Action-based Scheduling: Leveraging App Interactivity for Scheduler Efficiency

John Tadrous[1], Atilla Eryilmaz[2], and Ashutosh Sabharwal[3]

[1]Department of Electrical and Computer Engineering, Gonzaga University, WA. E-mail: tadrous@gonzaga.edu
[2]Department of Electrical and Computer Engineering, The Ohio State University, OH. E-mail: eryilmaz.2@osu.edu
[3]Electrical and Computer Engineering Department, Rice University, TX. E-mail: ashu@rice.edu

Abstract—**The dominant portion of smartphone traffic is generated by apps that involve human interactivity. Particularly, when human users receive information from a server, they spend a few seconds of information processing before taking an action. The user processing time creates an idle communication period during the app session. Moreover, the generation of future traffic depends on the service of the current query-response pair. In this work, we aim at leveraging the properties of such interactions to reap quality-of-experience (QoE) gains.**

Existing schedulers, both in practice and theory, are not designed in view of the aforementioned traffic characteristics. Theoretical works predominantly focus on scheduling of traffic that is either generated independently or directly controlled, but not governed by the the specific dynamics caused by human interactions. Schedulers in practice, on the other hand, employ round-robin and processor-sharing methods to serve multiple ongoing sessions. We show that neither of these approaches is effective for serving apps that involve human interactivity. Instead, we show that optimal scheduling for interactive traffic is non-randomized over packets, which we call action-based, as it avoids breaking ongoing service of actions in order to align human response times with the service of other actions. Since the design of optimal action-based policy is computationally prohibitive, we develop low-complexity suboptimal action-based policies that are optimal for two ongoing sessions. Our numerical studies based on a real-data trace reveal that our proposed action-based policies can reduce total delay by 22% with respect to packet-based equal processor sharing.

Index Terms—**Interactive apps, scheduling, non-convex optimization.**

I. Introduction

Recent analysis of wireless network traffic characteristics [1] has revealed that web-related apps are responsible for more than 34% of the aggregate wireless data volume, ranking as the second dominant source of traffic after video streaming. Web-browsing, as well as several other app categories, such as online gaming, news, social networking, and travel reservations, essentially require significant user interactions with the app promising of human behavioral impact on the data traffic generated by such apps in the relevant timescale of seconds. These apps are thus different from streaming and file download apps that do not involve human interaction throughout the whole session of app usage.

With the growing interest in improving end-users' QoE in the next generation networks, human behavior is now being included in resource allocation frameworks [2], [3], [4], [5]. A multitude of emerging schemes harness long timescale predictability of human demand, location, and economic responsiveness in design. Such prediction-based design is manifested in the emerging paradigms of proactive content caching [6], [7], [8], [9], [10], WiFi off-loading [11], [12], [13], [14], and time and content dependent pricing [15], [16], [17], [18]. Nevertheless, there has been no specific work that captures human's influence on the data traffic in the short timescale of seconds. In a recent work [19], traffic patterns of smartphone apps have been studied from a human behavioral perspective in the short time-scale of seconds. In a Yelp session, for instance, a search query is sent (forming an action initiated by an end-user) which results in a response (from the server's end). Based on the response, the user may initiate one of many specific actions, e.g., another search, or ask for a particular detail. Each action causes a burst of packets being exchanged between the user and server. While each action encompasses a user-server communication in the form of query-response pair, time gaps between actions have been reported to span several seconds. These gaps occur as the user processes data response from the server before generating the subsequent action. Fig. 1 illustrates a traffic pattern from a Yelp session. The figure highlights the sequence of actions constituting the exchange of information between the two communicating parties. We can also observe the few-seconds time gaps between consecutive actions.

In this work, we focus on developing app-aware scheduling policies that leverage the characteristics of the action-based model [19] for maximal QoE. In particular, wireless schedulers can provision their resources so that the service of an action for one user coincides with the thinking time of another, hence effectively minimizing contentions for the available resources and associated operational delays. Another key feature of app interactions pronounced in the timescale of seconds is that *the generation of a new action is contingent upon the service of the current one.* That is, each user needs to receive the server's response first before she can process it and initiate a consequent query. As such,

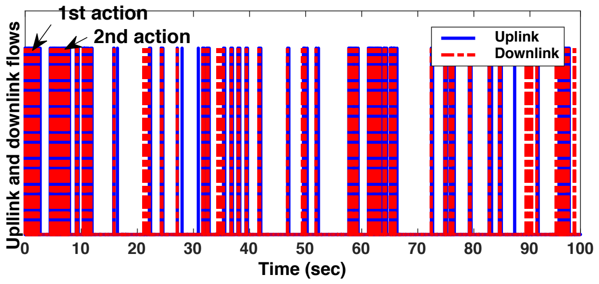


Fig. 1: Uplink and downlink packet flows of a Yelp session as scheduled by a WiFi access point. Every instant of packet transmission is represented by a line: a solid blue line for uplink, and a dashed red line for the downlink. The flows constitute a sequence of actions, each consisting of a query and a response. Actions are separated by time gaps determined by human processing time.

arrivals of new traffic are heavily dependent on the service strategy employed.

In this work, we consider a wireless scheduler (e.g., a basestation (BS) or an access point (AP)) deployed at the network edge that serves multiple ongoing sessions involving human interactions with the objective of minimizing the total delay for all sessions.

The proposed framework of *action-based* scheduling requires the infrastructure schedulers to become app-aware, which is a significant departure from current systems. While we appreciate the difficulty in this change, we are motivated by developing solutions that can possibly improve the efficiency of current systems by increasing system complexity, and without requiring any new wireless resources. In fact, there has been recently a considerable progress in fingerprinting smartphone apps at the network outlets [20], [21].

It is also worth noting that practical WiFi standards, e.g., the enhanced distributed channel access (EDCA) function of the 802.11e, offer differentiated service based on the traffic type for enhanced quality of service (QoS). These traffic types are called access categories [22], [23], [24]. In particular, voice traffic has the highest service privilege followed by video traffic, then best-effort and background traffic. We can thus see that interactive apps like Yelp, Web-browsing, online-gaming, etc., fall all under the best-effort traffic category of the EDCA function, despite each of these apps has its intrinsic user-server interaction characteristics and can be segmented for better QoE. In addition, the current WiFi schedulers (including 802.11e) employ randomized service policies that are based on random back-off times which do not ensure *action-based* scheduling, a necessary condition for minimum operational delay as proven in this work.

We list our contributions in this work as follows:

- We model and formulate the problem of scheduling data packets from multiple independent sessions sharing a common wireless scheduler for the minimization of the total delay of all sessions. The model accounts for the seconds-long time gap following the service of each action, as well as the fact that a new action from the same session can only be generated after the current one has been served.
- We show that optimal service policy is a non-randomized policy, or what we call *action-based*. That is, given any set of actions pending service at the scheduler, the optimal service decision is to serve all packets from one action, rather than switching service between packets from different actions.
- As the optimal solution to the problem is computationally intractable, we develop three low-complexity suboptimal action-based policies, namely a successive approximation policy (SAP), a greedy policy, and a two-session comparison policy (TSCP). The SAP and greedy policy are proved to outperform any packet-based (non-action-based) scheduler. The TSCP policy outperforms baseline packet-based schedulers that employ first-come-first-serve or equal processor sharing policies.
- We provide real-data driven numerical simulations to demonstrate efficiency of proposed algorithms. We show the average delay reduction of action-based scheduling compared to equal processor sharing is 22% when running apps are randomly picked from Google Maps, Google Chrome, Yelp, Expedia, and GT Racing 2, an online car racing game. The key element leading to higher delay reduction gain of action-based scheduling is the diversity in the level of interactivity. Higher gains are witnessed when some running apps have longer user thinking time with faster processing of actions at the scheduler (e.g., Expedia) while others have shorter user thinking times with slower processing of actions (e.g, Google Maps). For instance, when Expedia and Google Maps are simulated, delay reduction gain is 43%.

While there has been existing literature on scheduling of data traffic with interactive nature [25], [26], [27], [28], [29], [30], none of the earlier works has considered the aforementioned characteristics of app interactions, namely *human thinking time and coupling between service and future arrivals.* In [25], [26] traffic with human interaction is treated as a high-priority class in mutli-type traffic systems for its delay sensitivity, yet no other specific interactive characteristics have been captured. In [27], interactivity at the transport-layer TCP timescale has been considered for opportunistic routing. The only captured property of TCP interactive nature has been the signaling overhead. In [28], authors model interactions of users with web-browsing apps through an ON-OFF process that resembles the action-based model adopted in this paper. The ON-time dynamics are captured as an $M/M/1/K$ queue whereas the OFF-time dynamics are taken to be an $M/G/\infty$ queue, service times have then been assumed exponentially distributed in [29]. The fundamental difference between [28], [29] and our work is that they assume independent action arrivals (ON-times) of the service offered to each action, while we account for the coupling between future action generation and the service of the current action. In [30] interactive traffic has been modeled as a sequence of jobs with deadlines, yet future job arrivals are independent of the service offered.

The rest of this paper is organized as follows. In Section II, we layout the system model and formulate the problem of total time average expected delay minimization and show that the optimal solution is attained by a non-randomized, or what we call action-based, policy. In Section III, we investigate the optimal service policy design and show the merits of action-based scheduling leading to global and locally optimal solutions. We present our proposed policy designs in Section IV, and provide numerical simulations in Section V. We conclude the work in Section VI.

