Optimal Decision Making in Active Queue Management

Sounak Kar*, Bastian Alt†, Heinz Koeppl† and Amr Rizk‡

* EPFL, `sounak.kar@epfl.ch`
† Technical University of Darmstadt, `firstname.lastname@bcs.tu-darmstadt.de`
‡ University of Duisburg-Essen, `amr.rizk@uni-due.de`

Abstract—**Active Queue Management (AQM) aims to prevent bufferbloat and serial drops in router and switch FIFO packet buffers that usually employ drop-tail queueing. AQM describes methods to send *proactive* feedback to TCP flow sources to regulate their rate using selective packet drops or markings. Traditionally, AQM policies relied on heuristics to approximately provide Quality of Service (QoS) such as a target delay for a given flow. These heuristics are usually based on simple network and TCP control models together with the monitored buffer filling. A primary drawback of these heuristics is that their way of accounting flow characteristics into the feedback mechanism and the corresponding effect on the state of congestion are not well understood. In this work, we show that taking a probabilistic model for the flow rates and the dequeueing pattern, a Semi-Markov Decision Process (SMDP) can be formulated to obtain an optimal packet dropping policy. This policy-based AQM, named *PAQMAN*, takes into account a steady-state model of TCP and a target delay for the flows. Additionally, we present an inference algorithm that builds on TCP congestion control in order to calibrate the model parameters governing underlying network conditions. Using simulation, we show that the prescribed AQM yields comparable throughput to state-of-the-art AQM algorithms while reducing delays significantly.**

Keywords: Active Queue Management, Markov Decision Processes.

I. INTRODUCTION

Striking a balance between the two most common metrics of performance in IP networks, i.e., throughput (or utilization) and delay is a fundamental problem for routing devices. Active queue management (AQM) has evolved as a mechanism to augment prevalent end-system protocols such as TCP to tune the performance in terms of the said metrics. Most modern network switches have built-in buffers, which accumulate incoming data packets while the switch is busy processing and transmitting processed packets to the output ports [1]. A large buffer has the advantage that it can potentially minimize droptail packet loss, i.e., dropping incoming packets when the buffer is full. Understandably, this leads to higher network utilization while causing excess buffering of packets and, consequently, longer delays. This phenomenon is commonly referred to as bufferbloat [2]. Using a shallow buffer alleviates this problem, although at the expense of frequent packet drops, which may lead to diminished throughput of TCP flows[1] [4]. Evidently, the trade-off between higher utilization and lower

[1]In this work, we consider only TCP flows as they constitute the vast majority of the internet traffic according to estimates such as [3].

queueing delay is a matter of policy [5], which is delineated by the implemented AQM algorithm at the switch. The drop/admit decision usually depends on inputs such as current buffer-filling, packet delay history, or recent packet drop pattern.

In the last three decades, a range of algorithms have been proposed [6], [7], [8] to address the *AQM problem*, i.e., achieving an acceptable trade-off between link utilization and packet delay. Occasionally, these algorithms focus on additional aspects such as scalability, robustness, or fairness; for an extensive survey, see [9] and the references therein. The first AQM algorithm RED [6], proposed by Floyd and Jacobson in 1993, calculates an exponentially weighted average queue length and, as a linear function of this average, computes an initial packet drop probability in $[0, p_{\max}]$. To avoid serial drops, the initial probability is further transformed into a final drop probability that takes into the account the number of packets admitted since the last packet drop. An incoming packet is dropped according to this probability if the average queue length is between two threshold parameters for queue length and dropped/admitted deterministically otherwise. Clearly, the resulting policy depends critically on these threshold parameters and $p_{\max}$. Although RED is shown to be fairer to bursty traffic than the classic drop tail [6], *the main challenge lies in identifying the model parameters and there has been no consensus on the corresponding parameter engineering process.* To get around this problem, a range of variants and extensions of RED have been proposed, which address this issue with fairly limited success. We again refer the reader to [9] for a thorough discussion on these algorithms.

A popular AQM algorithm that claims to have successfully circumvented the problem of parameter engineering is CoDel [7]. Although CoDel is driven by two input parameters, one that signifies a *target delay* (at the AQM enabled node) and a window parameter specifying how often a packet should be dropped, their default value is hardly changed in practice. Starting with the default value of the window parameter, CoDel successively adapts its value until it meets the target delay. Although claimed to be knob-free due to the prescribed default values of the parameters under universal traffic conditions, these values can be tuned to yield superior performance for a given environment [10]. However, the analysis for the choice of the default values under varied conditions has been limited to minimal empirical investigations [10].

Different implementations exist for CoDel: although orig-

inally devised to drop a packet after it is already enqueued, it can also be implemented to obtain the current queueing delay and drop packets at the ingress depending on the information exposed on the data plane[2]. Similarly, the AQM algorithm called PIE [8] drops packets directly at the input port. PIE calculates the drop probability of an incoming packet by looking at the current queue-filling and the departure rate from the queue. This further improves the processing overhead compared to CoDel, which requires calculation of delay per packet. Additionally, the consideration of the current queue length to calculate the drop probability implies that congestion is directly controlled. Although claimed to be knob-free like CoDel, PIE actually requires default values for target delay, drop frequency and parameters for drop probability adaptation [10]. Further, the drop probability parameters are adapted according to a rule that is based on judgements. This phenomenon extends to most, if not all, popular AQM algorithms of today [10].

In light of the above, we propose a Markov Decision Process (MDP)-based approach to address the AQM problem, called PAQMAN, which only requires the target delay and the relative importance of delay violation to throughput reduction as inputs. To detect potential congestion, PAQMAN uses the current queue-filling, an estimate of the arrival rate as well as estimates of the flow RTTs. We make the case that to optimally decide on packet drops, it is necessary for an AQM algorithm to account for such a holistic description of the system state. In our framework, we encode the optimization goal through a reward function that combines delay and throughput objectives and reflects the immediate gain following a packet drop/admit decision. Consequently, we derive an optimal policy (the AQM algorithm) using tools from the MDP framework that take the state transition probabilities calculated from the model and the reward function as inputs. The policy provides the optimal decision for every possible instance of the system state. For the simpler case where the switch deals with a single flow having negligible RTT, we model the packet arrival process in greater generality, whereas a more tractable model is chosen for the general case with multiple flows. Finally, we compare the performance of PAQMAN with the state-of-the-art algorithm CoDel and the classic droptail queues. Our findings show that PAQMAN yields equivalent utilization/throughput to CoDel while minimizing delays considerably.

It should however be noted that PAQMAN, like most algorithms, is not universally applicable and is rather based on specific assumptions that are not always met in practice. This leads to following *limitations* for PAQMAN:

- The analytical underpinning of PAQMAN is derived based on TCP flows using the *same* congestion control algorithm.
- For TCP flows with non-negligible RTT, PAQMAN requires an estimate of flow RTT, which can be obtained

by passive measurement in certain cases [12], [13], [14] with considerable accuracy.
- While dealing with flows with non-negligible RTT, the analytical derivation of PAQMAN also assumes that the packet interarrival times can be modelled by exponential distribution.
- PAQMAN is based on a MDP model for which solutions suffer from the curse of dimensionality. Thus, like many AQM algorithms, the implementation of PAQMAN for many concurrent flows will impose memory requirement. A lighter version that trades accuracy for ease of implementation, e.g., through neural networks, is left as future work.
- For the multiple flows with non-negligible RTTs, the current version of PAQMAN is trained offline and implemented later.
- We do not investigate the aspect of fairness for PAQMAN and leave it for future work.

The remainder of the paper is structured as follows: we first an overview of the related work in Sect. II and subsequently formulate the AQM decision problem in Sect. III. Sect. III describes the algorithm when the switch deals with a single flow having negligible RTT, while we discuss the general case with non-negligible RTTs in Sect. V. Finally, we present numerical simulation results in Sect. VI and conclude the paper in Sect. VII.

II. RELATED WORK

AQM was developed as an additional module on top of congestion control functions with the aim to keep the congestion levels lower than traditional droptail queues. This was achieved by sending early congestion signals based on congestion indicator(s), which might include queue length, packet arrival/departure rate, flow round-trip time, link capacity, and number of flows. While the congestion signal was traditionally in form of a packet drop, ECN markings also came into use to enhance throughput. Further, the packet drop action for an AQM can be random as against deterministic packet drops in droptail queues. In this section, we compare these aspects of PAQMAN to that of the first AQM algorithm RED[6], as it formed the basis of AQM research, and the two most popular AQM as of today: PIE [8] and CoDel [7]. Occasionally, we also refer to variants of these AQMs.

As for congestion indicator, RED uses a moving average of queue length. Depending on a higher and a lower threshold parameter for queue length, an incoming packet is dropped or admitted if the (calculated) average queue length is outside these threshold, while, within these thresholds, the packet is dropped according to a drop probability that is a linear function of the average queue length. Similarly, PIE looks at the queue length and departure rate from the queue to calculate a drop probability, while CoDel uses the minimum delay of dequeued packets over an observation window as its congestion indicator. In comparison, PAQMAN uses the current packet arrival rate and the queue length to detect early congestion in the canonical case with one flow having

---

[2]This makes CoDel simpler to implement on modern programmable data plane devices [11].
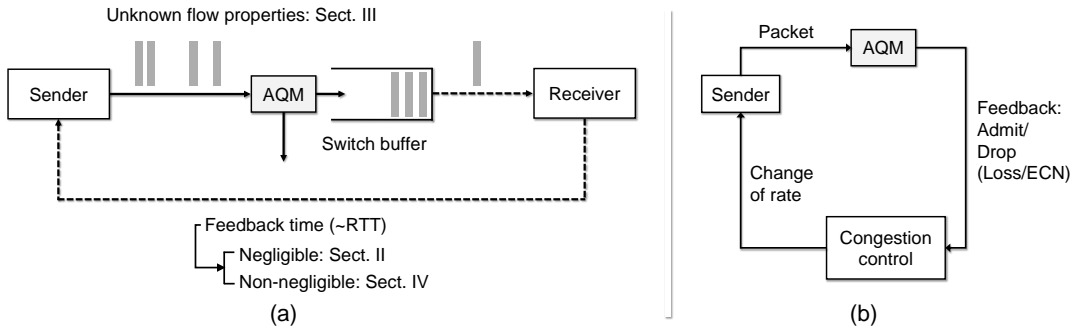
Fig. 1: Part (a) shows how PAQMAN distinguishes between different AQM scenarios: the time until the receipt of the feedback of drop/admit action is the transmission time between the switch and the sender via the receiver, which is approximated as one RTT. The case with negligible feedback time (RTT) is discussed in Sect. III and the other case is taken up in Sect. V. When traffic properties are unknown, we estimate corresponding parameters according to Sect. IV. Part (b) highlights how AQM acts in tandem with the underlying congestion control by sending early congestion signals and thereby influencing the sending rate to achieve a desired level of congestion.

negligible RTT. For the general case with multiple flows having non-negligible RTT, the flow index of the incoming packet, the time since the last decision event, and the RTTs are additionally included in the set of congestion indicators. Thus, we have a more holistic description of the state of the switch under PAQMAN, allowing us to account for more uncertainties impacting the state evolution.

In terms of packet drop action, RED can either drop randomly (according to a certain probability) or deterministically (i.e., drop or admit), depending on the average queue length. While CoDel uses deterministic drop action, PIE always calculates a drop probability to implement its random drop action. Similar to CoDel, PAQMAN's drop action is deterministic. In terms of congestion signal, packets are dropped under RED whereas they are ECN-marked in a switch deploying REM [15]. While we use packet drop for PAQMAN throughout this paper, it can also be used in conjunction with ECN marking and the corresponding changes in the derivations are straightforward. We highlight this in part (b) of Fig. 1, where the AQM is seen as an intermediate module between the sender and the congestion control mechanism of the network.

