(1-\lambda) + \mu], \qquad \pi_4 = \frac{\mu\lambda}{2}$$

We have:

$$AR_2(\lambda, \mu) = \mu(2-\alpha) + \lambda\mu[2\mu + \lambda - 1] \tag{9}$$

Thus, the optimal policy would decide to drop the packet in state 6 if and only if $AR_1(\lambda, \mu) > AR_2(\lambda, \mu)$. This will happen when:

$$AR_1(\lambda, \mu) > AR_2(\lambda, \mu) \leftrightarrow 2\mu + \lambda < 1$$

Therefore, the optimal policy is described below:

$$a^*(s_6) = \begin{cases} d, & \text{if } 2\mu + \lambda < 1 \\ \bar{d}, & \text{Otherwise} \end{cases} \tag{10}$$

Fig. 2 shows the boundary of the optimal policy. For any pair (λ, μ) below the red line, the optimal policy would drop from the head when at state (1,0). Similarly, for any pair (λ, μ) above the red line, the optimal policy wouldn't drop any packets at state (1,0).

Remark. *Note that DPGP is the same as the optimal policy for $\lambda = 0$. This is true as DPGP ignores any effect that a new arrival has on the system. Thus, for small arrival rates, the myopic gain computed by DPGP is close to the actual gain.*

[1]

$$P = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ \mu\frac{(1-\alpha)^2}{\alpha} & \mu(2-\alpha) & 0 & 0 & 0 & \lambda & \lambda\frac{1-\alpha}{\alpha} & 0 & 0 \\ \mu\frac{(1-\alpha)}{\alpha} & \mu & 0 & 0 & 0 & 0 & \frac{\lambda}{\alpha} & 0 & 0 \\ \frac{\mu}{\alpha} & 0 & 0 & 0 & 0 & 0 & \frac{\lambda}{\alpha} & 0 & 0 \\ 0 & 0 & 0 & \mu & \mu\frac{(1-\alpha)}{\alpha} & 0 & 0 & \lambda & \lambda\frac{1-\alpha}{\alpha} \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

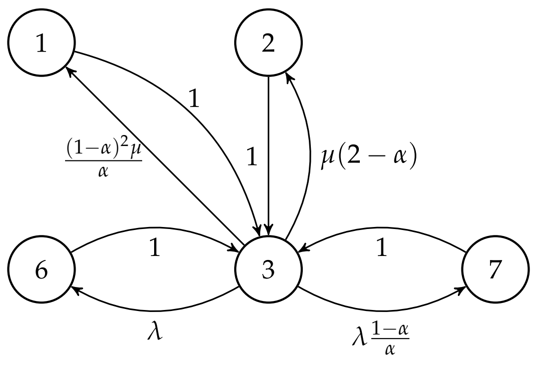
1
2
6
3
7
1
$\frac{(1-\alpha)^2\mu}{\alpha}$
1
$\mu(2-\alpha)$
1
1
λ
$\lambda\frac{1-\alpha}{\alpha}$