II. System Model and Problem Formulation

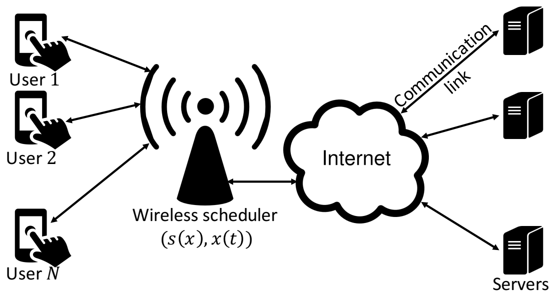


Fig. 2: System Model

A. System Model

We consider a wireless scheduler deployed at the network edge (a BS or an AP) that serves a set $\mathcal{N} := \{1, \cdots, N\}$ of N independent sessions involving human interactions. The scheduler is assumed to be aware of the interaction characteristics for each session. Time is divided into slots with the slot duration being the time taken by the scheduler to serve one packet in both uplink or downlink directions. Traffic packets are exchanged between end-users and associated servers. When the scheduler decides to serve a packet from a certain session, then in one slot, the packet is received from the source and forwarded to the destination. Fig. 2 shows the main system components. We assume that the wireless scheduler is the bottleneck element in the network that can at most serve one packet per time slot.

App interactivity: We adopt an action-based model to capture the short timescale dynamics of the data traffic. In particular, we assume that session n is represented by a random sequence of actions $\{a_n(t)\}_{t\geq 0}$, where $a_n(t) \in \{0, 1\}$ denotes the session state at time t, whether in action or not. Suppose that only session n is running, and no other sessions are competing for resources with session n. Then, if a query or a response packet has been exchanged between the user and the server in slot t, then $a_n(t) = 1$, in which case the system is witnessing an action for session n, and the user-server pair will complete the action in the next slot with probability μ_n, transiting to state $a_n(t+1) = 0$. On the

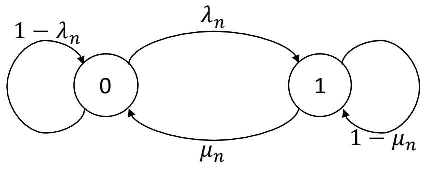


Fig. 3: Packet generation process for a system with a single session n.

other hand, if the query and response exchange has been completed in slot t, then $a_n(t) = 0$, in which case there is no action for session n at time t, and the user-server pair will initiate an action in the next slot with probability λ_n, transiting to state $a_n(t+1) = 1$. When session n has all its packets served upon generation without scheduling delays, i.e., the only session in the network, then process $\{a_n(t)\}_t$ follows the discrete time Markov process illustrated in Fig. 3. The assumed Markov behavior of the action process can be justified by the recent findings on the interactive-app traffic. It has been shown that action interarrivals and service durations can be reasonably modeled with light-tailed distributions [19]. In fact, Google Maps app has an exponentially distributed action interrarival distribution. Here we note that parameter λ_n is related to user thinking time: the larger λ_n is, the faster the user interacts with the app, and vice versa. Moreover, parameter μ_n quantifies the processing time of actions of session n: the larger μ_n is, the faster the scheduler can serve actions of session n.

Service of N concurrent sessions: The wireless scheduler can serve one packet at a time slot. We let $Q_n(t) \in \{0, 1\}$ represent the queue backlog of session n at slot t. If session n is in an action state and has a pending service at time t, then $Q_n(t) = 1$, else $Q_n(t) = 0$. The system state at time t, denoted $x(t)$, is then given by the set of all actions pending service in slot t. That is, $x(t) := \{n : Q_n(t) = 1, n \in \mathcal{N}\}$, and the system's state space is the power set $\mathcal{X} := 2^{\mathcal{N}}$. The service of packets from the N sessions is dictated by the scheduler through a service strategy $\pi := \{s(t, x(t))\}_{t\geq 0}$, where $s(t, x(t)) \in x(t)$ is the session scheduled to communicate a packet in slot t.

Service delay: We consider the operational delay to be the main QoE metric in this work. Since the scheduler can serve one packet every time slot, each session $n \in x(t)$ incurs a time-slot of delay if $s(t, x(t)) \neq n$. Formally, the cumulative delay of session n by time t is given by

$$d_n(t, \pi) = \sum_{l=0}^{t-1} \mathbf{1}_{\{s(l,x(l))\neq n\}} Q_n(l), \quad n \in \mathcal{N}, \qquad (1)$$

where $\mathbf{1}_{\{c\}}$ is the indicator function that takes on the value 1 if condition c is met, and 0 otherwise.

Coupling between service and new arrivals: Due to the interactive nature of apps, we assume that actions are generated sequentially in the sense that no new action is generated before all the packets of the current action have been served. As such, the queue backlog of each session is related to the action process through the following update equation:

$$Q_n(t+1) = a_n(t - d_n(t, \pi)), \forall n. \tag{2}$$

In the following subsection we formulate the optimal scheduling problem and introduce the baseline equal processor sharing policy.

B. Problem Formulation

We consider the problem of minimizing the time average aggregate expected delay of the N sessions. The time average delay for session n under service policy π is given as

$$D_n(\pi) = \limsup_{t\to\infty} \frac{1}{t}\mathbb{E}[d_n(t)]. \tag{3}$$

The aggregate time average delay under policy π is thus written as $D_{\text{tot}}(\pi) := \sum_{n=1}^{N} D_n(\pi)$, and the delay minimization problem is:

$$D^*_{\text{tot}} = \min_{\pi\in\tilde{\Pi}} D_{\text{tot}}(\pi), \tag{4}$$

where $\tilde{\Pi} := \{\pi = \{s(t, x(t))\}_t : s(t, x(t)) = 0 \text{ if } x(t) = \phi,\ s(t, x(t)) \in x(t), x(t) \neq \phi\}$ is the set of all feasible strategies.

Optimality of Action-based Scheduling: With the aggregate delay minimization problem formulated, we can conclude the following.

Proposition 1: The optimal solution to (4) is realized by a *stationary* non-randomized policy $\pi^* = \{s^*(x(t))\}_t$.

Proof. From (1), the queue update (2) can be re-written as

$$Q_n(t+1) = \begin{cases} 0 & \text{w.p. } \mu_n, & s(t,x(t)) = n, Q_n(t) = 1, \\ 1 & \text{w.p. } 1-\mu_n, & s(t,x(t)) = n, Q_n(t) = 1, \\ 1, & & s(t,x(t)) \neq n, Q_n(t) = 1, \\ 0 & \text{w.p. } 1-\lambda_n, & Q_n(t) = 0, \\ 1 & \text{w.p. } \lambda_n, & Q_n(t) = 0. \end{cases}$$

Thus, the future system state $x(t+1)$ depends only on the state and control of time t. As such, the process $\{x(t)\}_t$ is a controlled finite-state Markov chain with a finite decision space. Hence, it is optimized by a stationary non-randomized policy [31]. ■

Proposition 1 has thus established that the optimal service policy does not randomize decisions over the packets of contending sessions. Instead, the optimal decision is to serve all the packets belonging to an action from one session before switching to the packets of another session. This is what we refer to as *action-based* scheduling in order to highlight the significance of leveraging app interactivity characteristics to improve the QoE gains of the network (with the formal definition of action-based scheduling is provided in Definition 2 in Subsection III-B). However, to investigate the structure of the delay minimization problem, its solution, and the operation of other randomized policies, e.g., EPS, we consider the system's performance under a general stationary randomized policy.

Randomized Scheduling: Consider a general stationary randomized policy π, at any time slot t the control $s(t, x(t))$ depends only on the state $x(t)$ and not on the time index. That is, $s(t, x(t)) = s(x(t)), \forall t$. Accordingly, for any state x, the stationary control $s(x) = n$ w.p. $\alpha_n(x)$, $n \in x$, where $\alpha_n(x) \in [0, 1]$, and $\sum_{n\in x} \alpha_n(x) = 1$. Note that $(\alpha_n(x))_n$ defines a probability mass function over all sessions of state $x \in \mathcal{X}$. The value of $\alpha_n(x)$ determines the fraction of the scheduler resources assigned to session $n \in x$ whenever the system is in state x. In the sequel, we confine the solution space to the set of all feasible stationary policies, $\Pi \subset \tilde{\Pi}$, $\Pi := \{\pi = \{s(t, x(t))\}_t : s(t, x(t)) = s(x(t)), \forall t\}$.

Definition 1: We call the policy that assigns equal shares of the scheduler resources to all the sessions of a state x an equal processor sharing (EPS) policy. We denote an EPS policy by $\hat{\pi} = \{\hat{s}(x)\}_{x\in\mathcal{X}}$ with $\hat{\alpha}_n(x) = \frac{1}{|x|}$, and $|x|$ is the cardinality of the set x.

Baseline Service: We adopt the EPS policy as the baseline service strategy employed by traditional wireless schedulers. The reason behind such a choice is that current schedulers do not distinguish between different running apps' traffic, and hence treat all sessions equally irrespective of their app interaction characteristics. The EPS policy attains the same average delay performance as round-robin and first-come-first-serve (FCFS) policies that serve packets in the order of their arrival. When two or more sessions are backlogged, i.e., having packets pending service, their packets actually request service from the AP scheduler in the same slot. The FCFS policy will serve these packets in a round-robin fashion. That is, after serving one packet from a session, a new packet from the same session will request service in the next slot. Such packet will not be scheduled until one packet from every other session has been served. Thus FCFS exhibits a round-robin behavior in this case. We let $\hat{D}_{\text{tot}} := D_{\text{tot}}(\hat{\pi})$ denote the aggregate expected delay under EPS policy.