The core of the AQM algorithm can be viewed as a rule which translates the congestion indicator into the drop/admit action or, more generally, the likelihood of the same. There are primarily three approaches [9] to formulate this rule: heuristic-based, control theoretic, and deterministic optimization-based. While the early algorithms such as RED, its variants [5], [16] and the popular algorithm CoDel follow heuristic approaches, the newer algorithms (such as PIE) adopted control theoretic framework to circumvent the judgemental aspects of the former approaches such as parameter tuning. For simplicity and tractability, most of the control theoretic algorithms build on the Additive Increase Multiplicative Decrease (AIMD) principle of TCP ignoring other aspects like slow start and retransmission timeouts. Further, these works assume a linearized fluid model of TCP proposed in [17]. Some prominent examples under the control theoretic approach include: Proportional-Integral [17], Proportional-

Derivative [18], Proportional-Integral-Derivative [19], and Proportional-Integral-Enhanced [8] controller. For a thorough comparison of the algorithms under this approach, see Table VI-IX of [9]. Recently, an information compression approach has been adopted in [20], where starting from the TCP fluid model, a simpler relation between queueing delay and drop probability is formulated. The control law here specifies the change in drop probability to drive the delay towards a reference value. We note that the control theoretic approaches in general require certain input parameters whose default values are in the end set according to judgement. In comparison, PAQMAN employs an MDP-based approach and only requires the target delay and the relative importance of target delay violation to reduction in throughput as input parameters.

Works that adopt MDP as their choice of tool include [21], [22], where the authors first assume that the system dynamics is given by a deterministic fluid model for which they set out to perform a non-equilibrium analysis. Here, they first identify a Markov chain that closely approximates the behaviour of the deterministic system, i.e., estimate the transition matrix of the chain for a given level of state-space discretization. Subsequently, they adopt an MDP-based approach to derive the best policy from a set of candidate policies comprising drop-tail, RED, or a interpolated version of these two. In comparison, we use the MDP framework to derive the policy itself.

The MDP framework has been used directly for congestion control as well. While our objective is to determine a packet admission rule, the paper [23] focuses on update of congestion window using DQN. We see that most literature in this space uses closely-related but a different (non-binary) action space. For example, [24] sets the transmission rates of constituent flows in closed intervals for the next epoch, while [25] updates the congestion window for multimedia content using a POMDP framework. Optimal drop/admit actions were considered in [26] in the context of anticipative congestion control in a network of routers, where the objective is not to waste resources on processing a packet if it is likely to be dropped downstream.

In addition to delay-utilization trade-off, the aspects of drop rate, jitter and fairness also gained focus in AQM literature over the years. However, there is no standardized evaluation criteria for AQM schemes and the authors in [27] attribute the slow progress of AQM research to this fact. In addition to the measures above, the authors in [9] consider scalability, stability, responsiveness, and robustness to be crucial metrics of performance. Scalability focuses on the feasibility of implementation of an AQM algorithm as the number of flows increases, whereas stability measures, for example, the change in queue length as the number of flows varies. Further, the speed of convergence is referred to as responsiveness [28] and robustness signifies the ability of an AQM algorithm to work under diverse network conditions. Thus, robustness necessitates dynamic parameter tuning to suit changing traffic loads, i.e., variation of network parameters.

Following the inclusion of varied performance measures, RED was modified to meet the new objectives. For example, authors in [16] introduced SRED to enhance stability, ARED [5] used auto-tuning to increase robustness, and FRED [29] focused on fairness. We refer the reader to Table XIV- XXIV of [9] for a detailed comparison of the heuristic AQM schemes that address certain aspects of performance. Coming back to the issue of dynamic parameter tuning, the paper [30] introduces a general parameter tuning method that can be used in conjunction with any given AQM algorithm. Independently of the underlying AQM algorithm, the problem of parameter tuning is formulated in [30] as an MDP, where the (discretized) predicted value of the congestion indicator denotes the state of the system and the action set consists of all possible values of the AQM parameter. Taking the measured throughput-to-RTT ratio as the immediate reward, the authors adopt a Q-learning approach [31] to find the optimal AQM parameter value for a given estimate of the congestion predictor in the next time interval.

As mentioned, delay-throughput trade-off has been the primary objective of AQM algorithms. Later, most control theoretic approaches additionally focused on stability, responsiveness, and robustness by analyzing the transient response, the oscillation around the target queue length, and the steady-state error, respectively. However, fairness is usually not measured under control theoretic as well as deterministic optimization approaches. In comparison, we empirically evaluate PAQMAN in terms of delay, throughput, and the rate of convergence to steady-state behaviour. However, we do not evaluate its performance in terms of jitter, responsiveness, scalability, or robustness. [Todo: Fairness].

The algorithms that dominate the present-day AQM landscape are PIE and CoDel. They have been compared extensively to each other and to some variants of RED [32], [10]. The former paper considers both performance and scalability under multiplexing. It concludes that CoDel visibly leads to better performance in terms of delays, while the performance of PIE scales well for multiplexed flows. Similarly, the authors in [10] recommend CoDel over PIE after their extensive evaluation over the range of respective default parameters, as the empirical delay distribution under PIE was seen to have a longer tail. Hence, in this work, we evaluate PAQMAN against CoDel and droptail as they can be justly considered as AQM and non-AQM benchmarks, respectively.

Q_t: queue length immediately before t-th packet arrival,
α: common Gamma shape parameter of packet interarrival time distributions,
β_t: gamma rate parameter of the packet interarrival time immediately before t-th arrival,
S_t: state of the system observed by the packet corresponding to t-th arrival, given by (Q_t, β_t),
μ: service rate of the packets,
A_t: action taken upon t-th arrival, i.e. admit or drop,
$\mathsf{P}(S_{t+1}|S_t, A_t)$: transition probability to state S_{t+1} from S_t given A_t action was taken,
$R(S_t, A_t)$: reward when action A_t is taken in state S_t,
$\tau(S_t, A_t)$: expected transition time when action A_t is taken in state S_t,
X_t: time of t-th packet arrival.

TABLE I: Notation to define the AQM Problem in Sect. III

III. AQM AS AN OPTIMAL DECISION PROBLEM

In the following, we formulate the AQM problem as finding an optimal policy of a Semi Markov Decision Process where the underlying system is essentially an AQM-capable router or switch that carries IP traffic. We recognize that the problem can be framed in various ways depending upon the flow RTT and flow properties. Such distinctions, e.g., with respect to flow RTT can naturally arise during intra-datacenter or inter-datacenter communications as depicted in Fig. 1. Under the scenario of negligible RTT, we allow the arrival process to have a more general form whereas, keeping tractability in sight, a simpler model is adopted for the case with non-negligible RTT. In this section, the former is described in greater detail and we take up the latter in Sect. V.

In our framework where negligible RTT is assumed, the buffer is observed at arrival instants[3] and we aim to find out the optimal policy, i.e., whether it is ideal to drop or admit a packet given the state of the system. Apart from the action, the state evolution of the system is influenced by the packet interarrival time and the service time distributions. We formulate the problem using a model of the packet data arrivals that is given by gamma distributed interarrival times where the parameters of the distribution depend on the history. Our choice of distribution for the interarrival times allows us to fit a wide class of observed traffic flows to the model. Further, the service times are assumed to be exponentially distributed. Before formalizing the system description and the framework in general, we emphasize that optimality here is defined in terms of expected long term average reward. The expected long term average reward can be thought of as the long-term accumulation rate of instant rewards where the instant reward can be specified precisely depending upon

---

[3]We use instant and epoch interchangeably

the intended objective such as a function of throughput and/or delay. Specifically, we formulate the reward function to capture the immediate gain in relative throughput while the given delay threshold is adhered to. The required notations to formally define the AQM problem are introduced in Table I.

Further, we denote the state space by $\mathcal{S}$ and the action space by $\mathcal{A} = \{0, 1\}$ where 0 and 1 denote admitting and dropping of a packet, respectively. These notations are used across sections with minor variation which is mentioned in respective contexts[4].

Looking at the consequence of available actions, we see that admitting a packet causes the queue length to increase by one and the effective arrival rate of the TCP flow that is given as β_t/α increases to $f_u(\beta_t/\alpha)$ immediately[5]. Here, the function f_u is dependent upon the exact TCP congestion control algorithm. For example, for an additive increase multiplicative decrease (AIMD) algorithm [4], $f_u(\beta_t/\alpha) = \beta_t/\alpha + a$, for some $a > 0$. For tractability, we assume the shape parameter α remains fixed and vary the rate parameter β appropriately to reflect this change. This implies that following an admission action, β_{t+1} assumes the value $\alpha f_u(\beta_t/\alpha)$.

In contrast, packet drops evidently do not change the queue length although the effective arrival rate β_t/α drops to $f_d(\beta_t/\alpha)$ immediately where f_d is dependent upon the exact TCP congestion control algorithm. Again, for an AIMD algorithm, $f_d(\beta_t/\alpha) = b\beta_t/\alpha$, for some $0 < b < 1$. As before, we assume the shape parameter remains fixed and take $\beta_{t+1} = b\beta_t$. For the sake of simplicity, we derive our results with the simplest version of AIMD algorithm which uses $a = 1$ and $b = 1/2$. Our results remain valid under a wide class of elementary congestion control functions and can be obtained by replacing the increment and decrement of flow arrival rate with corresponding f_u and f_d.

To formalize the state evolution under the given AIMD algorithm, the action $A_t = 0$ causes the queue length to increase to $(Q_t + 1)$ instantaneously and the possible states in the next arrival epoch could be any of the elements of the set: $\mathcal{Q}_0 = \{(q, \beta_t + \alpha) : 0 \le q \le Q_t + 1\}$. That is, $S_{t+1} \in \mathcal{Q}_0$ and $\mathsf{P}((q, \beta_t + \alpha)|S_t, 0)$ is the probability that exactly $Q_t + 1 - q$ many packets are served until next packet arrival. Similarly, for the action $A_t = 1$, we have $\mathcal{Q}_1 = \{(q, \beta_t/2) : 0 \le q \le Q_t\}$ and $\mathsf{P}((q, \beta_t/2)|S_t, 1)$ is the probability of serving $Q_t - q$ many packets until the next packet arrival. The transition probabilities to any state outside these designated sets are zero. Next, we derive the expression for state transition probabilities which dictate the pattern of transition between the system states. These transition probabilities are crucial inputs to the optimal policy derivation process.

A. State Transition Probabilities

Recall that the system state is described as the vector consisting of the current queue length and the arrival rate, i.e.,

---

$S_t = (Q_t, \beta_t)$. We are interested in deriving an expression for the transition probabilities $\mathsf{P}(S_{t+1}|S_t, A_t)$ as these are inputs to the Bellman operator which iteratively determines the value of each state. The value of each system state in turn determines the policy; see Chap. 7 of [33] for details. To that end, we state the following lemma.

Lemma III.1. *For two independent Gamma random variables, $Y_{u,v} \sim Gamma(u,v)$ and $X_{w,z} \sim Gamma(w,z)$,*
$\mathsf{P}(Y_{u,v} > X_{w,z}) = \mathsf{P}(Y_{u-1,v} > X_{w,z}) + \frac{\Gamma(u+w-1)}{\Gamma(u)\Gamma(w)}\left(\frac{v}{v+z}\right)^{u-1}\left(\frac{z}{v+z}\right)^w$, *where $u > 1$. This implies*

$$\mathsf{P}(Y_{u,v} > X_{w,z}) = \sum_{k=0}^{u-1} \frac{\Gamma(k+w)}{\Gamma(k+1)\Gamma(w)}\left(\frac{v}{v+z}\right)^k\left(\frac{z}{v+z}\right)^w.$$

Proof. The proof is given in Sect. VIII. $\square$

Now, we can derive the transition probabilities as follows.