Fig. 1. The Markov chain of $a = d$

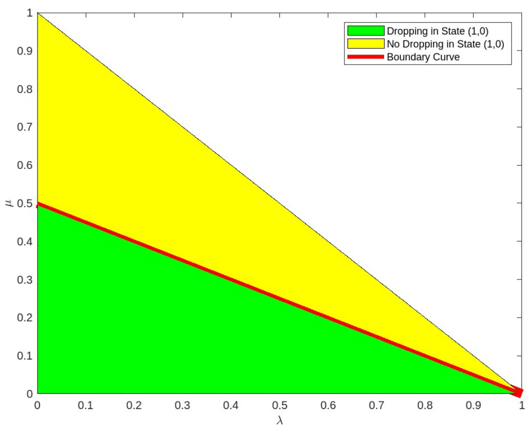

Fig. 2. Optimal policy for D=2

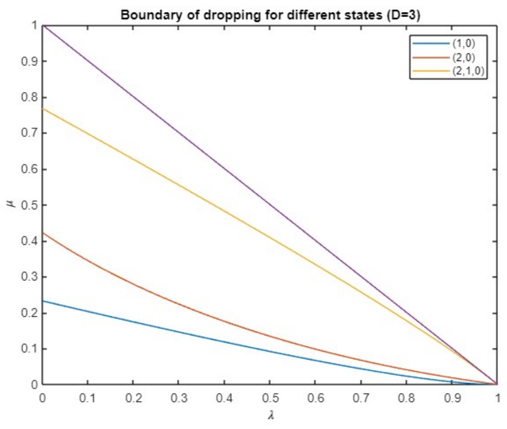

Fig. 3. optimal policy boundary for D=3

B. The case with D=3

Here, we will define the optimal policy for the special case where the deadline is equal to 3. We have the same considerations as D=2. Without going into further detail (for a complete analysis, visit Appendix A in [16]), we mention that the system will have 20 states, and the optimal policy should decide in 3 different states (that are non-trivial) whether to drop from the head or not drop at all. These states are ((1,0),1,0), ((2,0),1,0), and ((2,1,0),1,0) respectively. We see that, depending on (λ, μ), the optimal policy has one of the following forms:

(a) Drops in all of the above states.
(b) Drops in states ((2,1,0),1,0) and ((2,0),1,0).
(c) Only drops in state ((2,1,0),1,0).
(d) Drops in none of the above states.

Fig. 3 shows the boundaries of the optimal policy. For any pair (λ, μ) below the blue line, the optimal policy would be (a). For (λ, μ) between red and blue lines, policy (b) would be optimal. If the point is between red and yellow lines, we would only drop in the state $((2,1,0),1,0)$ (i.e., policy (c)). Finally, if we are above the yellow line, (d) is the optimal policy.

Alternatively, for any point below the yellow line, we would drop at state ((2,1,0),1,0). If (λ, μ) is below the red line, an optimal policy would drop at state ((2,0),1,0), and for all parameters below the blue line, we would drop at state ((1,0),1,0).

C. On properties of the boundaries

In this section, we will present some key properties of the boundaries of the optimal policies for general deadline D. (The boundaries for special cases can be seen in Fig. 2, 3, and 4)

From now on, for any state s, we will refer to the boundary point at $\lambda = 0$ as $M(s)$. Also, we let μ_{bound} to be the service rate where $gain_1^s(\mu_{bound}^s) = 0$.

Theorem 1. *For any state s, $M(s) = \mu_{bound}^s$.*

Proof: If the arrival rate is equal to zero, we have two important observations:

1) The reward only depends on the packets already in the system (there will not be packet arrivals).
2) As there are no arrivals in the future, we can not drop any packets in the future, and thus the probability of the i^{th} packet making the deadline is exactly $I_\mu(i, D-T_i-i+1)$ if we decide to keep the packet at the head and $I_\mu(i-1, D-T_i-i+2)$ if we decide to drop. (III-B)

Thus, the M(s) would be the service rate where the gain is indifferent to the dropping policy at s. In other words (let $H_j = D - T_j - j$):

$$\sum_{j=1}^{N} I_{M(s)}(j, H_j+1) = \sum_{j=2}^{N} I_{M(s)}(j-1, H_j+2)$$

By a little rearrangement, we get:

$$\sum_{j=2}^{N} \Big(I_{M(s)}(j-1, H_j+2) - I_{M(s)}(j, H_j+1) \Big) \\ - I_{M(s)}(1, D-T_1) = 0$$

Which means $gain_1^s(M(s)) = 0$ and thus $M(s) = \mu_{bound}^s$.

Finally, we will present a conjecture we suspect to be true regarding the boundary lines.

Conjecture 1. *For a given deadline D, non-trivial states have an ordering. More specifically, let $B^s(\lambda)$ be the boundary curve for the optimal policy at non-trivial state s. These curves don't cross. More precisely:*

$$\nexists s_1, s_2 \in S, \lambda < 1 : B^{s_1}(\lambda) = B^{s_2}(\lambda), B^{s_1}(\lambda) \neq 1-\lambda \tag{11}$$

VI. Heuristic Policy $(AB-n)$

As established in the previous section, finding the optimal policy using the MDP method is not feasible for larger deadlines. Thus, in this section, we will introduce a heuristic algorithm to approximate the optimal policy.

To do that, we will look back into the example provided in section III-A. Recall that we are looking into the first two packets at the head of the queue (with k and $k+k'$ timeslots until expiration), and we are trying to maximize the expected number of packets meeting their deadline out of the two packets.