In the rest of this paper, we investigate the design of the optimal stationary policy that harnesses the user-server interaction characteristics. Since in [19] it has been reported that action interarrival and service times occur in the timescale of a few seconds while packet service time occurs in the timescale of microseconds, we assume that the time slot duration allows for at most one event of either one action arrival or one action service. That is, $\lambda_n \ll 1$, $\mu_n \ll 1$ rendering the Markov chain of the system operate in a nearly continuous time fashion. The previous assumption, however, does not have any impact on the structure of the problem pertaining to the optimality results below; it is only adopted to simplify the analysis.

III. Optimal Policy Design

In this section, we investigate the structure of the delay minimization problem, (4) study its complexity, and show that all local optima are attained by action-based scheduling.

A. The Two-session System

We begin with the $N=2$, two-session, system to exactly characterize the optimal solution and glean main insights on the scheduling problem. The state space of the system with two users is given by $\mathcal{X}=\{\phi,\{1\},\{2\},\{1,2\}\}$. The system has only one state with more than one session having concurrent actions pending service, that is $\{1,2\}$. Any stationary policy $\pi\in\Pi$, therefore, schedules $s(\{1\})=1$, $s(\{2\})=2$, and $s(\{1,2\})=1$, w.p. $\alpha_1(\{1,2\})$ and $s(\{1,2\})=2$, w.p. $\alpha_2(\{1,2\})$. Since $(\alpha_n(\{1,2\}))_{n=1}^2$ is a probability mass function, we can write $\alpha_1(\{1,2\})=\alpha$ and $\alpha_2(\{1,2\})=1-\alpha$, $\alpha\in[0,1]$.

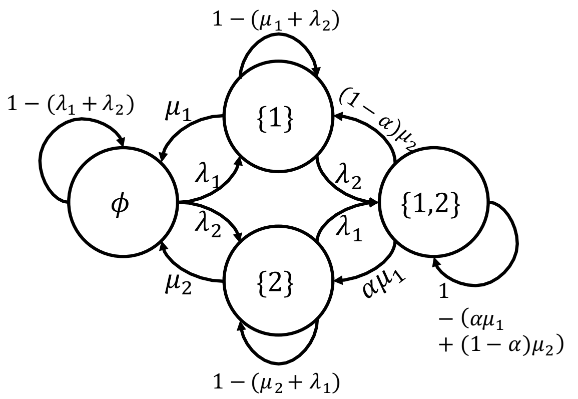


Fig. 4: State-diagram of the two-session system under a randomized stationary policy π.

The resulting system state process $\{x(t)\}_t$ is a discrete time Markov chain with a state evolution diagram depicted in Fig. 4.

Proposition 2: For $N=2$, the delay minimizing strategy π^* is given by

$$\alpha^*=\begin{cases}1, & \frac{\mu_1(\lambda_2^2+\lambda_1\lambda_2+\lambda_2\mu_1+\mu_1\mu_2)}{\mu_2(\lambda_1^2+\lambda_1\lambda_2+\lambda_1\mu_2+\mu_1\mu_2)}>1,\\ 0, & \text{otherwise,}\end{cases} \tag{5}$$

and the minimum average delay is given by

$$D_{\text{tot}}^*=\frac{\lambda_1\lambda_2(\lambda_1+\lambda_2+\mu_1+\mu_2)}{(\lambda_{n^*}+\mu_{n^*})(\mu_1\mu_2+\lambda_{3-n^*}(\lambda_1+\lambda_2+\mu_1+\mu_2))},$$

where $n^*=1$ if $\alpha^*=1$ and $n^*=2$ if $\alpha^*=0$.

Proof. Let $P(\{1,2\})$ be the steady-state distribution of state $\{1,2\}$ under policy π. We have $\mathbb{E}[\mathbf{1}_{\{s(t)\neq 1\}}Q_1(t)]=(1-\alpha)P(\{1,2\})$. Since $\mathbb{E}[d_1(t)]=\sum_{l=0}^{t-1}\mathbb{E}[\mathbf{1}_{\{s(l)\neq 1\}}Q_1(l)]$, then $D_1=(1-\alpha)P(\{1,2\})$. In the same way, we get $D_2=\alpha P(\{1,2\})$. Thus the total time average delay is given by $D_{\text{tot}}=P(\{1,2\})$, i.e., the fraction of time spent in state $\{1,2\}$ is always a delay for one of the two sessions.

Solving for the steady-state distribution of the Markov chain, we obtain $P(\{1,2\})=\frac{\nu(\alpha)}{\delta(\alpha)}$, where

$$\begin{aligned}\nu(\alpha)=&\lambda_1\lambda_2(\lambda_1+\lambda_2+\mu_1+\mu_2),\\ \delta(\alpha)=&\nu(\alpha)+\alpha\mu_1(\lambda_2+\mu_2)(\lambda_1+\lambda_2+\mu_1)+\\ &(1-\alpha)\mu_2(\lambda_1+\mu_1)(\lambda_1+\lambda_2+\mu_2).\end{aligned}$$

We observe that ν is independent of α while $\delta(\alpha)$ is linear in it. Thus, D_{tot} is monotone in α. Checking the first derivative of D_{tot} w.r.t. α, $\frac{\partial D_{\text{tot}}}{\partial\alpha}=\frac{\delta(\alpha)\nu'(\alpha)-\nu(\alpha)\delta'(\alpha)}{\delta(\alpha)^2}$. Since $\delta(\alpha)^2$ is non-negative, then D_{tot} is monotonically decreasing in α if and only if $\delta(\alpha)\nu'(\alpha)<\nu(\alpha)\delta'(\alpha)$ and non-decreasing otherwise. Through simple algebraic manipulation, it follows that D_{tot} is decreasing in α if and only if $\frac{\mu_1(\lambda_2^2+\lambda_1\lambda_2+\lambda_2\mu_1+\mu_1\mu_2)}{\mu_2(\lambda_1^2+\lambda_1\lambda_2+\lambda_1\mu_2+\mu_1\mu_2)}>1$, in which case the optimal choice of α is $\alpha^*=1$. Else, $\alpha^*=0$. ■

We note from Proposition 2 that the optimal scheduling procedure always favors actions of session n^*. Whenever the system is in state $\{1,2\}$, the scheduler gives all resources to session n^* until all packets of its ongoing action have been served. Then the system transits to state $\{3-n^*\}$ where resources are assigned to the action of session $\{3-n^*\}$. The system then either completes the service of session $\{3-n^*\}$'s action and becomes empty (returns to state ϕ), or a new action from session n^* is initiated in which case the system goes to state $\{1,2\}$ and assigns its resources to session n^*. The optimal policy is action-based because the scheduler decides to serve a whole action in every state.

The condition for selecting the session to serve in state $\{1,2\}$ is dependent on user thinking and action processing times for each session. We can see from (5) that having large thinking time (i.e., small λ_n) and small action processing time (i.e., large μ_n) are favorable properties towards serving actions of session n first whenever more than one action are awaiting service. In fact, if $\lambda_1\leq\lambda_2$ and $\mu_1>\mu_2$ then $\alpha^*=1$. The rationale is that, larger thinking times leave more system resources available for serving actions from the other session while actions with smaller processing times create less delay when served first.

If the two sessions are scheduled under our baseline EPS policy, then $\hat{\alpha}=0.5$, and the total expected delay is $\hat{D}_{\text{tot}}=\frac{2\lambda_1\lambda_2}{\lambda_1\lambda_2+(\lambda_1+\mu_1)(\lambda_2+\mu_2)}$. We measure the relative efficiency of optimal action-based policy π^* over EPS $\hat{\pi}$ as $\eta^*=\frac{\hat{D}_{\text{tot}}-D_{\text{tot}}^*}{\hat{D}_{\text{tot}}}\times 100\%$, which quantifies the delay reduction gain. By substitution with system parameters $(\lambda_n,\mu_n)_n$, we get η^* in (6) where n^* is as defined in Proposition 2.

Remark 1: Comparing (6) with the allocation of α^* in (5), we have $\eta^*\geq 0$ with equality if and only if $\lambda_1=\lambda_2$ and $\mu_1=\mu_2$, in which case the system is indifferent to both sessions, and D_{tot} is independent of α.

Remark 2: Since state $\{1,2\}$ is recurrent, contention for scheduler resources between the two sessions can never be avoided. For each contention event, i.e., being in state $\{1,2\}$, the optimal action-based policy reduces delay by 50% relative to the EPS, however, resolving contentions faster allows for more contentions than under the EPS policy, thus the maximum delay reduction gain is 50%.