Proposition III.2. *Given the state $S_t = (Q_t, \beta_t)$ at t-th arrival epoch, Q_t and β_t being the queue length and arrival rate, respectively, the transition probabilities to the state $S_{t+1} = (Q_{t+1}, \beta_{t+1})$ in the next epoch under the action $A_t = 0$ are given by*

$$\mathsf{P}((Q_{t+1}, \beta_{t+1})|S_t, 0) = \mathbb{1}_{\beta_{t+1}=\beta_t+\alpha}$$
$$\frac{\Gamma(Q_t + 1 - Q_{t+1} + \alpha)}{\Gamma(Q_t + 2 - Q_{t+1})\Gamma(\alpha)}\left(\frac{\mu}{\mu+\beta_{t+1}}\right)^{Q_t+1-Q_{t+1}}\left(\frac{\beta_{t+1}}{\mu+\beta_{t+1}}\right)^\alpha,$$

for $1 \le Q_{t+1} \le Q_t + 1$ and

$$\mathsf{P}((0, \beta_{t+1})|S_t, 0) = \mathbb{1}_{\beta_{t+1}=\beta_t/2}$$
$$\left(1 - \sum_{k=0}^{Q_t} \frac{\Gamma(k+\alpha)}{\Gamma(k+1)\Gamma(\alpha)}\left(\frac{\mu}{\mu+\beta_{t+1}}\right)^{Q_t+1-k}\left(\frac{\beta_{t+1}}{\mu+\beta_{t+1}}\right)^\alpha\right).$$

Proof. See Sect. VIII. $\square$

The transition probabilities for the case when a packet is dropped are derived similarly. Note that only the rate parameter of the interarrival time variable changes to $\beta_t/2$ due to the drop and the possible queue length in the next decision epoch is at most Q_t. Thus,

$$\mathsf{P}((Q_{t+1}, \beta_{t+1})|S_t, 1) = \mathbb{1}_{\beta_{t+1}=\beta_t/2}$$
$$\frac{\Gamma(Q_t - Q_{t+1} + \alpha)}{\Gamma(Q_t + 1 - Q_{t+1})\Gamma(\alpha)}\left(\frac{\mu}{\mu+\beta_{t+1}}\right)^{Q_t-Q_{t+1}}\left(\frac{\beta_{t+1}}{\mu+\beta_{t+1}}\right)^\alpha,$$

for $1 \le Q_{t+1} \le Q_t$ and

$$\mathsf{P}((0, \beta_{t+1})|S_t, 1) = \mathbb{1}_{\beta_{t+1}=\beta_t/2}$$
$$\left(1 - \sum_{k=0}^{Q_t-1} \frac{\Gamma(k+\alpha)}{\Gamma(k+1)\Gamma(\alpha)}\left(\frac{\mu}{\mu+\beta_{t+1}}\right)^{Q_t-k}\left(\frac{\beta_{t+1}}{\mu+\beta_{t+1}}\right)^\alpha\right).$$

Unlike the MDP framework, the expected time between two decision epochs plays a significant role in the SMDP framework that forms the basis of our analysis. Since drop/admit action affects the packet sending rate of TCP, the expected time between two arrival events varies. We see that the expected transition times are: $\tau(S_t, 0) = \alpha/(\beta_t + \alpha)$ and
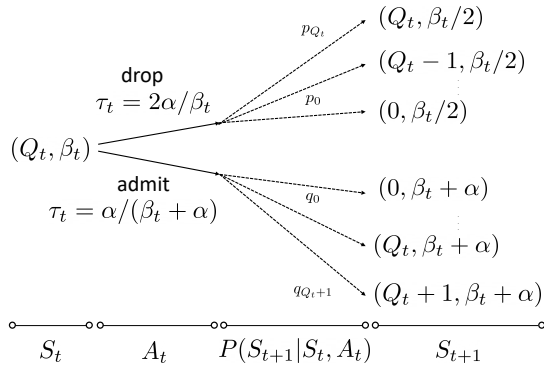
---

[4]In Sect. V (non-negligible RTT case), β_t denotes the rate parameter of the exponentially distributed interarrival time *at t-th arrival*.

[5]In contrast, the change in arrival rate occurs with a lag when RTT is non-negligible; see Sect. V for details.

Fig. 2: Starting from a system state (Q_t, β_t), i.e., the current buffer filling and the current arrival rate parameter, the figure shows the state transition after packet admit/drop action. Solid arrows represent actions, whereas dotted arrows signify subsequent state transitions annotated with corresponding probabilities. The expected time until the next decision, i.e., the mean interarrival time is denoted by τ_t.

$\tau(S_t, 1) = 2\alpha/\beta_t$, following the AIMD principle of TCP. We occasionally abbreviate $\tau(S_t, A_t)$ as τ_t. A brief sketch of state transitions in this framework is provided in Fig. 2.

B. Reward function

Equipped with the transition probabilities, we now focus on choosing the reward function which shapes the objective of the policy learning process. The reward function represents the immediate incentive and the policy learning process aims to maximize the average accumulated reward in the long run. To align the learning process with our objective of maximizing throughput given a user-defined delay constraint, we use the reward function described in the following. The reward function penalizes heavily whenever the delay constraint is breached and provides immediate incentive that equals the change in the square rate of throughput. The rationale here is to reflect the relative change rather than its absolute value. Formally,

$$R(S_t, 0) = -M\mathbb{1}_{\{(Q_t+1)/\mu > \eta\}} + \left(\sqrt{\frac{\beta_t + \alpha}{\alpha}} - \sqrt{\frac{\beta_t}{\alpha}} \right) ,$$

$$R(S_t, 1) = -M\mathbb{1}_{\{Q_t/\mu > \eta\}} + \left(\sqrt{\frac{\beta_t}{2\alpha}} - \sqrt{\frac{\beta_t}{\alpha}} \right) ,$$
(1)

where η denotes a target delay threshold and M is a large penalty value for breaching it. Thus, the reward function adopts the goal of state-of-the-art heuristic AQM algorithms that take a target delay as a single input variable [34].

Equipped with the framework above and the reward function, we focus on finding the AQM policy using the value-iteration algorithm for SMDP's. The AQM policy runs on the buffer and takes the system state (Q_t, β_t), essentially, the current buffer filling and the current arrival rate parameter as input and provides the action, i.e., whether to drop or admit to maximize accumulated predefined reward.

Formally, a policy π is defined as a mapping $\pi : \mathcal{S} \mapsto \mathcal{A}$, i.e., given the system state in terms of buffer filling and arrival rate, the policy returns one of the possible actions, i.e., admission or dropping. Let the accumulated reward up to time x be denoted by $Z(x)$, i.e., $Z(x) = \sum_{t=1}^{V(x)} R(S_t, A_t)$, where $V(x) = \max\{t : X_t \leq x\}$, and X_t is the arrival time of the t-th packet. The expected average long-term reward is then given by:

$$g_i(\pi) = \lim_{x \to \infty} \frac{1}{x} \mathsf{E}_{i,\pi}[Z(x)] ,$$
(2)

where i denotes the fact that the initial state was S_i and π is the used policy.

As mentioned, our objective is to find a policy π which maximizes g for each state S_i. To that end, we convert the SMDP problem to a discrete-time MDP using the data transformation method [33] by appropriately scaling the rewards and transforming the transition probabilities. This is due to the fact that an SMDP with average reward criterion can be converted for the purpose of solving it to a discrete time MDP where the rewards are modified to reward accumulation rate and the transition probabilities are adjusted to reflect the changes from continuous time to discrete time transformation. Subsequently, we use the value iteration method for discrete time MDP's to determine the optimal policy. The transformed reward function $\bar{R}$ and transition probability $\bar{\mathsf{P}}$ are respectively given by:

$$\bar{R}(S_t, A_t) = \frac{R(S_t, A_t)}{\tau(S_t, A_t)} ,$$

$$\bar{\mathsf{P}}(S_{t+1}|S_t, A_t) = \mathsf{P}(S_{t+1}|S_t, A_t)\frac{\tau}{\tau(S_t, A_t)}, \ S_{t+1} \neq S_t ,$$

$$\bar{\mathsf{P}}(S_t|S_t, A_t) = \mathsf{P}(S_t|S_t, A_t)\frac{\tau}{\tau(S_t, A_t)} + 1 - \frac{\tau}{\tau(S_t, A_t)} ,$$
(3)

where τ is chosen such that $0 < \tau \leq \min_{s,a} \tau(s, a)$.

IV. AQM FOR UNKNOWN TRAFFIC FLOW PROPERTIES

In this section, we consider the problem of finding the optimal AQM policy for unknown traffic flow properties. As established in the Sect. III, the optimal AQM policy requires accurate knowledge of the current traffic arrival rate which is determined by the initial arrival rate and the series of drop or admit actions. Taking the current arrival rate and the queue filling as inputs, the AQM policy then prescribes whether it is optimal to drop or admit an incoming packet. Therefore, to propose a policy under unknown arrival characteristics, we first need to infer the respective parameters of the arrival flows. However, estimating the model parameters while learning the optimal policy introduces the classical problem of dual control [35]. However, this can be alleviated by an exploration-exploitation Bandit-heuristic [36], which is computationally tractable.

A. Estimation of Traffic Flow Parameters

Next, we infer the parameters governing the arrival process at data packet arrivals. To that end, we derive Maximum Likelihood Estimates (MLE) of arrival shape α and arrival

rates β_t as used by the parametric models in Sect. III. Let $\{X_n\}$ and $\{A_n\}$ denote the sequence of arrival times and the actions respectively. Further, the corresponding packet interarrival times are denoted by W_t, i.e., $W_t = X_{t+1} - X_t$. Recall that $A_t = 1$ signifies a packet drop and $A_t = 0$ denotes packet admission. We take a tractable model of Gamma interarrival times, i.e.

$$W_t | \alpha, \beta_t \sim \text{Gamma}(\alpha, \beta_t) ,$$
$$\beta_{t+1} = (\beta_t + \alpha)^{1-A_t} (\frac{\beta_t}{2})^{A_t} , \tag{4}$$

following the AIMD congestion control principle of TCP. Given the sequence of interarrival times $\mathbf{W} = \{W_n\}_{n=1}^k$ and actions $\mathbf{A} = \{A_n\}_{n=1}^k$, we aim to estimate the common shape parameter α and the initial rate parameter β_1 to derive the transition probabilities $\mathsf{P}(S_{t+1} | S_t, A_t)$ described in Sect. III to find the optimal policy. The likelihood of the parameters for the observed sequence $(\mathbf{W}, \mathbf{A})$ is given by

$$L(\alpha, \beta_1 | \mathbf{W}, \mathbf{A}) = \frac{(\prod_{n=1}^k \beta_n)^\alpha (\prod_{n=1}^k W_n))^{\alpha-1} e^{-\sum_{n=1}^k \beta_n W_n}}{(\Gamma \alpha)^k} .$$

Note that the RHS is simply product of Gamma densities and the dependence on $\mathbf{A}$ is implicit. Now, taking logarithm we obtain the log-likelihood $l(\alpha, \beta_1)$ as

$$l(\alpha, \beta_1) = \alpha \sum \log \beta_n + (\alpha - 1) \sum \log W_n$$
$$- \sum \beta_n W_n - k \log(\Gamma \alpha) . \tag{5}$$

To optimize the likelihood we obtain

$$\frac{\partial l}{\partial \alpha} = \sum \log \beta_n + \sum \left(\frac{\alpha}{\beta_n} - W_n \right) \frac{\partial \beta_n}{\partial \alpha}$$
$$+ \sum \log W_n - k \frac{\partial}{\partial \alpha} \log(\Gamma \alpha) , \tag{6}$$
$$\frac{\partial l}{\partial \beta_1} = \sum \left(\frac{\alpha}{\beta_n} - W_n \right) \frac{\partial \beta_n}{\partial \beta_1} ,$$

where

$$\frac{\partial \beta_n}{\partial \alpha} = \left(1 + \frac{\partial \beta_{n-1}}{\partial \alpha} \right)^{1-A_{n-1}} \left(\frac{1}{2} \frac{\partial \beta_{n-1}}{\partial \alpha} \right)^{A_{n-1}} ,$$
$$n = 3, 4, \ldots, k,$$

with $\frac{\partial \beta_2}{\partial \alpha} = \mathbb{1}_{\{A_1 = 0\}}$ and $0^0 = 1$, by convention.