If we decide to drop the packet at the head of the queue, the expectation is equal to $A_2(k,k') = 1-(1-\mu)^{k+k'}$. Now, we compute the expectation without dropping the packet. Call this number $B_2(k,k')$. We calculate this value by conditioning on the next event (arrival or departure). There are three types of possibilities for the next event.

1) An arrival event happens before the deadline of the first packet: In this case, the expected value is equal to $max(A_2(k-i,k'), B_2(k-i,k'))$ where i is the time of the packet arrival.

2) A departure event happens before the deadline of the first packet: In this case, the expected value is equal to $2-(1-\mu)^{k+k'-i}$ where i is the time of the packet departure.
3) An arrival or departure event happens after the deadline of the first packet: In this case, the expected value is equal to $1-(1-\mu)^{k+k'-i}$ where i is the time of the event happening.

Putting all the results above together, we get the definition of $B_2(k,k')$ in equation 12.

Remark. *Note that $B_2(k,k')$ only depends on $B_2(l,k')$ with $l \leq k$. Thus, we can calculate $B_2(k,k')$ without the need to solve linear equations.*

Now, that we have calculated $B_2(k,k')$, we can introduce the "$AB-2$" policy:

$AB-2$ Policy: While deciding whether to drop in state T, the $AB-2$ policy will drop the packet if and only if either $T_1 \geq D$ or $|T|>1$ and $A_2(D-T_1, T_2-T_1) > B_2(D-T_1, T_2-T_1)$.

Remark. *If the deadline is equal to 2 (D=2), the $AB-2$ policy would drop in state $T=(1,0)$ (the only nontrivial state) iff $\lambda+2\mu<1$ which is the same as the optimal policy.*

Note that by extending the number of packets considered in the expected value, we can improve the $AB-2$ policy (We call it the $AB-n$ policy where n is the number of packets considered). A_3 and B_3 for $AB-3$ policy can be seen in equations 13 and 14 respectively.

VII. Numerical results and experiments

A. The D=4 case

Here, we will experimentally find the optimal policy for the case where the deadline is equal to 4. In this case, we have 7 non-trivial states where our policy has to decide. To find the optimal policy, we implement the M/M/1 queue with different λ and μ using each policy and find the policy that maximizes the percentage of packets meeting their deadline. Fig. 4 shows the boundary of the optimal policies. For any pair (λ,μ) below each line, the optimal policy would drop from the head when at that state. For instance, if (λ,μ) is below the green line, we would drop at state $((3,1,0),1,0)$.

B. $AB-n$ with different n values

Fig. 5, compares the performance of $AB-n$ policies with different values of n for $\lambda=0.3, \mu=0.2$. As can be seen, the percentage of packages meeting their deadlines improves by increasing the value of n, but the rate of improvement decreases as n gets larger.

C. $AB-5$ vs DPGP and EDF

Fig. 6 compares the performance of $AB-5$ policy with DPGP and EDF introduced in section III. Both EDF with frequent and infrequent monitoring are used. $AB-5$ outperforms both DPGP and infrequent EDF for shorter deadlines. For larger deadlines, the performance of DPGP and $AB-5$ becomes almost identical. Generally speaking, we observe that, by increasing the deadline, DPGP will eventually outperform $AB-n$ (albeit slightly). We note that in comparison to DPGP, $AB-n$ policy disregards certain packets (if the queue length is more than n) to gain farsightedness and consider all future interactions of the n packets. Thus, we believe that the above phenomenon happens as a result of the benefit of farsightedness being outweighed by the drawbacks caused by disregarding packets. Hence, it is not surprising that by increasing n this phenomenon happens later.

VIII. Conclusion

In this paper, we looked into discrete time M/M/1 queues where packets have a hard deadline. We assumed that continuous monitoring of the system is not feasible. Thus, we introduced infrequent monitoring, where the system is only monitored after a packet arrival event happens. We tried to maximize the percentage of packets meeting their deadline.

We had two approaches to this problem. First, the queue was modeled as an MDP. We presented the optimal policy for small deadlines (D=2, 3). Some properties of the optimal policies were discussed.