B. The N-session Scenario

In this subsection, we study the optimality of action-based scheduling for the N-session system. Under any randomized stationary policy π, the system follows a discrete-time

$$\eta^* = \frac{\mu_{n^*}(\lambda_{3-n^*}^2 + \lambda_1\lambda_2 + \lambda_{3-n^*}\mu_{n^*} + \mu_1\mu_2) - \mu_{3-n^*}(\lambda_{n^*}^2 + \lambda_1\lambda_2 + \lambda_{n^*}\mu_{3-n^*} + \mu_1\mu_2)}{(\lambda_{n^*} + \mu_{n^*})(\lambda_{3-n^*}^2 + \lambda_1\lambda_2 + \mu_1\mu_2 + \lambda_{3-n^*}(\mu_1 + \mu_2))} \times 50\%. \tag{6}$$

Markov process with state space $\mathcal{X}$ of 2^N different states. The Markov process is also aperiodic and irreducible, hence has a limiting distribution $\mathbf{p} := (P(x))_{x\in\mathcal{X}}$, that is given by the unique solution to the linear system

$$\mathbf{A}\mathbf{p} = \mathbf{b},$$

where $\mathbf{A}$ is the $2^N \times 2^N$ state transition matrix with its $2^{N^{th}}$ row containing all ones, and $\mathbf{b}$ is a $2^N \times 1$ vector with all elements but the last one are zeros, and the last element is one. For the construction of $\mathbf{A}$ and $\mathbf{b}$, we introduce the mapping $J : \mathcal{X} \to \mathbb{Z}_{++}$ where $J(x) = \sum_{n\in x} 2^{n-1} + 1$ is a one-to-one indexing of each state $x \in \mathcal{X}$. We denote the element in $\mathbf{A}$ whose row index is $J(x)$ and column index is $J(y)$ by $\mathbf{A}(x, y)$, for any $x, y \in \mathcal{X}$. The elements of $\mathbf{A}$ are thus given as:

$$\mathbf{A}(x,y) = \begin{cases} -\sum_{n\notin x}\lambda_n - \sum_{n\in x}\alpha_n(x)\mu_n, & x = y, x \neq \mathcal{N}, \\ 1, & x = \mathcal{N}, y \in \mathcal{X}, \\ \alpha_n(y)\mu_n, & y\backslash x = \{n\}, \forall n \notin x, \\ \lambda_n, & x\backslash y = \{n\}, \forall n \notin y, x \neq \mathcal{N}, \\ 0, & \text{otherwise.} \end{cases}$$

Similarly, we denote the element in $\mathbf{b}$ whose index is $J(x)$ with $\mathbf{b}(x)$, where $\mathbf{b}(\mathcal{N}) = 1$, and $\mathbf{b}(x) = 0$, $\forall x \in \mathcal{X}\backslash\{\mathcal{N}\}$.

Proposition 3: Let $\mathbf{B}_x$ denote the matrix formed by replacing the column $J(x)$ in $\mathbf{A}$ with $\mathbf{b}$, and define $\mathcal{D} := \{x \in \mathcal{X} : |x| > 1\}$ as the set of all delay states. Then

$$D_{\text{tot}}(\pi) = \frac{1}{\det(\mathbf{A})}\sum_{x\in\mathcal{D}}(|x|-1)\det(\mathbf{B}_x). \tag{7}$$

Proof. For any session $n \in \mathcal{N}$, $\mathbb{E}[\mathbf{1}_{\{s(t)\neq n\}}Q_n(t)] = \sum_{x\in\mathcal{D},n\in x}(1-\alpha_n(x))P(x)$. Hence

$$\begin{aligned} D_{\text{tot}}(\pi) &= \sum_{n=1}^{N} D_n(\pi) \\ &= \sum_{n=1}^{N}\sum_{x\in\mathcal{D},n\in x}(1-\alpha_n(x))P(x). \end{aligned}$$

Rearranging terms, we obtain

$$D_{\text{tot}} = \sum_{x\in\mathcal{D}}(|x|-1)P(x).$$

By applying Cramer's rule with $P(x) = \frac{\det(\mathbf{B}_x)}{\det(\mathbf{A})}$, we obtain (7). ∎

Expression (7) captures the impact of optimization parameters $(\alpha_n(x))_n$, $x \in \mathcal{X}$, on the aggregate expected delay of the system. As such, we can re-write the problem (4) as

$$\begin{aligned} D_{\text{tot}}^* = \min_{(\alpha_n(x))_n, x\in\mathcal{X}} \quad & \frac{1}{\det(\mathbf{A})}\sum_{x\in\mathcal{D}}(|x|-1)\det(\mathbf{B}_x) \\ \text{s.t.,} \quad & \alpha_n(x) \in [0,1], \quad \forall n \in x, x \in \mathcal{X}, \\ & \sum_{n\in x}\alpha_n(x) = 1, \quad \forall x \in \mathcal{X}. \end{aligned} \tag{8}$$

While the new optimization (8) has a convex constraint set, its objective function is non-convex in the optimization parameters because of the non-linearity of the determinant operator. Therefore, obtaining an optimal allocation of the scheduler resources is a computationally formidable problem. Nevertheless, we harness the structure of the objective function to establish that all locally optimal solutions to the problem are non-randomized (action-based).

Definition 2 (Action-based policy): We say that a policy $\pi \in \Pi$ is action-based if it is a non-randomized policy. That is, if for every state $x \in \mathcal{D}$, $\exists n \in x$ such that $\alpha_n(x) = 1$. In other words, an action-based policy is such that for every delay state, the scheduler allocates all resources to serve the action of a particular session.

On the action level, an action-based policy is in general preemptive. That is, if $\alpha_n(x) = 1$ for some session n in state x, then $\alpha_n(x \cup \{m\})$ is not necessarily 1, for any $m \notin x$.

Theorem 1 (Optimality of action-based scheduling): Any opitmal solution to (8) is attained by an action-based policy.

Proof. Notation: For matrix $\mathbf{A}$ (or respectively $\mathbf{B}_z$), we use $\mathbf{A}\langle x, y\rangle$ ($\mathbf{B}_z\langle x, y\rangle$) to denote the $2^N - 1 \times 2^N - 1$ matrix resulting from eliminating the row and column corresponding to states x, y respectively from $\mathbf{A}$ ($\mathbf{B}_z$). We also use $\text{sgn}(x, y)$ to denote the sign of either $\mathbf{A}(x, y)$ or $\mathbf{B}_z(x, y)$. With this notation, we can expand $\det(\mathbf{A})$ with respect to $(\alpha_n(x))_n$ as

$$\begin{aligned} \det(\mathbf{A}) = \sum_{n\in x}\alpha_n(x)\mu_n\,\text{sgn}(x\backslash\{n\}, x)\det(\mathbf{A}\langle x\backslash\{n\}, x\rangle) \\ + \text{sgn}(\mathcal{N}, x)\det(\mathbf{A}\langle\mathcal{N}, x\rangle) + R(x), \end{aligned} \tag{9}$$

where

$$R(x) := \begin{cases} -\left(\sum_{n\in x}\alpha_n(x)\mu_n + \sum_{m\notin x}\lambda_m\right)\det(\mathbf{A}\langle x, x\rangle) \\ +\,\text{sgn}(\mathcal{N}, x)\det(\mathbf{A}\langle\mathcal{N}, x\rangle), \quad x = \mathcal{N}\backslash\{n\}, \forall n. \\ \det(\mathbf{A}\langle\mathcal{N}, \mathcal{N}\rangle), \quad x = \mathcal{N}. \\ \sum_{m\notin x}\lambda_m\,\text{sgn}(x\cup\{m\}, x)\det(\mathbf{A}\langle x\cup\{m\}, x\rangle) \\ +\,\text{sgn}(\mathcal{N}, x)\det(\mathbf{A}\langle\mathcal{N}, x\rangle) \\ -\left(\sum_{n\in x}\alpha_n(x)\mu_n + \sum_{m\notin x}\lambda_m\right)\det(\mathbf{A}\langle x, x\rangle), \\ \hfill \text{otherwise.} \end{cases} \tag{10}$$

In the same way, we can expand $\det(\mathbf{B}_y)$ with respect to $(\alpha_n(x))_n$ where

$$\det(\mathbf{B}_y) = \begin{cases} \text{Replace every occurrence of } \mathbf{A} \text{ in} & \\ \text{(9), (10) with } \mathbf{B}_y, & y \neq x. \\ \operatorname{sgn}(\mathcal{N}, x)\det(\mathbf{B}_x\langle \mathcal{N}, x\rangle), & y = x. \end{cases} \tag{11}$$

Lemma 1: Let $K \in \mathbb{Z}_{++}$ be a positive integer, $\mathbf{c}, \mathbf{g} \in \mathbb{R}^K$ be real $K-$dimensional vectors, and $e, h \in \mathbb{R}$ be real numbers such that $\mathbf{g}^T\mathbf{v} + h \neq 0$, for any probability mass function $\mathbf{v} = (v_k)_{k=1}^K$. Then any optimal solution to

$$\begin{aligned} \min_{\mathbf{v}\in[0,1]^K} & \frac{\mathbf{c}^T\mathbf{v} + e}{\mathbf{g}^T\mathbf{v} + h} \\ \text{s.t.} & \sum_{k=1}^{K} v_k = 1, \end{aligned} \tag{12}$$

denoted $\mathbf{v}^* = (v_k^*)_k$ is such that $v_k^* \in \{0, 1\}$.

Proof Lemma. Taking the first derivative of the objective function w.r.t. v_k, we obtain

$$\frac{\sum_{l=1,l\neq k}^{K} v_l(g_l c_k - g_k c_l) + hc_k - eg_k}{(\mathbf{g}^T\mathbf{v} + h)^2},$$

where c_l, g_l are the l^{th} elements of $\mathbf{c}, \mathbf{g}$ respectively.