Further,

$$\frac{\partial \beta_n}{\partial \alpha} = \left(\frac{\partial \beta_{n-1}}{\partial \beta_1} \right)^{1-A_{n-1}} \left(\frac{1}{2} \frac{\partial \beta_{n-1}}{\partial \beta_1} \right)^{A_{n-1}}$$
$$= \left(\frac{1}{2} \right)^{A_{n-1}} \frac{\partial \beta_{n-1}}{\partial \beta_1} , \quad n = 2, 4, \ldots, k .$$

To get the maximum likelihood estimates $\hat{\alpha}$ and $\hat{\beta}_1$, we numerically find the zero point of (6) that maximizes (5).

B. Inference under unknown TCP Congestion Control

In case the exact congestion control algorithm of TCP is not known beforehand, we assume the effective arrival rate β_{t+1}/α in the next epoch is changed according to a polynomial function (e.g., TCP CUBIC [37]) of the present effective arrival rate β_t/α. We can then rewrite (4) as

$$W_t | \alpha, \beta_t \sim \text{Gamma}(\alpha, \beta_t) ,$$
$$\frac{\beta_{t+1}}{\alpha} = \left(f_u \left(\frac{\beta_t}{\alpha} \right) \right)^{1-A_t} \left(f_d \left(\frac{\beta_t}{\alpha} \right) \right)^{A_t} , \tag{7}$$

where f_u and f_d are polynomials governing the change of the arrival rate following an admit and drop, respectively. The log-likelihood has a similar form as shown above, although estimates of the coefficients of the polynomials, together with α and β_1, are required. This can either be done numerically or by calculating partial derivatives explicitly, similar to (6). Subsequently, one can find their zeros corresponding to the maxima, albeit with additional equations for $\partial l / \partial c_j$, for each coefficient c_j. For example, if f_u is a cubic polynomial and f_d is a linear function, this method will lead to calculation of six additional partial derivatives.

Using the parameter estimates along with the observed series of drop or admit actions, we infer the current arrival rate $\hat{\beta}_t$ which together with the current queue filling Q_t describes the system state. We thus use the AQM policy similar to Sect. III to find the optimal action.

V. AQM UNDER NON-NEGLIGIBLE RTT

In this section, we focus on the case where the RTT between the sender and the receiver of a flow is non-negligible. As mentioned in Sect. III, we trade model flexibility for tractability for this case. This tractability is essential given the case where the switch acts on multiple concurrent flows. We derive our policy for the single flow case in Sect. V-A whereas the multiple flow case is described in Sect. V-B. As before, our objective is to maximize the total throughput subject to a user-defined delay constraint.

A. Single Flow

In the following, we assume that both packet interarrival times and the packet service times are exponentially distributed. Further, the RTT is denoted as r and we have $r > 0$. We assume that an estimate of the exact flow RTT is known to the switch. Note that the sender adapts the sending rate, i.e., the arrival rate to the switch, only after it receives a congestion signal from the receiver, which has a propagation time of $r/2$.

Thus, the delay between an action at the switch and the arrival of the congestion signal at the sender corresponds to the propagation time from the switch to the sender via the receiver, which approximately equals r. Thus an action takes effect at the switch only after a time period r, and all arrivals meanwhile are admitted given there is enough room in the output buffer. For tractability, we further assume that the arrivals and the corresponding admit actions in this period have no direct consequence on the way arrival rate is changed. Although an approximation, we expect this assumption to have
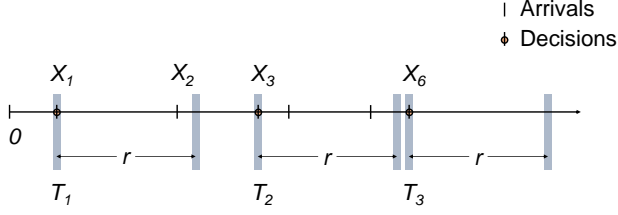
Fig. 3: Arrival times X_i and decision times T_t are shown on the time axis. The first job arrives at time X_1 with rate β_1. After X_1, X_2 is the only arrival in $(X_1, X_1+r]$ and is assumed to have no bearing on the arrival rate. We see $X_3 = T_2$ and the arrival rate is changed to β_2 at T_2 according to the action taken at X_1. The rate again changes to β_3 at $T_3 = X_6$.

no major consequence and take advantage of the corresponding simpler state representation.

Recall that we denote the arrival times as $\{X_s\}_{s \in \mathbb{N}}$. We take a decision whether to admit or drop on the first packet arrival at time X_1. Thereafter all arrivals in the interval $(X_1, X_1+r]$ are admitted subject to enough room in the buffer and the first arrival after $(X_1 + r)$ can be potentially dropped. The arrival rate changes at this new arrival after $(X_1 + r)$ according to the action taken at time X_1. Subsequently, the decision making process goes dormant for another time period of length r and the drop/admit decision is taken again at the first arrival after this period. The change in arrival rate also takes place at this time. An illustrative example is given in Fig. 3. We call the arrival epochs where a drop decision can be possibly made as *decision times* and denote them as $\{T_t\}_{t \in \mathbb{N}}$. As explained, T_i's ($i \geq 2$) are also the time points where the arrival rate changes according to the action taken at T_{i-1}.

The service times are assumed independently and identically distributed (iid) as $\text{Exp}(\mu)$ and given the relevant arrival rate parameter β_k, we have

$$X_{i+1} - X_i \sim \text{Exp}(\beta_k), \ i \in \mathbb{N},$$

for some $k \in \mathbb{N}$. Note that, unlike Sect. III, the arrival rates do not change at every arrival. Rather *the change happens at the decision times* owing to the delay factor of r and the fact that intermediate arrivals do not influence the arrival rate directly. Focusing on the inter-decision time, we observe that the residual time to the next decision time after one RTT r is also distributed as $\text{Exp}(\beta_j)$, for some $j \in \mathbb{N}$, following the memoryless property of exponential distribution. Further, the change in the arrival rate β happens only at the decision times. Therefore, we know for the inter-decision times

$$T_{i+1} - T_i | \beta_i \sim r + \text{Exp}(\beta_i), \ i \in \mathbb{N}, \tag{8}$$

as β_i is the arrival rate during the interval $[T_i, T_{i+1})$.

To derive the optimal decision at times points T_i, we adopt the SMDP framework similar to the derivations of Sect. III. We retain much of our notations from Sect. III with the exception of β_t which denotes the rate parameter of packet interarrival

times *at* t-th decision epoch. In contrast to Sect. III, β_t in this case is already known at $(t-1)$-th arrival epoch and, hence, is a valid input to the policy.

Note that in the time interval $(T_i, T_i + r]$, the queue length grows as the population of a birth-death process, where the transition matrix $\text{P}(s)$ over a time interval of length s is given by e^{sG_j}. Here, G_j is the $(L+1) \times (L+1)$ intensity matrix of the process in the relevant interval where L is the buffer size. It is given by

$$G_j = \begin{bmatrix} -\beta_j & \beta_j & 0 & \cdots & \cdots \\ \mu & -\mu - \beta_j & \beta_j & 0 & \cdots \\ 0 & \mu & -\mu - \beta_j & \beta_j & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & \cdots & \cdots & \mu & -\mu \end{bmatrix},$$

for some j. Further, in the interval $(T_i + r, T_{i+1})$ there can only be packet departures and hence the transition matrix for any interval of length s is given by e^{sG} where

$$G = \begin{bmatrix} 0 & 0 & 0 & \cdots & \cdots \\ \mu & -\mu & 0 & 0 & \cdots \\ 0 & \mu & -\mu & 0 & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & \cdots & \cdots & \mu & -\mu \end{bmatrix}. \tag{9}$$

The following fact helps us derive the transition matrix between two consecutive decision epochs.

Fact V.1. *If all eigenvalues of the matrix A are positive,*

$$\int_0^\infty e^{-uA} du = A^{-1} .$$

Proof. This well-known result follows from the fact that $\lim_{t \to \infty} e^{-tA} = 0$ for a positive definite matrix A. $\square$

We now derive the transition matrix for the queue length over the interval between two consecutive decision epochs. The first step towards this is to find the transition probabilities for the queue length as the change of arrival rate following a transition is deterministic.

Proposition V.2. *The transition probability matrix P_i for queue length Q over the interval (T_i, T_{i+1}) is given by*

$$P_i = \beta_i e^{rG_i}(\beta_i I - G)^{-1}.$$

Proof. See Sect. VIII. $\square$

Observe that $\text{P}(Q_{t+1}|Q_t, 1)$ corresponds to the Q_t-th row of the transition matrix over the interval $[T_t, T_{t+1})$. Under $A_t = 1$, the transition matrices over intervals $[T_t, T_{t+1})$ and (T_t, T_{t+1}) are identical as the incoming packet is dropped. Hence, $\text{P}(Q_{t+1}|Q_t, 1)$ is given by the (Q_t, Q_{t+1})-th element of P_t. However, under $A_t = 0$, the queue length jumps to $\max(Q_t+1, L)$ at time T_t and the the $(Q_t, \max(Q_t+1, L))$-th element represents $\text{P}(Q_{t+1}|Q_t, 0)$. Finally, the state transition probabilities can be expressed as follows:

$$\begin{aligned} \text{P}(S_{t+1}|S_t, 0) &= \text{P}(Q_{t+1}|Q_t, 0) \mathbb{1}_{\beta_{t+1}=\beta_t+1}, \\ \text{P}(S_{t+1}|S_t, 1) &= \text{P}(Q_{t+1}|Q_t, 1) \mathbb{1}_{\beta_{t+1}=\beta_t/2}. \end{aligned} \tag{10}$$

Further, similar to (1), we define the reward function as:

$$R(S_t, 0) = -M \mathbb{1}_{\{(Q_t+1)/\mu > \eta\}} + \left(\sqrt{\beta_t + 1} - \sqrt{\beta_t} \right),$$

$$R(S_t, 1) = -M \mathbb{1}_{\{Q_t/\mu > \eta\}} + \left(\sqrt{\beta_t/2} - \sqrt{\beta_t} \right). \tag{11}$$

Also, it is immediate from (8) that the expected inter-decision time $\tau(S_t, A_t) = r + 1/\beta_t$. Equipped with the transition probabilities, reward function and expected transition times, we can now transform the SMDP problem into a discrete time MDP problem like Sect. III and perform value iteration on the transformed MDP to find out the best policy.

B. Multiple Flows

In this section, we extend our SMDP formulation from Sect. V-A in presence of concurrent flows with different RTT. We assume that the flow RTT's are known or estimated at the switch running AQM.

We define the decision times for a flow of interest as the instances of the first packet arrivals that are at least an RTT apart. That is for a fixed flow, when an AQM decision is carried out for one packet, the subsequent packets within one RTT window are admitted. The next AQM decision is taken for the first packet after the RTT window has passed. We will see in the following that, unlike the single flow case, the time since the last decision epoch involving a packet from each flow plays an important role in formalizing the framework here.

Let us denote the packet arrival times for the j-th flow as $\{X_{js}\}$, with packet index $s \in \mathbb{N}$ and, similar to the single flow case, we assume for the jth flow

$$X_{j(i+1)} - X_{ji} | \beta_{jk} \sim \text{Exp}(\beta_{jk}), \ i \in \mathbb{N},$$

for some $k \in \mathbb{N}$, $j \in [n]$ and $[n] = \{1, 2, \ldots, n\}$, n being the number of concurrent flows. The decision times are denoted as $\{T_s\}_{s \in \mathbb{N}}$ and we see that the mean arrival rate for each flow remains constant between two decision epochs. This is because we assume that the AQM signal reaches the sender after an RTT.