As a second approach, we introduced a heuristic policy ($AB-n$) which improves the performance of the queue compared with previous algorithms (DPGP and EDF). Finally, some numerical simulations were provided to verify the results.

As a line of future works, this approach can be applied to latency based utility optimization (cases without a hard deadline). Another natural extension could be generalizing the results and methods to Geom/G/1 queues. This can be done by recording the number of timeslots the packet at the head has been served. We have designed the MDP for this queue but omitted it in interest of conciseness.

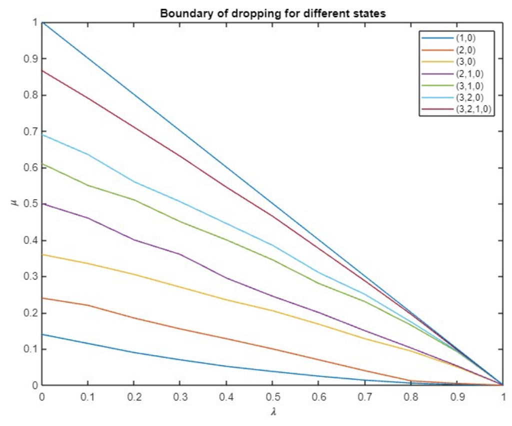

Fig. 4. Optimal policy boundary for D=4

$$B_2(k,k') = \sum_{i=1}^{k} \lambda(1-\lambda-\mu)^{i-1} \max(A_2(k-i,k'), B_2(k-i,k')) + \sum_{i=1}^{k} \mu(1-\lambda-\mu)^{i-1}(2-(1-\mu)^{k+k'-i}) + \sum_{i=k+1}^{k+k'} (\mu+\lambda)(1-\lambda-\mu)^{i-1}(1-(1-\mu)^{k+k'-i}) \tag{12}$$

$$A_3(k_1,k_2,k_3) = \max(A_2(k_1+k_2,k_3), B_2(k_1+k_2,k_3)) \tag{13}$$

$$\begin{aligned} B_3(k_1,k_2,k_3) = &\sum_{i=1}^{k_1} \lambda(1-\lambda-\mu)^{i-1} \max\left(A_3(k_1-i,k_2,k_3), B_3(k_1-i,k_2,k_3)\right) + \sum_{i=1}^{k_1} \mu(1-\lambda-\mu)^{i-1}(1+B_2(k_1+k_2-i,k_3)) \\ &+ \sum_{i=k_1+1}^{k_1+k_2} \lambda(1-\lambda-\mu)^{i-1} \max\left(B_2(k_1+k_2-i,k_3), A_2(k_1+k_2-i,k_3)\right) + \sum_{i=k_1+1}^{k_1+k_2} \mu(1-\lambda-\mu)^{i-1} B_2(k_1+k_2-i,k_3) \\ &+ \sum_{i=k_1+k_2+1}^{k_1+k_2+k_3} \lambda(1-\lambda-\mu)^{i-1}(1-(1-\mu)^{k_1+k_2+k_3-i}) + \sum_{i=k_1+k_2+1}^{k_1+k_2+k_3} \sum_{j=i+1}^{k_1+k_2+k_3} (\mu^2+\mu\lambda)(1-\lambda-\mu)^{j-2}(1-(1-\mu)^{k_1+k_2+k_3-j}) \end{aligned} \tag{14}$$