Since the numerator is independent of v_k, the objective function is monotone in every coordinate. Since the denominator is independent of k, the index w.r.t. which we compute the first derivative, then the optimal allocation of $\mathbf{v}$ satisfies

$$v_k^* = \begin{cases} 1, & k = \arg\min_l \frac{c_l + e}{g_l + h}, \\ 0, & \text{otherwise.} \end{cases}$$

■

Now let $(\alpha_n^*(z))_{n\in z}$ denote an optimal allocation when the system is in state $z \in \mathcal{D}$. Suppose that in every state $z \in \mathcal{D}\backslash\{x\}$ we use $(\alpha_n^*(z))_{n\in z}$, and we use a general allocation $(\alpha_n(x))_{n\in x}$ with state $x \in \mathcal{D}$. From the expansions of $\det(\mathbf{A})$, $\det(\mathbf{B}_y)$ w.r.t. the controls $(\alpha_n(x))_{n\in x}$ (9), (11), we observe that these expansions are affine in $(\alpha_n(x))_n$. As such, delay minimization problem can be written in the form of (12), with the optimization variables being $(\alpha_n(x))_n$. Hence, that optimal allocation satisfies $\alpha_n^*(x) = 1$ for some $n \in x$, and $\alpha_m^*(x) = 0$, $\forall m \in x\backslash\{n\}$. ■

Theorem 1 thus establishes that any randomized policy is always outperformed by at least one action-based policy. Thus, harnessing the query-response nature of app-user interactions in the form of action-based services holds the potential to maximally enhance end-users QoE. In addition, the theorem significantly reduces the complexity of the optimization from an infinite dimensionality problem in which the search for optimal solution runs over an uncountable set of values to a search over a set of finitely many points. Nevertheless, the search for optimal strategy does still suffer major complexity as the number of sessions grows. This growing complexity calls for the design of efficient action-based schedulers that strike a balance between minimal delay performance and operational complexity.

IV. Proposed Service Policies

While Theorem 1 reduces the problem's complexity to a finite search, the global optimal solution has to be obtained through an exhaustive search problem with a worst case complexity of $\prod_{n=2}^{N} n\binom{N}{n}$, which is still formidable for large N. Accordingly, we study the design of low-complexity service policies to reap the potential gains of interactive traffic characteristics. We consider two main approaches. The first utilizes successive approximations of the non-convex problem, whereas the second utilizes priority-based scheduling.

A. Successive Approximations Policy

The complexity of policy design essentially emanates from the non-convexity of the objective function of (8). We tackle such a problem through an iterative technique that solves a sequence of approximate convex problems of the original problem. In particular, in each iteration i, the non-convex objective function, $D_{\text{tot}}(\pi)$, is replaced with an approximate quasiconvex function $\tilde{D}_{\text{tot}}^{(i)}(\pi)$, then we use the solution to the resulting convex problem to generate a new approximate function $\tilde{D}_{\text{tot}}^{(i+1)}(\pi)$, and so on. The series of solutions to these approximate convex problems converges to a point that satisfies the KKT conditions of (8).

Lemma 2: Let $\tilde{D}_{\text{tot}}^{(i)}$ be a convex function in $(\alpha_n(x))_n$, $x \in \mathcal{D}$, that replaces the objective function $\tilde{D}_{\text{tot}}$ in (8) at iteration i. Denote by $\pi^{(i-1)} := (\alpha_n^{(i-1)}(x))_n$, $x \in \mathcal{D}$, the optimal solution to the resulting convex optimization problem at the $i - 1^{st}$ iteration, $i = 1, 2, \cdots$. If

1) $\tilde{D}_{\text{tot}}^{(i)}(\pi) \geq D_{\text{tot}}(\pi)$, $\forall \pi \in \Pi$,
2) $\nabla\tilde{D}_{\text{tot}}^{(i)}(\pi^{(i-1)}) = \nabla D_{\text{tot}}(\pi^{(i-1)})$,
3) $\tilde{D}_{\text{tot}}^{(i)}(\pi^{(i-1)}) = D_{\text{tot}}(\pi^{(i-1)})$,

then $D_{\text{tot}}(\pi^{(i)}) \leq D_{\text{tot}}(\pi^{(i-1)})$, $\forall i$, and the sequence of policies $\{\pi^{(i)}\}_i$ converges to a policy $\acute{\pi} := (\acute{\alpha}_n(x))_{n\in x, x\in\mathcal{D}}$ which is a locally optimal solution to (8).

Lemma 2 is a special case of Theorem 1 in [32] which aims to provide locally optimal solutions to non-convex problems.

Then we construct the approximate objective function $\tilde{D}_{\text{tot}}^{(i)}(\pi)$ as follows.

$$\tilde{D}_{\text{tot}}^{(i)} := \begin{cases} \frac{1}{\det(\mathbf{A}^{(i)})}\sum_{x\in\mathcal{D}}(|x| - 1)\det(\mathbf{B}_x^{(i)}), & i = 1, 2, \cdots, \\ \hat{D}_{\text{tot}}, & i = 0, \end{cases} \tag{13}$$

where

$$\mathbf{A}^{(i)}(x, y) := \begin{cases} \mathbf{A}(x, y), \quad \forall x \in \mathcal{X}, i \mod 2^N + 1 = J(y). \\ \mathbf{A}^{(i-1)}(x, y), \quad \forall x \in \mathcal{X} \\ \qquad\qquad , i \mod 2^N + 1 \neq J(y). \\ \hat{\mathbf{A}}, \quad i = 0. \end{cases} \tag{14}$$

Moreover, $\mathbf{B}_z^{(i)}(x, y)$ is obtained as in (14) by replacing every occurrence of $\mathbf{A}$ and $\mathbf{A}^{(i)}$ with $\mathbf{B}_z$ and $\mathbf{B}_z^{(i)}$, respectively, $\forall z \in \mathcal{X}$. The matrices $\hat{\mathbf{A}}$, $\hat{\mathbf{B}}_z$, $z \in \mathcal{X}$ are obtained for a scheduler that employs our baseline EPS policy. In particular, they respectively consist of elements of $\mathbf{A}$ and

$\mathbf{B}_z$ with every $\alpha_n(x)$ replaced with $\hat{\alpha}_n(x)$, $\forall n \in x$, $x \in \mathcal{X}$, $z \in \mathcal{X}$.

The idea is to have iterations run over all states sequentially, where in each iteration resource sharing controls are optimized for one state assuming other states employ most recent controls from the previous iterations. Since there are 2^N possible states, a state $x \in \mathcal{X}$ is visited once every 2^N iterations, whenever $i \mod 2^N + 1 = J(x)$.

Theorem 2: Let $x \in \mathcal{X}$ be such that $J(x) = i \mod 2^N + 1$, for some iteration i. The approximate objective function $\tilde{D}_{\text{tot}}^{(i)}$ of (13) is quasiconvex in $(\alpha_n(x))_n$. Further, the sequence of solutions resulting from replacing D_{tot} of (8) with $\{\tilde{D}_{\text{tot}}^{(i)}\}_i$ converges to a locally optimal solution of (8).

Proof. First, for $x \in \mathcal{X}$ satisfies $i \mod 2^N + 1 = J(x)$, we note that $\tilde{D}_{\text{tot}}^{(i)}$ is a linear fractional mapping in $(\alpha_n(x))_n$ since $\det(\mathbf{A}^{(i)}) \neq 0$ for any randomized stationary policy. While $\tilde{D}_{\text{tot}}^{(i)}$ is quasilinear, the resulting optimization problem at iteration i, i.e.,

$$\begin{aligned} \min_{(\alpha_n(x))_n} \quad & \tilde{D}_{\text{tot}}^{(i)}(\pi) \\ \text{s.t.,} \quad & \alpha_n(x) \in [0,1], \quad \forall n \in x, \\ & \sum_{n \in x} \alpha_n(x) = 1 \end{aligned} \tag{15}$$

can be transformed into a convex (linear) optimization problem through means of change of variables (c.f. Chapter 4 in [33]).

Second, we consider the three conditions specified in Lemma 2. Since D_{tot} is continuous in $(\alpha_n(x))_{n,x}$ and the set of constraints is compact, then D_{tot} has a maximum value U that can be added to $\tilde{D}_{\text{tot}}^{(i)}$ to maintain Condition 1). Nevertheless, adding a constant to the objective function does not change the solution, which is the main point of interest. For Condition 2), we have

$$\begin{aligned} &\partial \frac{\tilde{D}_{\text{tot}}^{(i)}}{\alpha_n(x)}\Bigg|_{\left(\alpha_n^{(i-1)}(x)\right)_n} \\ &= \sum_{y \in \mathcal{D}} \frac{|y|-1}{\det(\mathbf{A}^{(i)})^2}\left(\det(\mathbf{A}^{(i)})\frac{\partial \det(\mathbf{B}_y^{(i)})}{\partial \alpha_n(x)}\right. \\ &\quad \left.\left. - \det(\mathbf{B}_y^{(i)})\frac{\partial \det(\mathbf{A}^{(i)})}{\partial \alpha_n(x)}\right)\right|_{\left(\alpha_n^{(i-1)}(x)\right)_n} \\ &= \sum_{y \in \mathcal{D}} \frac{|y|-1}{\det(\mathbf{A})^2}\left(\det(\mathbf{A})\frac{\partial \det(\mathbf{B}_y)}{\partial \alpha_n(x)}\right. \\ &\quad \left.\left. - \det(\mathbf{B}_y)\frac{\partial \det(\mathbf{A})}{\partial \alpha_n(x)}\right)\right|_{\left(\alpha_n^{(i-1)}(x)\right)_{n,x}} \\ &= \partial \frac{D_{\text{tot}}}{\alpha_n(x)}\Bigg|_{\left(\alpha_n^{(i-1)}(x)\right)_{n,x}}, \end{aligned}$$

where the last equality holds by the construction of $\mathbf{A}^{(i)}$ and $\mathbf{B}_x^{(i)}$ (14). Finally, Condition 3) in the same lemma need not be satisfied; it is mainly stated in Theorem 1 of [32] for a non-convex *constraint* function. Such condition ensures the satisfaction of complementary slackness conditions by the approximate function and the original function. Since we are interested in the objective function, not a constraint function, Condition 3) is not necessary for convergence to a KKT point. ■

Before we present the algorithm for the proposed successive approximation policy (SAP), we establish that the policy is essentially action-based.