Let $\boldsymbol{\beta}_t = (\beta_{1t}, \beta_{2t}, \ldots, \beta_{nt})^T$ be the parameter vector of packet interarrival times at the t-th decision epoch and $\boldsymbol{r} = (r_1, r_2, \ldots, r_n)^T$ denote the vector of flow RTTs. Further, we express the *age* vector at t-th decision epoch by $\boldsymbol{u}_t = (u_{1t}, u_{2t}, \ldots, u_{nt})^T$ where u_{jt} denotes the time since the last decision epoch concerning a packet from the j-th flow. In absence of a last decision time concerning flow j, i.e., before the arrival of the first packet from flow j, we define $u_{jt} = r_j$. Observe that if the t-th decision epoch involves a packet from flow j, we have $u_{jt} = 0$. An illustration of the decision epochs for the case of three concurrent flows is provided in Fig. 4.

At the t-th decision epoch, let Y_{jt} denote the time until the next AQM decision on flow j. In other words, Y_{jt} indicates the time until the arrival of a packet from flow j that is at least one RTT after the last decision epoch from flow j. Given the RTT r_j of flow j, we can write
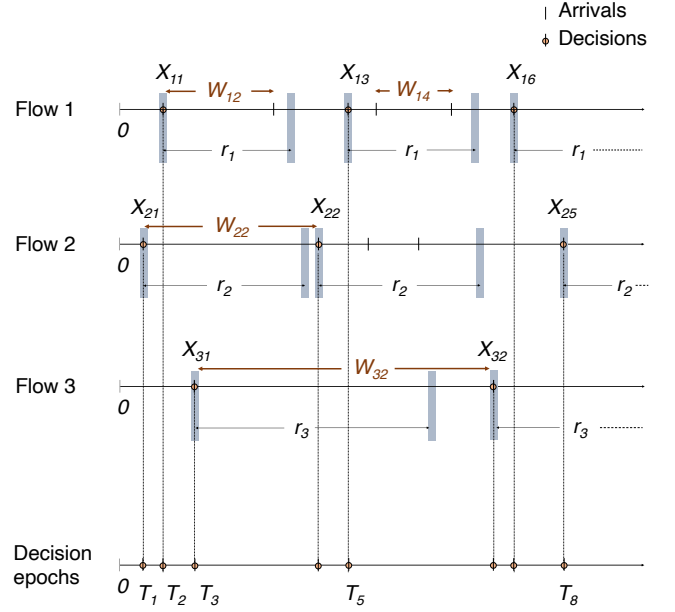
$$Y_{jt} = W_j + \max(0, r_j - u_{jt}),$$

Fig. 4: Illustration of decision times when three flows pass through the switch buffered link. The first decision time is the first arrival, which takes place at $T_1 = X_{21}$. Next decision time concerning flow 2 cannot be earlier than $X_{21} + r_2$. The first arrival from flow 1 or 3 can potentially be the next decision time, which turns out to be $T_2 = X_{11}$. Likewise, the decision process on flow 1 goes dormant in $(X_1, X_1 + r]$. Note that decision times are the times where the arrival rates change for the associated flow. The interarrival time W_{ij} is exponentially distributed with a parameter that was fixed at the last decision time concerning flow i, according to the action taken at the second last decision time on that flow.

where W_j denotes the residual time to the packet arrival from j-th flow after r_j amount of time has passed since the last decision epoch involving flow j. By the memoryless property of exponential distribution we know that $W_j \sim \text{Exp}(\beta_{jt})$, for $j \in [n]$. Thus, the time until the $(t+1)$-st decision epoch across all flows is given by

$$Y_t = \min_{j \in [n]} Y_{jt}. \tag{12}$$

Therefore, we can write the inter-decision time $T_{t+1} - T_t = Y_t$ as

$$Y_t | \boldsymbol{\beta}_t, \boldsymbol{u}_t \sim \min_{j \in [n]} \{W_j + \max(0, r_j - u_{jt})\}, \ t \in \mathbb{N}.$$

Equipped with the description above we now focus on formalizing the SMDP to derive the optimal AQM action corresponding to any given switch state. Apart from the individual flow arrival rates and the queue length, the state description in this case includes the *age* vector $\boldsymbol{u}_t$ and the flow index of the incoming packet. To formulate the AQM problem as an SMDP, we use the notations introduced in Sect. III with the following exceptions:

$\boldsymbol{\beta}_t$: flow-wise rate parameter *vector* of packet interarrival times *at* the t-th decision epoch,

$\boldsymbol{u}_t$: flow-wise age *vector*, i.e., time since last decision epoch involving a packet from the concerned flow at the t-th decision epoch,

S_t: state of the system observed by the packet corresponding to the t-th decision epoch, given by $(j, Q_t, \boldsymbol{\beta}_t, \boldsymbol{u}_t)$ where j denotes the flow index of the packet for which an AQM decision is to be taken.

With the flow-wise age $\boldsymbol{u}_t$ being a continuous variable we derive next the corresponding transition kernels. Let $B = (j, Q_{t+1}, \boldsymbol{\beta}_{t+1}, U)$ denote the set that the system state at $t + 1$, i.e., S_{t+1}, can possibly belong to. As usual, we restrict our focus on $U \in \mathcal{B}(\mathbb{R}^n)$, the Borel sigma algebra over $\mathbb{R}^n$. Due to the known fact that right semi-closed rectangles generate $\mathcal{B}(\mathbb{R}^n)$, it is sufficient to consider the sets U having the form $U = I_1 \times I_2 \times \ldots I_n$ where each I_j is a right semi-closed interval.

Going back to (12), we see that the value of the age vector at the decision epoch $(t+1)$ is given by $\boldsymbol{u}_t + Y_t$. This motivates us to define the translates

$$I'_j = \{x : x + u_{jt} \in I_j\} , \ j \in [n] .$$

Subsequently, the transition kernel can be written as:

$$\mathsf{P}(S_{t+1} \in B | S_t, 0) = \mathbb{1}_{\boldsymbol{\beta}_{t+1} = \boldsymbol{\beta}_t + e_j}$$
$$\mathsf{P}\left(Y_t \in \bigcap_{k \in [n]} I'_k, Y_t = Y_{jt}, V_t = Q_{t+1} - Q_t | A_t = 0\right),$$
$$\mathsf{P}(S_{t+1} \in B | S_t, 1) = \mathbb{1}_{\boldsymbol{\beta}_{t+1} = h(\boldsymbol{\beta}_t, j)}$$
$$\mathsf{P}\left(Y_t \in \bigcap_{k \in [n]} I'_k, Y_t = Y_{kt}, V_t = Q_{t+1} - Q_t | A_t = 1\right),$$

(13)

where V_t denotes the change in queue length in the interval $[T_t, T_{t+1})$ and $h(\boldsymbol{x}, j) = (x_1, x_2, \ldots, x_i/2, \ldots, x_n)$ and e_j denotes the unit vector whose j-th coordinate equals 1. Essentially, the indicator function involving $\boldsymbol{\beta}_t$ in the RHS of (13) denotes the fact that the arrival rate can only change to specific values which are determined by the drop or admit action and the AIMD principle of TCP. Further, $(t + 1)$-th decision epoch involves flow j iff the corresponding inter-decision time is $Y_t = Y_{jt}$. Finally, the age vector $\boldsymbol{u}_{t+1}$ in the

next epoch belongs to the set U iff $Y_t \in I_k - u_{kt}$, $1 \le k \le n$. We further introduce the following shorthand notations:

$$\bigcap_{j \in [n]} I'_j = (a, b], \ \text{when} \ \bigcap_{j \in [n]} I_j \ne \phi ,$$
$$\max(0, r_j - u_{jt}) := c_j, \quad j \in [n] .$$

Given a decision at time zero we use c_j to denote the time span during which there cannot be a decision concerning flow j. In particular, this time span equals the RTT as $c_i = r_i$, since $S_t = \{i, Q_t, \boldsymbol{\beta}_t, \boldsymbol{u}_t\}$. Now lets order the time spans as $c_{i_1} \le c_{i_2} \le \cdots \le c_{i_n}$. Evidently, for $(a, b] \subset (-\infty, c_{i_1}]$, we have for the time to the next decision of the flow i_1 that $\mathsf{P}(Y_t \in (a, b]) = 0$. Further, for $(a, b] \subset (c_{i_l}, c_{i_{l+1}}]$ we can express the probability of the time to the next decision falling in this time span as $(*)$, where we have used the shorthand:

$$G_j(a, b) = P(Y_t \in (a, b], Y_t = Y_{jt}, V_t = Q_{t+1} - Q_t) .$$

See Sect. VIII for proof.

From $(*)$, the probability under the drop action $A_t = 1$ is given by the (Q_t, Q_{t+1})-th element of the kernel whereas the $(Q_t, \max(Q_t + 1, L))$-th gives the required probability under the admit action $A_t = 0$. Finally, similar to the single flow case, we define the reward function $R(S_t, A_t)$ as

$$R(S_t, 0) = -M\mathbb{1}_{\{(Q_t+1)/\mu > \eta\}} + \left(\sqrt{\beta_j + 1} - \sqrt{\beta_j}\right),$$
$$R(S_t, 1) = -M\mathbb{1}_{\{Q_t/\mu > \eta\}} + \left(\sqrt{\beta_j/2} - \sqrt{\beta_j}\right) .$$

(14)

To derive the optimal policy, we can now simulate the trajectory of the system according to $(*)$ and use a function approximator, e.g., DQN [38] to learn the Q-values for each state-action pair. Subsequently, the optimal policy is given by the action with higher Q-value for each state. The advantage of our approach over model-free learning is that it requires a lot less data to reliably predict the optimal action.

VI. EVALUATION

In this section, we evaluate PAQMAN for TCP Reno traffic under different network scenarios using simulations. We first

$$G_j(a,b) = \mathbb{1}_{j \in \{i_1 \ldots i_l\}} \prod_{1 \le k \le l} e^{\left((c_{i_k} - c_{i_{k-1}})H_{i_k}\right)} \int_a^b e^{\left(\sum_{1 \le k \le l, i_k \ne j} -\beta_{i_k}(y - c_{i_k})\right)} e^{(y - c_{i_l})H_{i_{l+1}}} \beta_j e^{-\beta_j(y - c_j)} dy$$

$$= \mathbb{1}_{j \in \{i_1 \ldots i_l\}} \beta_j e^{\left(\sum_{1 \le k \le l} \beta_{i_k} c_{i_k}\right)} \prod_{1 \le k \le l} e^{(c_{i_k} - c_{i_{k-1}})H_{i_k} - c_{i_l} H_{i_{l+1}}} \int_a^b e^{\left(\sum_{1 \le k \le l} -\beta_{i_k} y\right)} e^{y H_{i_{l+1}}} dy \qquad (*)$$

$$= \mathbb{1}_{j \in \{i_1 \ldots i_l\}} \beta_j e^{\left(\sum_{1 \le k \le l} \beta_{i_k} c_{i_k}\right)} \prod_{1 \le k \le l} e^{(c_{i_k} - c_{i_{k-1}})H_{i_k} - c_{i_l} H_{i_{l+1}}} \left(\sum_{1 \le k \le l} \beta_{i_k} I - H_{i_{l+1}}\right)^{-1}$$

$$\left(e^{a\left(\sum_{1 \le k \le l} \beta_{i_k} I - H_{i_{l+1}}\right)} - e^{b\left(\sum_{1 \le k \le l} \beta_{i_k} I - H_{i_{l+1}}\right)}\right) ,$$