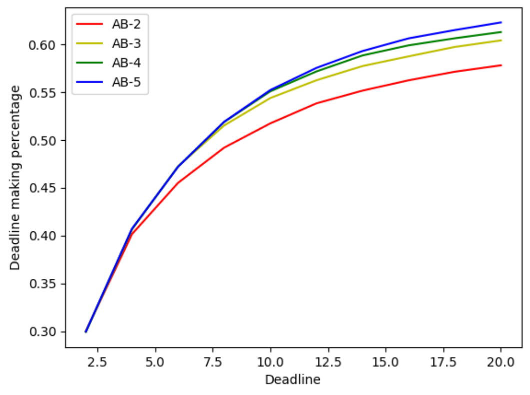

Fig. 5. $AB-n$ Performance

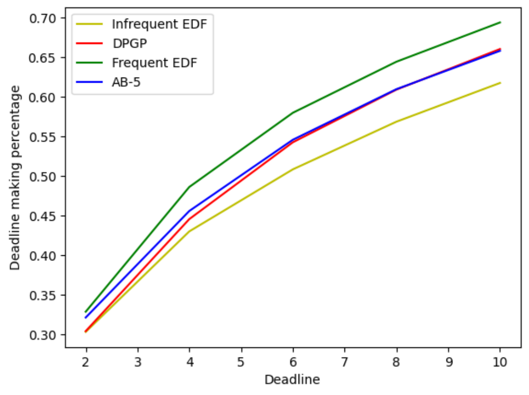

Fig. 6. $AB-n$ performance compared with previous algorithms

References

[1] C. Wilson, H. Ballani, T. Karagiannis, and A. Rowtron, "Better never than late: Meeting deadlines in datacenter networks," in *Proceedings of the ACM SIGCOMM 2011 Conference*, SIGCOMM '11, (New York, NY, USA), p. 50–61, Association for Computing Machinery, 2011.

[2] M. Alizadeh, A. Greenberg, D. A. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, and M. Sridharan, "Data center tcp (dctcp)," in *Proceedings of the ACM SIGCOMM 2010 Conference*, SIGCOMM '10, (New York, NY, USA), p. 63–74, Association for Computing Machinery, 2010.

[3] B. Vamanan, J. Hasan, and T. Vijaykumar, "Deadline-aware datacenter tcp (d2tcp)," in *Proceedings of the ACM SIGCOMM 2012 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, SIGCOMM '12, (New York, NY, USA), p. 115–126, Association for Computing Machinery, 2012.

[4] Łukasz Kruk, J. Lehoczky, K. Ramanan, and S. Shreve, "Heavy traffic analysis for EDF queues with reneging," *The Annals of Applied Probability*, vol. 21, no. 2, pp. 484 – 545, 2011.

[5] S. S. Panwar and D. Towsley, "On the optimality of the ste rule for multiple server queues that serve," tech. rep., USA, 1988.

[6] J. Hong, X. Tan, and D. Towsley, "A performance analysis of minimum laxity and earliest deadline scheduling in a real-time system," *IEEE Transactions on Computers*, vol. 38, no. 12, pp. 1736–1744, 1989.

[7] S. Shakkottai and R. Srikant, "Scheduling real-time traffic with deadlines over a wireless channel," in *Proceedings of the 2nd ACM international workshop on Wireless mobile multimedia*, pp. 35–42, 1999.

[8] J. R. Haritsa, M. Livny, and M. J. Carey, "Earliest deadline scheduling for real-time database systems," tech. rep., University of Wisconsin-Madison Department of Computer Sciences, 1991.

[9] L. Zhang, Y. Cui, J. Pan, and Y. Jiang, "Deadline-aware transmission control for real-time video streaming," in *2021 IEEE 29th International Conference on Network Protocols (ICNP)*, pp. 1–6, IEEE, 2021.

[10] L.-O. Raviv and A. Leshem, "Maximizing service reward for queues with deadlines," *IEEE/ACM Transactions on Networking*, vol. 26, no. 5, pp. 2296–2308, 2018.

[11] B. Doytchinov, J. Lehoczky, and S. Shreve, "Real-time queues in heavy traffic with earliest-deadline-first queue discipline," *The Annals of Applied Probability*, vol. 11, no. 2, pp. 332 – 378, 2001.

[12] R. Atar, A. Biswas, and H. Kaspi, "Fluid limits of g/g/1+g queues under the nonpreemptive earliest-deadline-first discipline," *Mathematics of Operations Research*, vol. 40, no. 3, pp. 683–702, 2015.

[13] S. Mohanty and W. Panny, "A discrete-time analogue of the m/m/1 queue and the transient solution: a geometric approach," *Sankhyā: The Indian Journal of Statistics, Series A*, pp. 364–370, 1990.

[14] F. Farahvash and A. Tang, "Delay performance optimization with packet drop," in *2023 59th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pp. 1–7, 2023.

[15] M. Abramowitz, I. A. Stegun, and R. H. Romer, "Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables," *American Journal of Physics*, vol. 56, pp. 958–958, 10 1988.

[16] F. Farahvash and A. Tang, "Optimizing queues with deadlines under infrequent monitoring," 2024. arXiv 2403.14525.