Theorem 3: For every iteration i, the solution to (15) is action-based, hence, the limiting point $\acute{\pi}$ satisfies that for every state $x \in \mathcal{D}$, $\exists n \in x$ such that $\acute{\alpha}_n(x) = 1$, and $\acute{\alpha}_n(x) = 0 \ \forall m \in x$, $m \notin x$.

Proof. We note that (15) is a linear-fractional optimization with the optimization variables being $(\alpha_n(x))_n$. Thus, from Lemma 1, it follows that $\alpha_n^{(i)}(x) = 1$ for some $n \in x$, while $\alpha_m^{(i)}(x) = 0 \ \forall m \neq n$. Since every iteration leads to an action-based allocation policy $\pi^{(i)}$, the limiting point $\acute{\pi}$, which is a locally optimal solution to (8), is also action-based. ■

The action-based property of SAP allows for a simplified solution to every iteration. In particular, for state $x \in \mathcal{D}$ such that $i \mod 2^N + 1 = J(x)$, the solution to (15) is given by $\alpha_{n^{(i)}}^{(i)}(x) = 1$ and $\alpha_m^{(i)}(x) = 0$, $\forall m \neq n^{(i)}$, where $n^{(i)}$ is specified in (16). With the above simplification we present steps leading to our proposed SAP in Algorithm 1.

Algorithm 1 Algorithm for Successive Approximation Policy (SAP)

```
1:  Initialization: iteration $i = 0$, set $\mathbf{A}^{(0)}$, $\mathbf{B}^{(0)}$ as in (14).
2:  while Target convergence accuracy is not reached do
3:      $i = i + 1$.
4:      Compute $\mathbf{A}^{(i)}$, $\mathbf{B}_y^{(i)}$ $\forall y \in \mathcal{X}$ as in (14).
5:      Find $x$ such that $i \mod 2^N + 1 = J(x)$
6:      if $|x| > 1$ then
7:          Compute $n^{(i)}$ from (16).
8:          Set $\alpha_{n^{(i)}}^{(i)}(x) = 1$, and $\alpha_m^{(i)}(x) = 0$ for every $m \in x \backslash \{n^{(i)}\}$.
9:      end if
10: end while
```

While we do not specify a particular criterion for stopping the iterations in Algorithm 1, we note that the algorithm converges in a finite number of steps. The reason is that the SAP is an action-based policy, in which the search for the optimal solution at every iteration runs over a finite set, as such the number of iterations taken to convergence is finite.

Remark 3 (A coordination-game perspective): The proposed SAP algorithm can be viewed as the evolution of *best-response* dynamics in a repeated coordination game [34]. Players are represented by delay states with a common utility function $-D_{\text{tot}}$ and a strategy space for state $x \in \mathcal{D}$ being $\{(\alpha_n(x))_n : \alpha_n(x) \in \{0,1\}, \sum_{n \in x} \alpha_n(x) = 1\}$. From (15), we can see that each player adheres to the best-response strategy given the strategies played by other players in the previous steps. With the finite strategy space for each player, the game is a finite ordinal potential game, and, according

$$n^{(i)} = \arg\min_{n\in x} \frac{\sum_{y\in\mathcal{D}}(|y|-1)\left(\mu_n \operatorname{sgn}(x\backslash\{n\},x)\det(\mathbf{B}_y^{(i)}\langle x\backslash\{n\},x\rangle)\mathbf{1}_{\{y\neq x\}} + \operatorname{sgn}(\mathcal{N},x)\det(\mathbf{B}_y^{(i)}\langle\mathcal{N},x\rangle) + R_{\mathbf{B}_y^{(i)}}(n,x)\right)}{\operatorname{sgn}(x\backslash\{n\})\det(\mathbf{A}^{(i)}\langle x\backslash\{n\},x\rangle) + \operatorname{sgn}(\mathcal{N},x)\det(\mathbf{A}^{(i)}\langle\mathcal{N},x\rangle) + R_{\mathbf{A}^{(i)}}(n,x)}, \quad (16)$$

where

$$R_{\mathbf{M}}(n,x) = \begin{cases} -(\mu_n + \sum_{m\notin x}\lambda_m)\det(\mathbf{M}\langle x,x\rangle) + \operatorname{sgn}(\mathcal{N},x)\det(\mathbf{M}\langle\mathcal{N},x\rangle), & x=\mathcal{N}\backslash\{n\}, \\ \det(\mathbf{M}\langle\mathcal{N},\mathcal{N}\rangle), & x=\mathcal{N}, \\ \sum_{m\notin x}\lambda_m \operatorname{sgn}(x\cup\{m\},x)\det(\mathbf{M}\langle x\cup\{m\},x\rangle) + \operatorname{sgn}(\mathcal{N},x)\det(\mathbf{M}\langle\mathcal{N},x\rangle) & \\ -\left(\mu_n + \sum_{m\notin x}\lambda_m\right)\det(\mathbf{M}\langle x,x\rangle), & \text{otherwise,} \end{cases} \quad (17)$$

and $\mathbf{M}\in\{\mathbf{A}^{(i)},\mathbf{B}_y^{(i)}\}$.

to Theorem 19 in [34], best-response dynamics converge to a Nash equilibrium of the game in a finite number of steps. From Theorem 3, we note that SAP policy can not lead to a higher delay than EPS policy, i.e., $\hat{D}_{\text{tot}} \geq D_{\text{tot}}(\acute{\pi})$. However, we provide a stronger result in the following theorem.

Theorem 4: The SAP strictly outperforms EPS policy unless all ongoing sessions have identical statistics. That is $\hat{D}_{\text{tot}} \geq D_{\text{tot}}(\acute{\pi})$ with equality if and only if $(\lambda_n,\mu_n) = (\lambda_m,\mu_m)$, $\forall m,n\in\mathcal{N}$.

Proof. $(\Rightarrow)$ Suppose that $(\lambda_n,\mu_n) = (\lambda,\mu)$, $\forall n\in\mathcal{N}$. We then can note that the Markov process $\{x(t)\}_t$ under any randomized stationary policy $\pi\in\Pi$ is lumpable to a new process $L_x(t) := |x(t)|$, which captures the number of actions pending service at any time t irrespective of the sessions generating these actions. The lumped process has only $N+1$ states, where the total time average expected delay is given by

$$D_{\text{tot}}(\pi) = \sum_{n=2}^{N}(n-1)\frac{\frac{N!}{(N-n)!}(\lambda/\mu)^n}{\sum_{l=0}^{N}\frac{N!}{(N-l)!}\frac{\lambda}{\mu}^l}.$$

Clearly, $D_{\text{tot}}(\pi)$ is independent of the scheduling policy π, in which case $D_{\text{tot}}(\pi) = D^*_{\text{tot}} = \hat{D}_{\text{tot}}$, $\forall\pi\in\Pi$.

$(\Leftarrow)$ Suppose that $D_{\text{tot}}(\acute{\pi}) = \hat{D}_{\text{tot}}$, and towards contradiction that, W.L.O.G. $(\lambda_1,\mu_1)\neq(\lambda,\mu)$, while $(\lambda_n,\mu_n) = (\lambda,\mu)$, $n=2,\cdots,N$. Now, consider the first occurrence of iteration i for which $i=J(\{1,2\})$, and let $\alpha_1(\{1,2\}) = \alpha$, and $\alpha_2(\{1,2\}) = 1-\alpha$, while the first occurrence of i ensures that $\alpha_n^{(i-1)}(x) = \frac{1}{|x|}$ for all $n\in x$, $x\in\mathcal{D}\backslash\{1,2\}$. Then, $\tilde{D}_{\text{tot}}^{(i)}$ is a monotone function in α for which the minimum is realized at $\alpha\in\{0,1\}$. Hence, $\tilde{D}_{\text{tot}}^{(i)}(\pi^{(i)}) < \hat{D}_{\text{tot}}$. As such, $D_{\text{tot}}(\acute{\pi}) < \hat{D}_{\text{tot}}$, from Theorem 3, which is a contradiction. Thus, $(\lambda_1,\mu_1) = (\lambda,\mu)$. ■

Corollary 1: Optimal action-based scheduling strictly reduces total average expected delay below that of EPS if and only if at least two of the ongoing sessions exhibit different statistics.

Remark 4 (SAP outperforms any non-action-based policy): Corollary 1 can be further generalized to any non-action-based policy other than EPS. That is, for any non-action-based policy $\grave{\pi}$ with less delay than EPS, i.e., $D_{tot}(\grave{\pi}) < \hat{D}_{tot}$, SAP algorithm can be modified such that its initial condition is $\grave{\pi}$ instead of $\hat{\pi}$ and the resulting action-based policy of SAP algorithm will lead to less expected delay than that of $\grave{\pi}$ as long as two of the ongoing sessions exhibit different statistics. The reason is that, starting from any initial condition, the iterations of SAP algorithm yield a sequence of monotonically decreasing expected delay until convergence to a local optimal which is essentially an action-based policy.