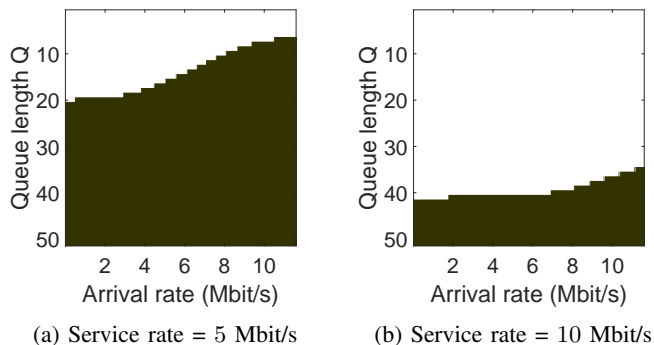(a) Service rate = 5 Mbit/s   (b) Service rate = 10 Mbit/s

Fig. 5: PAQMAN under negligible RTT for two different service rates: the plot considers arrival rates $\in [0.01, 12]$ Mbit/s and the buffer size is set to 50 packets. The target delay threshold $\eta = 50$ ms corresponds to 20 packets in (a) and to 40 packets in (b). The darker region shows where an incoming packet should be dropped.
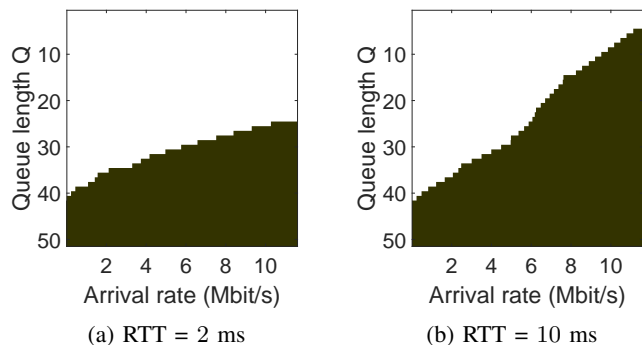

(a) RTT = 2 ms   (b) RTT = 10 ms

Fig. 6: PAQMAN with known RTT: the left subplot assumes an RTT of 2 ms and the right corresponds to a higher RTT of 10 ms. Like Fig. 6b, we take arrival rate $\in [0.01, 12]$ Mbit/s, service rate = 10 Mbit/s, and a buffer size of 50 packets and show the packet drop area in dark. As PAQMAN admits all packets arriving between two decision events which is at least one RTT long, it compensates by dropping packets in (b) earlier than (a). In contrast to queue length based AQM heuristics these figures show that an estimate of the arrival rate is crucial to optimize the AQM performance.

focus on the case where the switch deals with a single flow having negligible RTT. The evaluation under non-negligible RTT is subsequently taken up in Sect. VI-B. Throughout our simulations, PAQMAN aims to achieve a delay shorter than the target delay $\eta = 50$ ms while optimizing the throughput. Recall from (1) and (11) that the reward function combines individual delay and throughput objectives. The penalty amount in the reward function for breaching this delay is fixed at $M = 10^6$. To make comparisons fair, we set the target delay parameter of CoDel to the same target delay, wherever applicable. All other internal Codel parameters are left unchanged.

While comparing the performance of different policies, we look at the empirical stationary behaviour of the corresponding system. Here, we consider the behavior of long-lived flows. To derive the stationary behaviour, we conduct 200 simulation runs. Each run spans across 5×10^4 packet arrivals to simulate the stationary behaviour and the run starts with an arrival rate equal to the link service rate. To generate comparison plots, we subsequently time-average the congestion indicators or the state of the system, given by the buffer filling of the switch and the flow arrival rate. Note that the stationary arrival rate equals the stationary throughput and the queue length determines the *expected* delay. We use these immediate connotations to interpret the findings from our plots. To show the transient behaviour of the respective AQM algorithm in these plots, time propagation is indicated by varying the colour of the state from blue to yellow.

A. Negligible RTT

In this section, we analyze PAQMAN for the case of a single flow having negligible RTT. We follow the method described in Sect. III to compute the packet drop policy.

We first plot the derived policies using the inputs from (3) for two different average service rates. The simulation setup for the first scenario corresponds to a service rate $\mu = 5$ Mbit/s, while the second is given by $\mu = 10$ Mbit/s. Further, we calculate the policy for arrival rates in the range $[0.01, 12]$ Mbit/s, simulated using a Gamma distributed packet interarrival times with fixed shape parameter $\alpha = 1.5$. For our simulations, we only change the rate parameter β of the Gamma random variable to reflect the impact of the drop/admit action on the effective arrival rate β/α. The resulting policies are shown in Fig. 5, where an incoming packet should be dropped if the current state of the switch, i.e., *buffer filling and the packet arrival rate belong to the dark region*. As expected, a lower service rate at the switch entails more aggressive packet drops reflected by the difference in Fig. 5a and 5b. The non-trivial effect of flow arrival rate on the policy is also noteworthy.

Next, we compare PAQMAN to CoDel and droptail queues. As mentioned earlier, the target delay parameter of CoDel is set to the delay threshold for PAQMAN. The evaluation is shown in Fig. 7 where the policy from Fig. 5b is used to generate the left subplot. We plot the system state, given by the *(i)* current buffer filling and *(ii)* flow arrival rate, time-averaged over multiple runs. Our plots suggest that, in stationarity, PAQMAN results in an arrival rate that is comparable to CoDel, although the queue length appears to be much shorter. This immediately translates to the fact that PAQMAN yields equivalent stationary throughput while keeping the delay much shorter. As already known, the droptail policy in Fig. 7c generates near-perfect utilization at the cost of a longer delays.

B. Non-negligible RTT

Next, we consider the case when the flow RTT is non-negligible. We assume throughout this subsection that the RTT is estimated by (*or* known to) the switch. We follow the method from Sect. V to derive PAQMAN under each setting
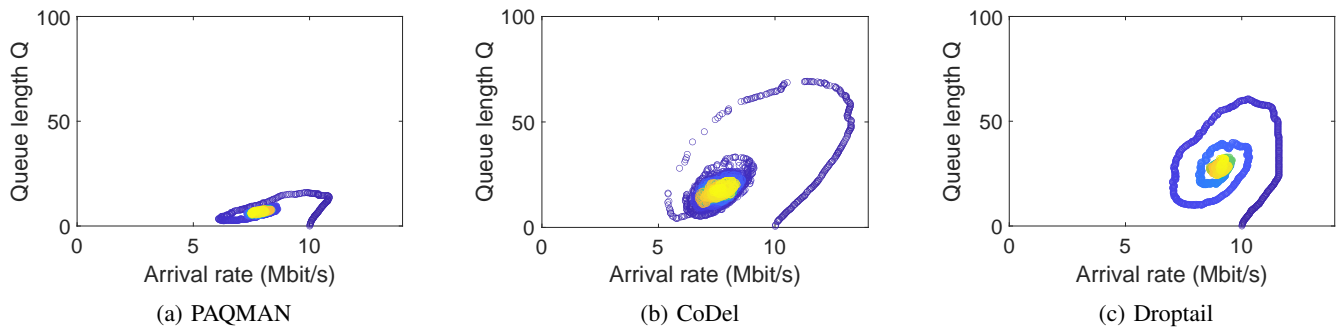
(a) PAQMAN

(b) CoDel

(c) Droptail

Fig. 7: State evolution under different policies for a switch port with service rate = 10 Mbit/s. As already known, the droptail queue achieves higher stationary utilization than the AQMs at the cost of higher delay. PAQMAN leads to much shorter delay than CoDel, while achieving comparable stationary throughput and faster convergence to steady-state.
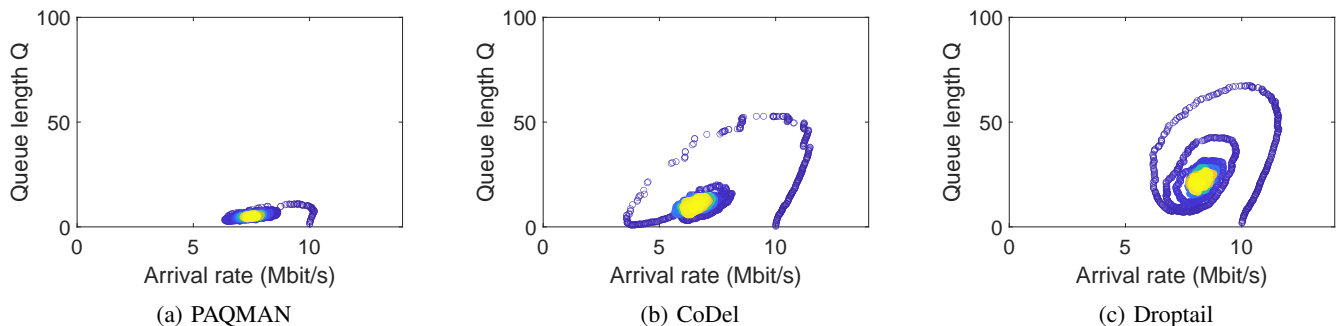

(a) PAQMAN

(b) CoDel

(c) Droptail

Fig. 8: State evolution under different policies for a flow with known RTT = 2 ms and switch service rate = 10 Mbit/s. Similar to Fig. 7, PAQMAN converges faster to the steady-state characterized by shorter delay and equivalent throughput.
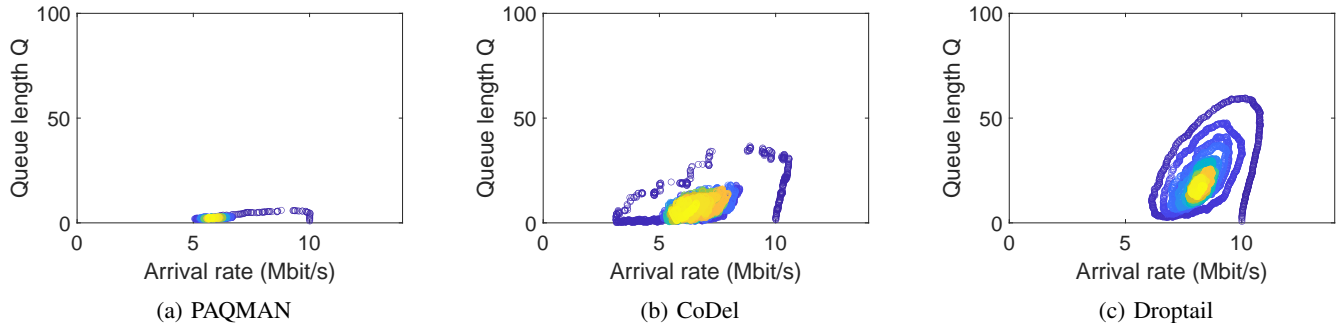

(a) PAQMAN

(b) CoDel

(c) Droptail

Fig. 9: State evolution under different policies for a setup same as Fig. 8 except that the flow RTT is much higher (10 ms). We observe similar phenomena as Fig. 8 although both the stationary delay and the throughput decrease across AQM policies.

and accordingly simulate the system. As before, we compare the resulting performance with CoDel and droptail queues.

In Fig. 6, we illustrate PAQMAN under low and high RTTs. We fix the packet service rate at 10 Mbit/s and restrict our attention to arrival rate $\in [0.01, 12]$ Mbit/s. Recall that in this case, PAQMAN admitted all arrivals between two decision epochs subject to sufficient room at the buffer of the switch. This leads to increased packet buffering for higher RTTs, especially in high arrival regimes as the inter-decision times are RTT-based. To alleviate this problem, the policy is more aggressive and starts dropping earlier as seen via a comparison

of Fig. 6a and 6b.