The SAP has been proposed to tackle the prohibitive complexity of the globally optimal solution. Under SAP, the search for locally optimal solution scales grows with N as 2^N. While exponential in N, it significantly relaxes the product form $\prod_{n=2}^{N} n\binom{N}{n}$. Moreover, in Section V we show that convergence of SAP algorithm takes as few cycles as two, where each such cycle consists of the $2^N - N - 1$ iterations needed to span the set $\mathcal{D}$ of delay states.

B. Priority-based Scheduling

To further seek simplified action-based scheduling, we consider priority-based scheduling where sessions are assigned to different priority classes and served accordingly. An action from the session with the highest priority always receives service once it has been initiated, thus the highest priority session incurs *no delay*. An action from the session with the second highest priority receives service only when there is no highest priority action in the system, and so on, actions from less priority sessions can only receive service after actions from higher priority sessions have been fully served. Priority-based scheduling has been shown optimal in Section III-A for the two-session system where session n^*'s actions are always served first whenever there is a contention for resources.

The potential of reduced complexity through priority-based scheduling is that the policy is developed in as much as $N-1$ steps, where in each step one session is assigned a certain priority level. We use the notation n_p to denote the session that is served with priority p, $p=1,\cdots,N$. Priority $p=1$ is the highest and $p=N$ is the lowest. Whenever the system is in state x, priority-based scheduling implies that

$\alpha_n(x) = 1$ for $n = n_q$, $q = \min\{p : n_p \in x\}$, while $\alpha_n(x) = 0$, $n \neq n_q$. In this section, we propose two priority-based scheduling policies that achieve efficient delay performance at considerably reduced complexity.

1) Greedy Policy: We begin with a greedy policy determined by an algorithm which sequentially assigns priority to sessions. In every step, one session is selected to receive a current priority order which leads to minimum delay. The total number of steps needed is at most $N-1$. Initially, all sessions are served by EPS policy. Then the N sessions are tested to determine the one that can receive the highest priority and attain minimum total average delay while others remain under EPS. The remaining $N-1$ sessions are then evaluated to select the one that receives the second highest priority given the highest priority session and the EPS scheduling of the other $N-2$ sessions. The algorithm proceeds in the same way until every session has been assigned a priority order.

We denote by $\pi_m^{(p)}$ the policy employed by the algorithm while testing the assignment of session m to priority order p given $n_1, \cdots, n_{p-1}$, and that sessions that have not yet assigned a priority order are served according to EPS. That is, $\pi_m^{(p)}$ assigns the scheduler resources $(\alpha_n(x))_{n,x}$ as follows

$$\alpha_n(x) = \begin{cases} 1, & n = n_q, q = \min\{j : n_j \in x\}, \\ & q = 1, \cdots, p-1, \\ 0, & n = n_q, q > \min\{j : n_j \in x\}, \\ & q = 2, \cdots, p-1, \\ 1, & n = m, m \in x, n_q \notin x, \\ & q = 1, \cdots, p-1, \\ 0, & n \neq m, m \in x, n_q \notin x, \\ & q = 1, \cdots, p-1, \\ \frac{1}{|x|}, & \text{otherwise}, \end{cases} \tag{18}$$

$\forall n \in x$, $x \in \mathcal{X}$, while we use the initial condition that $\pi_m^{(0)} = \hat{\pi}$, $\forall m \in \mathcal{N}$. Then the algorithm selects a session

$$m^* = \arg \min_{\mathcal{N} \setminus \{n_1, \cdots, n_{p-1}\}} D_{\text{tot}}(\pi_m^{(p)}) \tag{19}$$

as a candidate for priority order p. The session m^* is assigned such a priority order only if $D_{\text{tot}}(\pi_{m^*}^{(p)}) < D_{\text{tot}}(\pi_{n_{p-1}}^{(p-1)})$, otherwise, the algorithm terminates at step $p-1$, with $\pi_{n_{p-1}}^{(p-1)}$ being the final greedy policy.

We summarize the operation of the proposed greedy algorithm yielding to the greedy policy in Algorithm 2.

Remark 5: From the construction of the greedy policy we can see that it mixes between action-based scheduling in some states, and EPS in others. However, it is always guaranteed that the policy can never increase delay above that of EPS.

Remark 6: (Greedy policy outperforms any non-action-based policy.) Similar to Remark 4, the initial condition of Algorithm 2 can be modified to be with any non-action-based policy so that the resulting greedy action-based policy strictly outperforms the initial non-action-based one.

Algorithm 2 Algorithm for Greedy Policy

```
1:  Initialization: step 0, $\pi_m^{(0)} = \hat{\pi}$, $\forall m \in \mathcal{N}$.
2:  for $p = 1$ to $N-1$ do
3:      for $m \in \mathcal{N} \setminus \{n_1, \cdots, n_{p-1}\}$ do
4:          Compute $\pi_m^{(p)}$ for which $\alpha_n(x)$ is specified in (18).
5:          Compute $D_{\text{tot}}(\pi_m^{(p)})$ from (7).
6:      end for
7:      Compute $m^*$ from (19).
8:      if $D_{\text{tot}}(\pi_{m^*}^{(p)}) < D_{\text{tot}}(\pi_{m_{p-1}}^{(p-1)})$ then
9:          $n_p = m^*$
10:     else
11:         Terminate algorithm with $\pi_{n_{p-1}}^{(p-1)}$ being the greedy
            policy.
12:     end if
13: end for
14: Terminate the algorithm with $\pi_{n_{N-1}}^{(N-1)}$ being the greedy
    policy.
```

Remark 7: The search complexity for the priority order of the greedy policy scales quadratically with the number of sessions.

2) Two-session Comparison Policy (TSCP): In this approach we aim to further simplify the priority-based scheduler design by utilizing the structure of the optimal solution to the two-session scenario. In particular, we define a metric $\phi_{n,m}$ to compare any two sessions as in (5) and accordingly give one session priority over the other. We then rank the sessions based on their relative comparisons with respect to $\phi_{n,m}$, that ranking is harnessed to assign the priority-based schedule of the TSCP. Our proposed two-session comparison metric $\phi_{n,m} :=$

$$\begin{cases} 1, & \frac{\mu_n}{\lambda_n^2 + \lambda_n\lambda_m + \lambda_n\mu_m + \mu_n\mu_m} > \frac{\mu_m}{\lambda_m^2 + \lambda_n\lambda_m + \lambda_m\mu_n + \mu_m\mu_n}, \\ 0, & \text{otherwise}, \end{cases} \tag{20}$$

$\forall n, m \in \mathcal{N}$, and

$$\phi_n := \sum_{m=1}^{N} \phi_{n,m}. \tag{21}$$

We then rank all the sessions through the ϕ_n metric and assign them service priorities accordingly. The TSCP thus determines session n_p such that

$$\phi_{n_p} \geq \phi_m, \quad \forall m \in \mathcal{N} \setminus \{1, \cdots, n_{p-1}\}, \tag{22}$$

with ties broken arbitrarily.

The steps leading to the construction of the TSCP are summarized in Algorithm 3.

Remark 8: We note that the development of the TCSP does not require any sophisticated matrix-related operations such as computation of determinants. Instead, it relies on simple operations specified by the $\phi_{n,m}$ metric. Hence, the overall complexity of TCSP is remarkably less than that of the SAP and greedy policy.

Remark 9: Despite the reduced complexity of the TSCP, closed form expressions asserting its performance guarantees

Algorithm 3 Algorithm for Two-session Comparison Policy (TSCP)

```
1: Initialization: step 0, compute $\phi_{n,m}$ for all sessions $n, m \in \mathcal{N}$.
   Compute $\phi_n$ for all sessions $n \in \mathcal{N}$ from (21).
2: for $p = 1$ to $N$ do
3:     Compute $n_p$ as in (22)
4: end for
```

are analytically formidable for a general number of sessions. Nevertheless, numerical simulations in Section V demonstrate its relative efficiency over the EPS policy.

V. Numerical Simulations

In this section we provide numerical simulations based on real-data trace of smart-phone app traffic involving human interactions. The data trace reported in [35] encompasses packet-level detail of 1500 sessions collected from 5 different smart-phone apps, namely Google Maps (travel), Google Chrome (web-browsing), Yelp (dinning and local search), Expedia (travel planner), and GT Racing 2 (online gaming). From the findings reported in [19] on the number of packets per action as well as the time gaps between consecutive actions, we capture the main characteristics of app sessions as follows. Over the WiFi network the AP transmission rate is 54Mbps, thus we take the time slot duration to be 326μs which is the packet transmission time. From Table I in [19] which lists the average number of packets per action, we compute the parameter μ_n for every app session n as the reciprocal of the average number of packets per action. From Table II in [19] which lists the time gap statistics between any two consecutive actions, we compute the parameter λ_n as the ratio of slot duration over the average time gap between two actions. As such, we summarize the values of (λ_n, μ_n) pairs for the five tested apps in Table I.

TABLE I: Pairs of (λ_n, μ_n) for the five tested apps.

Maps	$(0.126, 1.837) \times 10^{-3}$
Chrome	$(0.109, 2.041) \times 10^{-3}$
Yelp	$(0.131, 4.032) \times 10^{-3}$
Expedia	$(0.121, 14.492) \times 10^{-3}$
GT Racing 2	$(0.085, 6.238) \times 10^{-3}$

Our numerical simulations are divided over two subsections. In Subsection V-A, the system's performance is evaluated against the baseline centralized EPS scheduling, whereas in Subsection V-B, the performance is evaluated against the distributed service of the WiFi 802.11e's EDCA function.