In Fig. 8 and 9, we compare PAQMAN with CoDel and Droptail when there is only one flow passing through the buffered switch port. The state evolution under PAQMAN (left subplot) in these figures is generated using the corresponding policy from Fig. 6, which follows the workflow described in Sect. V-A. Similar to Fig. 7, we see that PAQMAN achieves shorter delay than CoDel while yielding comparable stationary throughput. Further, a quick comparison of Fig. 7-9 reveals that a longer flow RTT leads to a reduction in both throughput and delay, which we investigate further in Fig. 11. Here, we
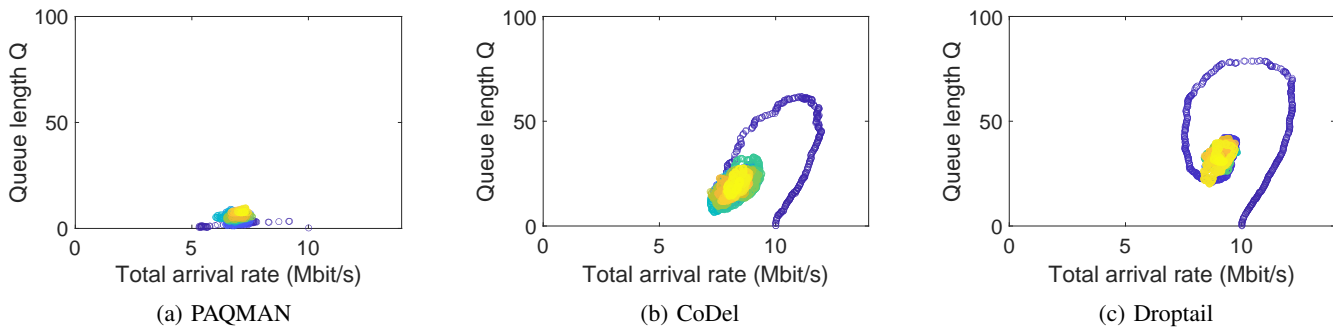
(a) PAQMAN   (b) CoDel   (c) Droptail

Fig. 10: Evolution of queue length and the total (aggregate) arrival rate for 3 flows under different policies for service rate = 10 Mbit/s with target delay = 50 ms and RTT = = $[2, 4, 6]$ ms.

first derive PAQMAN and subsequently simulate the system for long-lived flows (5×10^4 packet arrivals) for every given RTT. The box plots per RTT in Fig. 11 are generated using 200 simulation runs. Consistent with Fig. 7-9, we see that the stationary delay and the stationary throughput tend to diminish as the RTT increases.

Finally, in Fig. 10, we compare PAQMAN's performance in the multi-flow case. Here, we consider three flows with RTT $[2, 4, 6]$ ms and the service is assumed to be $\mu = 10$ Mbit/s. We start the system in a phase where the total arrival rate equals the switch port service rate and the individual arrival rates are equal, i.e., each flow has equal share of the bandwidth. Similar to the experiments in the single flow case, we simulate the system for 200 runs, where each run spans across 5×10^4 packet arrivals. To learn the policy, we pre-simulate the system and use a DQN to approximate the Q-values for each system state, which in turn decides the optimal action. As with any offline-learned algorithm, the efficacy can be verified only under the assumption that underlying conditions do not vary much from the training sample which we ensure during the simulation. The time-averaged plot of the aggregated system state expressed in terms of queue length and total arrival rate is shown in Fig. 10. We see that that PAQMAN achieves similar throughput to CoDel, while keeping the delay considerably low here as well.

VII. CONCLUSION

In this paper we presented PAQMAN, a principled approach to AQM with corresponding algorithmic design. In contrast to many AQM heuristics, we take a direct approach to incorporate the packet arrival rate in our formulation of the state of the system and optimize with respect to a linear combination of delay and throughput objectives. Our work takes a probabilistic approach to combine a model of congestion control (mainly Additive Increase- Multiplicative Decrease) with AQM packet drops to formulate the AQM problem as finding optimal packet dropping policy in a Semi-Markov Decision Process, given a target delay parameter. Simulation results show that the direct approach of incorporating the arrival rate in the state leads to comparable throughput of the system to the widely used AQM policy CoDel, while outperforming it in terms of latency.
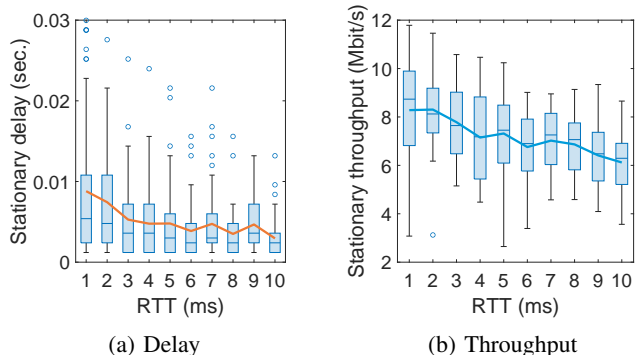

(a) Delay   (b) Throughput

Fig. 11: Impact of flow RTT on the stationary delay and throughput. Given an RTT, we simulate the system for long-lived flows using PAQMAN. Each box plot is generated using 200 simulation runs. Consistent with previous plots, both delay and throughput decrease with an increasing RTT.

VIII. APPENDIX

Proof of Lemma III.1. : Let us first denote the upper incomplete gamma integral by $\gamma(\cdot, \cdot)$, which satisfies the following recursion:

$$\gamma(x + 1, y) = \int_y^\infty t^x e^{-t} dt = x\gamma(x, y) + y^x e^{-y},$$

for $\Re(x) > 0$. We see that

$$P(Y_{u,v} > X_{w,z})$$
$$= \int_0^\infty P(Y_{u,v} > X_{w,z}|X_{w,z} = t)dP_X(t)$$
$$= \int_0^\infty P(Y_{u,v} > t)dP_X(t), \ Y_{u,v} \perp\!\!\!\perp X_{w,z}$$
$$= \int_0^\infty \frac{\gamma(u, vt)}{\Gamma(u)} \frac{z^w t^{w-1} e^{-zt}}{\Gamma(w)} dt$$
$$= \int_0^\infty \frac{(u-1)\gamma(u-1, vt) + (vt)^{u-1}e^{-vt}}{\Gamma(u)} \frac{z^w t^{w-1} e^{-zt}}{\Gamma(w)} dt$$
$$= P(Y_{u-1,v} > X_{w,z}) + \int_0^\infty \frac{v^{u-1}z^w t^{u+w-2}e^{-(v+z)t}}{\Gamma(u)\Gamma(w)} dt$$
$$= P(Y_{u-1,v} > X_{w,z}) + \frac{\Gamma(u+w-1)v^{u-1}z^w}{\Gamma(u)\Gamma(w)(v+z)^{u+w-1}}.$$
$$\int_0^\infty \frac{(v+z)^{u+w-1}t^{u+w-2}e^{-(v+z)t}}{\Gamma(u+w-1)} dt$$
$$= P(Y_{u-1,v} > X_{w,z}) +$$
$$\frac{\Gamma(u+w-1)}{\Gamma(u)\Gamma(w)} \left(\frac{v}{v+z}\right)^{u-1} \left(\frac{z}{v+z}\right)^w,$$

which proves the first part of the lemma. By induction on u,

$$P(Y_{u,v} > X_{w,z}) = P(Y_{1,v} > X_{w,z}) +$$
$$\sum_{k=2}^u \frac{\Gamma(k+w-1)}{\Gamma(k)\Gamma(w)} \left(\frac{v}{v+z}\right)^{k-1} \left(\frac{z}{v+z}\right)^w. \quad (15)$$

Since $Y_{1,v}$ is exponentially distributed with parameter v,

$$P(Y_{1,v} > X_{w,z}) = \int_0^\infty e^{-vt} \frac{z^w t^{w-1} e^{-zt}}{\Gamma(w)} dt$$
$$= \left(\frac{z}{v+z}\right)^w \int_0^\infty \frac{(v+z)^w t^{w-1} e^{-(v+z)t}}{\Gamma(w)} dt$$
$$= \left(\frac{z}{v+z}\right)^w.$$

Hence from (15),

$$P(Y_{u,v} > X_{w,z}) = \sum_{k=0}^{u-1} \frac{\Gamma(k+w)}{\Gamma(k+1)\Gamma(w)} \left(\frac{v}{v+z}\right)^k \left(\frac{z}{v+z}\right)^w.$$

$\square$

Proof of Prop. III.2. : We start by observing that $P((q,\beta_{t+1})|S_t, 0)$ denotes the probability of the event that exactly $Q_t + 1 - q$ packets are served between two packet arrivals for $1 \leq q \leq Q_t + 1$. Now, at least n packets are served between two arrivals if and only if the corresponding service time $Y_{n,\mu}$ is less than the respective interarrival time $X_{\alpha,\beta_{t+1}}$. The change in the arrival rate parameter is given by the fact that following an action on the t-th packet arrival, the arrival rate parameter is instantly changed to $\beta_{t+1} = \beta_t + \alpha$

which *determines the distribution of the next interarrival time*. Thus, given $\beta_{t+1} = \beta_t + \alpha$, for $1 \leq q \leq Q_t + 1$,

$$P((q,\beta_{t+1})|S_t, 0)$$
$$= P(Y_{Q_t+2-q,\mu} > X_{\alpha,\beta_{t+1}}) - P(Y_{Q_t+1-q,\mu} > X_{\alpha,\beta_{t+1}})$$
$$= \frac{\Gamma(Q_t+1-q+\alpha)}{\Gamma(Q_t+2-q)\Gamma(\alpha)} \left(\frac{\mu}{\mu+\beta_{t+1}}\right)^{Q_t+1-q} \left(\frac{\beta_{t+1}}{\mu+\beta_{t+1}}\right)^\alpha.$$

For $q = 0$, we observe that no more than $Q_t + 1$ packets can be served between two arrivals. Hence, given $\beta_{t+1} = \beta_t + \alpha$,

$$P((0,\beta_{t+1})|S_t, 0)$$
$$= P(Y_{Q_t+1,\mu} \leq X_{\alpha,\beta_{t+1}})$$
$$= 1 - P(Y_{Q_t+1,\mu} > X_{\alpha,\beta_{t+1}})$$
$$= 1 - \sum_{k=0}^{Q_t} \frac{\Gamma(k+\alpha)}{\Gamma(k+1)\Gamma(\alpha)} \left(\frac{\mu}{\mu+\beta_{t+1}}\right)^{Q_t+1-k} \left(\frac{\beta_{t+1}}{\mu+\beta_{t+1}}\right)^\alpha,$$

which completes the proof. $\square$

Proof of Prop. V.2. The queue length grows like the population of a birth-death process in the interval $(T_i, T_i + r]$ with intensity matrix G_i. Thus the transition matrix for this interval is given by e^{rG_i}. Further $T_{i+1} - (T_i + r) \sim \text{Exp}(\beta_i)$ implying

$$P_i = e^{rG_i} \int_0^\infty e^{tG} \beta_i e^{-\beta_i t} dt, = \beta_i e^{rG_i} \int_0^\infty e^{-(\beta_i I - G)t} dt.$$

We have used the facts that $e^{-c}I = e^{-cI}$ and cI commutes with any matrix of similar dimension $\forall c \in \mathbb{R}$. Now,

$$\beta_i I - G = \begin{bmatrix} \beta_i & 0 & 0 & \dots & \dots \\ -\mu & \mu+\beta_i & 0 & 0 & \dots \\ 0 & -\mu & \mu+\beta_i & 0 & \dots \\ \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & \dots & -\mu & \mu+\beta_i \end{bmatrix}.$$

Therefore, by Gershgorin's theorem [39], all eigenvalues of $\beta_i I - G$ lie in the following closed balls: $B(\beta_i, 0), B(\mu + \beta_i, \mu)$. Hence, by lemma V.1

$$P_i = \beta_i e^{G_i r} (\beta_i I - G)^{-1}.$$

$\square$

Proof of $(*)$. : Observe that

$$P\left(Y_t \in (a,b], Y_t = Y_{jt}, V_t = Q_{t+1} - Q_t\right)$$
$$= \int_a^b P\left(Y_t = y, V_t = Q_{t+1} - Q_t|Y_{jt} = y\right)dF_{Y_{jt}}(y)$$
$$= \mathbb{1}_{j\in\{i_1\dots i_l\}} \int_a^b P\left(\bigcap_{k\leq l, i_k\neq j} Y_{i_k t} > y\right)$$
$$P(V_t = Q_{t+1} - Q_t|Y_t = y)dF_{Y_{jt}}(y) \quad (16)$$
$$= \mathbb{1}_{j\in\{i_1\dots i_l\}} \int_a^b \prod_{k\leq j, i_k\neq j} P(Y_{i_k t} > y)$$
$$P(V_t = Q_{t+1} - Q_t|Y_t = y)dF_{Y_{jt}}(y).$$

Here, the indicator function $\mathbb{1}_{j\in\{i_1\dots i_l\}}$ indicates the fact that in order to have the flow index of the packet involving the next decision epoch as j, $c_j \leq c_{i_l}$ should hold. Further, the
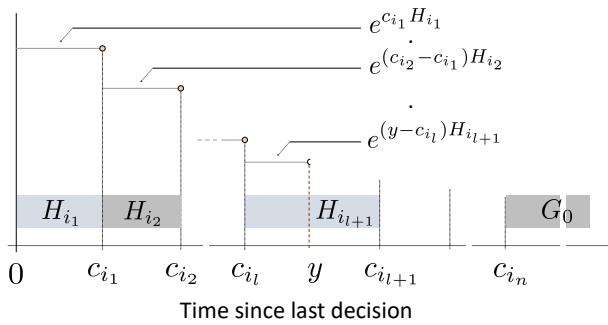
Fig. 12: Intensity and corresponding transition matrices (TMs) between two decision epochs: for the time since last decision epoch, the intensity matrices (H) are highlighted in grey for successive intervals. Corresponding TMs over a (sub) interval of length x is given by e^{xH}. The TM for an interval consisting of contiguous subintervals is the product of respective transition matrices as shown for $(0, y)$.

first term in the integrand corresponds to the fact that given $Y_{jt} = y$, the events $\{Y_t = Y_{jt}\}$ and $\{Y_{i_k t} > y \; \forall k \leq l, i_k \neq j\}$ are equivalent. Now, to evaluate the expression $P(V_t = Q_{t+1} - Q_t | Y_t = y)$ in (16), we notice that the queue length of the underlying process grows like the population of a birth-death process where the death rate is always μ subject to a positive queue length. The birth rate, however, depends upon the time interval. Observe that there cannot be any packet arrival from i_k-th flow once the time since the last decision epoch exceeds c_{i_k}. Thus, the $(L+1) \times (L+1)$ intensity matrix in the interval $(c_{i_{k-1}}, c_{i_k}], 1 \leq k \leq n$ is given by

$$
H_{i_k} = \begin{bmatrix}
-b_k & b_k & 0 & \dots & \dots \\
\mu & -\mu - b_k & b_k & 0 & \dots \\
0 & \mu & -\mu - b_k & b_k & \dots \\
\dots & \dots & \dots & \dots & \dots \\
0 & \dots & \dots & \mu & -\mu
\end{bmatrix},
$$

with the sum of flow rates $b_k = \sum_{m \geq k} \beta_{i_m}$ and $c_{i_0} = 0$ by definition. Further, the intensity matrix in the interval (c_{i_n}, ∞) is given by G from (9). The different intensity matrices for the arrivals in the different time spans after the last AQM decision is illustrated in Fig. 12.

Replacing $P(V_t = Q_{t+1} - Q_t | Y_t = y)$ in (16) by the corresponding transition matrix and denoting the respective transition kernel for $P(Y_t \in (a, b], Y_t = Y_{jt}, V_t = Q_{t+1} - Q_t)$ as $G_j(a, b)$, we obtain the formulation $(*)$. $\qquad \square$

REFERENCES

[1] S. Iyer, R. Zhang, and N. McKeown, "Routers with a single stage of buffering," in *Proceedings of the 2002 conference on Applications, technologies, architectures, and protocols for computer communications*, 2002, pp. 251–264.

[2] J. Gettys, "Bufferbloat: Dark buffers in the internet," *IEEE Internet Computing*, vol. 15, no. 3, pp. 96–96, 2011.

[3] D. Lee, B. E. Carpenter, and N. Brownlee, "Observations of udp to tcp ratio and port numbers," in *2010 Fifth International Conference on Internet Monitoring and Protection*, 2010, pp. 99–104.

[4] D.-M. Chiu and R. Jain, "Analysis of the increase and decrease algorithms for congestion avoidance in computer networks," *Computer Networks and ISDN systems*, vol. 17, no. 1, pp. 1–14, 1989.

[5] S. Floyd, R. Gummadi, S. Shenker *et al.*, "Adaptive red: An algorithm for increasing the robustness of red's active queue management," 2001.

[6] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Transactions on networking*, vol. 1, no. 4, pp. 397–413, 1993.

[7] K. Nichols and V. Jacobson, "Controlling queue delay," *Communications of the ACM*, vol. 55, no. 7, pp. 42–50, 2012.

[8] R. Pan, P. Natarajan, C. Piglione, M. S. Prabhu, V. Subramanian, F. Baker, and B. VerSteeg, "Pie: A lightweight control scheme to address the bufferbloat problem," in *2013 IEEE 14th International Conference on High Performance Switching and Routing (HPSR)*, 2013, pp. 148–155.

[9] R. Adams, "Active queue management: A survey," *IEEE Communications Surveys Tutorials*, vol. 15, no. 3, pp. 1425–1476, 2013.

[10] N. Khademi, D. Ros, and M. Welzl, "The new aqm kids on the block: An experimental evaluation of codel and pie," in *2014 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2014, pp. 85–90.

[11] R. Kundel, A. Rizk, J. Blendin, B. Koldehofe, R. Hark, and R. Steinmetz, "P4-codel: Experiences on programmable data plane hardware," in *ICC 2021 - IEEE International Conference on Communications*, 2021, pp. 1–6.

[12] H. Jiang and C. Dovrolis, "Passive estimation of tcp round-trip times," *ACM SIGCOMM Computer Communication Review*, vol. 32, no. 3, pp. 75–88, 2002.

[13] B. Veal, K. Li, and D. Lowenthal, "New methods for passive estimation of tcp round-trip times," in *International workshop on passive and active network measurement*. Springer, 2005, pp. 121–134.

[14] B. A. Arouche Nunes, K. Veenstra, W. Ballenthin, S. Lukin, and K. Obraczka, "A machine learning framework for tcp round-trip time estimation," *EURASIP Journal on Wireless Communications and Networking*, vol. 2014, no. 1, pp. 1–22, 2014.

[15] D. Lapsley and S. Low, "Random early marking for internet congestion control," in *Seamless Interconnection for Universal Services. Global Telecommunications Conference. GLOBECOM'99.(Cat. No. 99CH37042)*, vol. 3. IEEE, 1999, pp. 1747–1752.

[16] T. J. Ott, T. Lakshman, and L. H. Wong, "Sred: stabilized red," in *IEEE INFOCOM'99. Conference on Computer Communications. Proceedings. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. The Future is Now (Cat. No. 99CH36320)*, vol. 3. IEEE, 1999, pp. 1346–1355.

[17] C. V. Hollot, V. Misra, D. Towsley, and W.-B. Gong, "A control theoretic analysis of red," in *Proceedings IEEE INFOCOM 2001. Conference on Computer Communications. Twentieth Annual Joint Conference of the IEEE Computer and Communications Society (Cat. No. 01CH37213)*, vol. 3. IEEE, 2001, pp. 1510–1519.

[18] K. B. Kim, A. Tang, and S. H. Low, "Design of aqm in supporting tcp based on the well-known aimd model," in *GLOBECOM'03. IEEE Global Telecommunications Conference (IEEE Cat. No. 03CH37489)*, vol. 6. IEEE, 2003, pp. 3226–3230.

[19] F. Yanfie, R. Fengyuan, and L. Chuang, "Design a pid controller for active queue management," in *Proceedings of the Eighth IEEE Symposium on Computers and Communications. ISCC 2003.* IEEE, 2003, pp. 985–990.

[20] Z. Liu, J. Sun, S. Hu, and X. Hu, "An adaptive aqm algorithm based on a novel information compression model," *IEEE Access*, vol. 6, pp. 31 180–31 190, 2018.

[21] T. Alpcan, P. G. Mehta, T. Basar, and U. Vaidya, "Control of non-equilibrium dynamics in communication networks," in *Proceedings of the 45th IEEE Conference on Decision and Control.* IEEE, 2006, pp. 5216–5221.

[22] T. Alpcan, P. Wang, P. G. Mehta, and T. Başar, "A non-equilibrium analysis and control framework for active queue management," *Automatica*, vol. 44, no. 10, pp. 2474–2486, 2008.

[23] Y. Wang, L. Wang, and X. Dong, "An intelligent tcp congestion control method based on deep q network," *Future Internet*, vol. 13, no. 10, p. 261, 2021.

[24] G. Wu, E. K. Chong, and R. Givan, "Burst-level congestion control using hindsight optimization," *IEEE Transactions on Automatic Control*, vol. 47, no. 6, pp. 979–991, 2002.

[25] O. Habachi, H.-P. Shiang, M. Van Der Schaar, and Y. Hayel, "Online learning based congestion control for adaptive multimedia transmission," *IEEE transactions on signal processing*, vol. 61, no. 6, pp. 1460–1469, 2013.

[26] P. Jacko and B. Sansò, "Optimal anticipative congestion control of flows with time-varying input stream," *Performance Evaluation*, vol. 69, no. 2, pp. 86–101, 2012.

[27] A. Bitorika, M. Robin, and M. Huggard, "An evaluation framework for active queue management schemes," in *11th IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer Telecommunications Systems, 2003. MASCOTS 2003*. IEEE, 2003, pp. 200–206.

[28] C. Hollot, Y. Liu, V. Misra, and D. Towsley, "Unresponsive flows and aqm performance," in *IEEE INFOCOM 2003. Twenty-second Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE Cat. No. 03CH37428)*, vol. 1. IEEE, 2003, pp. 85–95.

[29] D. Lin and R. Morris, "Dynamics of random early detection," in *Proceedings of the ACM SIGCOMM'97 conference on Applications, technologies, architectures, and protocols for computer communication*, 1997, pp. 127–137.

[30] C. A. Gomez, X. Wang, and A. Shami, "Intelligent active queue management using explicit congestion notification," in *2019 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2019, pp. 1–6.

[31] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[32] I. Järvinen and M. Kojo, "Evaluating codel, pie, and hred aqm techniques with load transients," in *39th Annual IEEE Conference on Local Computer Networks*. IEEE, 2014, pp. 159–167.

[33] H. C. Tijms, *A first course in stochastic models*. John Wiley and sons, 2003.

[34] K. Nichols, V. Jacobson, and A. McGregor, "and j. iyengar, ed.," controlled delay active queue management," RFC 8289, DOI 10.17487/RFC8289, January 2018,< https://www. rfc-editor. org . . . , Tech. Rep.

[35] A. Feldbaum, "Dual control theory. i," *Avtomatika i Telemekhanika*, vol. 21, no. 9, pp. 1240–1249, 1960.

[36] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. The MIT Press, 2018. [Online]. Available: http://incompleteideas.net/book/the-book-2nd.html

[37] S. Ha, I. Rhee, and L. Xu, "Cubic: a new tcp-friendly high-speed tcp variant," *ACM SIGOPS operating systems review*, vol. 42, no. 5, pp. 64–74, 2008.

[38] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[39] S. A. Gershgorin, "Über die Abgrenzung der Eigenwerte einer Matrix," *Bull. Acad. Sci. URSS*, vol. 1931, no. 6, pp. 749–754, 1931.