A. Comparison with EPS:

To highlight the potential gains of action-based scheduling, we begin with the two-session scenario where every possible pair of the tested apps is considered for scheduling. As the optimal solution is fully determined in Proposition 2, we provide the delay reduction gain η^* (6) in Table II. Recall that delay reduction gain η^* measures the relative delay improvement over the equal processor sharing policy. We can see from the table that action-based scheduling promises of considerable delay reduction gain reaching values of 43% of delay savings. The gains associated with the the Expedia app are the highest. The reason is the high processing speed of actions of such app, that essentially gives it privilege to receive service first at minimum delay expense for other sessions sharing the same AP. We also observe 0 gains associated with the service of two sessions from the same app since the scheduler is indifferent to traffic characteristics of each session. The service of Maps and Chrome leads to relatively small delay savings for their user thinking dynamics and action service times exhibit highly similar characteristics.

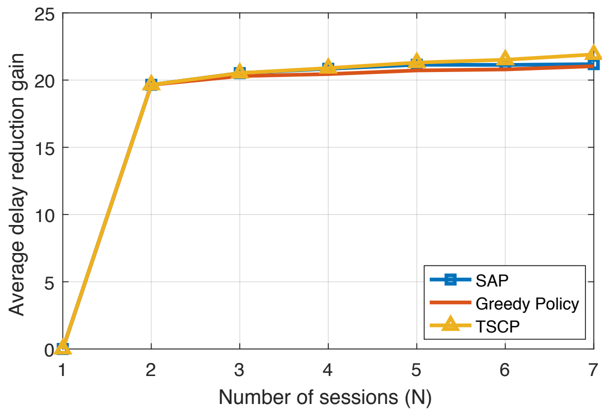
TABLE II: Delay reduction gain η^* for every pair of tested apps.

	Maps	Chrome	Yelp	Expedia	GT Racing 2
Maps	0	5.1%	26.34%	43.3%	34.9%
Chrome	5.1%	0	23.78%	42.6%	33.25%
Yelp	26.34%	23.78%	0	35.8%	17.68%
Expedia	43.3%	42.6%	35.8%	0	28.19%
GT Racing 2	34.9%	33.25%	17.68%	28.19%	0

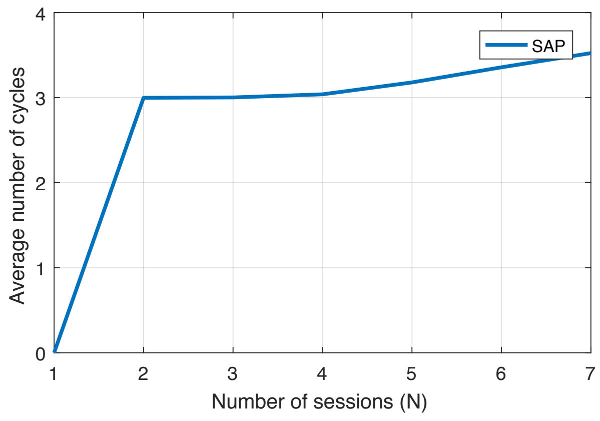
For the multi-session case, we evaluate the performance of our proposed three policies versus the number of sessions. Driven by the data of Table I, we randomly generate app sessions with λ_n being uniformly distributed on the interval $[0.085, 0.131] \times 10^{-3}$ and μ_n uniformly distributed on $[1.837, 14.92] \times 10^{-3}$. For every number of sessions we average the results over 10^4 simulation runs. We compute the delay reduction gain, η, for each policy, and plot the results in Fig. 5a. We can observe that the three policies have close performance, and interestingly TSCP is highly efficient despite its remarkable low complexity relative to the SAP and the greedy policy. We note also that the delay reduction gain is substantial reaching more than 22% as the number of sessions grows.

We evaluate the complexity of the SAP by measuring the average number of cycles of iterations over which the SAP algorithm runs over the delay states until convergence. Each cycle comprises $2^N - N - 1$ iterations. In Fig. 5b, we plot the average number of such cycles against the number of sessions. The figure clearly shows that convergence is realized in no more than two cycles, on average.

We have noted from Table I that λ_n's are at least one order of magnitude smaller than μ_n's, which is reasonable noting that human's spend more time comprehending and processing information sent from servers in every action. Under such order of magnitude difference, we find that the TSCP exhibits high efficiency. Nevertheless, the TSCP's performance falls dramatically relative to the SAP and the greedy policy as the difference between λ_n and μ_n declines.

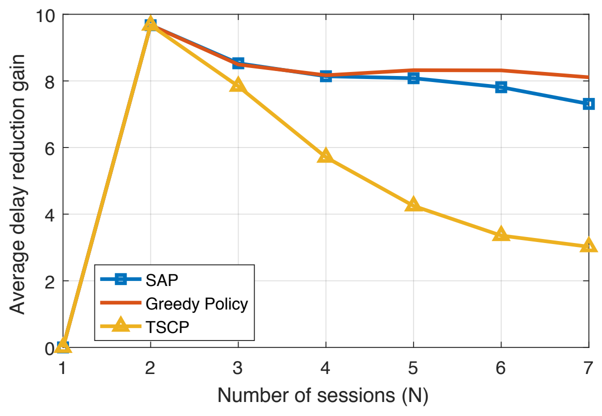


(a) Delay reduction gain vs. number of sessions.

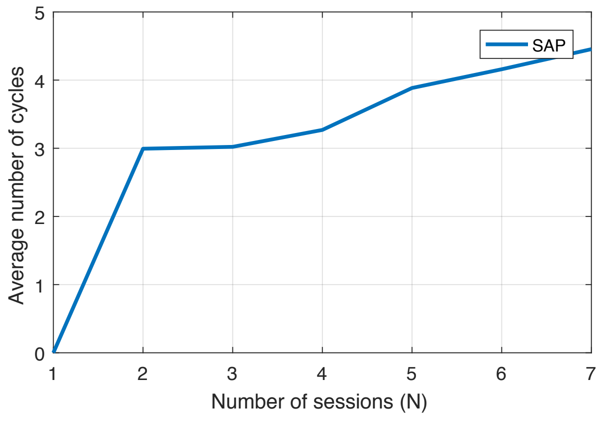


(b) Average number of $2^N - (N+1)$ cycles to convergence vs. N.

Fig. 5: Performance of proposed policies for λ_n and μ_n being uniformly distributed on $[0.085, 0.131] \times 10^{-3}$ and $[1.837, 14.92] \times 10^{-3}$, respectively.



(a) Delay reduction gain vs. number of sessions.



(b) Average number of $2^N - (N+1)$ cycles to convergence vs. N.

Fig. 6: Performance of proposed policies for μ_n being uniformly distributed on $[1.837, 14.92] \times 10^{-3}$ and $\lambda_n = \rho_n \mu_n$, where ρ_n is uniformly distributed on $[0.3, 0.8]$.

We simulate a system where λ_n is a linear fraction of μ_n as $\lambda_n = \rho_n \mu_n$ where ρ_n is uniformly distributed on $[0.3, 0.8]$ to capture the scenario when λ_n and μ_n are of the same order of magnitude, which hypothetically indicates fast thinking time by end-users. Results are also averaged over 10^3 simulation runs. In Fig. 6a, the delay reduction gain efficiency is in general significantly reduced compared to that of Fig. 5a as the system gets more congested with interactions. We can note that the greedy policy attains the best performance among the three tested policies while TSCP's performance drops fast as the number of sessions grows. In Fig. 6b we study the convergence speed of SAP where it takes more cycles to convergence compared to the realistic system of Fig. 5b. Overall, we can conclude that the proposed priority-based scheduling policies deliver a favorable balance between performance efficiency and implementation complexity.

B. Comparison with EDCA

The WiFi 802.11e standard allows for different QoS scheduling by prioritizing the service of four different types of traffic, called access categories (ACs). This QoS differentiation is realized through an enhanced distributed channel access (EDCA) function that assigns smaller access parameters to ACs with higher priority. These parameters are the arbitration inter-frame spacing (AIFS) and the minimum and maximum collision window sizes (CWMin) and (CWMax), respectively, [22], [23], [24].

In this subsection, we simulate and compare the random access EDCA function with our app-aware TSCP scheduling policy for the four apps: Google Chrome, Yelp, Expedia, and GT Racing 2 whose traffic characteristics are listed in Table I. The simulated EDCA operation assumes that each ongoing app session sends packets randomly, yet upon collision, the sending sessions must back-off for a number of time slots drawn uniformly random from the range

$$[1, \min\{2^i * \text{CWMin}(p), \text{CWMax}(p)\}],$$

where i is the collision stage and p is the priority index of the session involved in the collision event.

The CWMin and CWMax values for the four priority indexes considered in this simulation are as listed in Table III where aCWmin is a simulation parameter that we vary

TABLE III: Contention window boundaries for the simulated priority levels.

Priority Index (p)	CWMin	CWMax
1	$\frac{\text{aCWmin}+1}{4} - 1$	$\frac{\text{aCWmin}+1}{2} - 1$
2	$\frac{\text{aCWmin}+1}{2} - 1$	aCWmin
3	aCWmin	$2(\text{aCWmin} + 1) - 1$
4	$2(\text{aCWmin} + 1) - 1$	$4(\text{aCWmin} + 1) - 1$

in simulation. We further consider AIFS to be similar for all sessions in this simulation. Thus priority scheduling of EDCA is mainly determined based on the values of CWMin(p) and CWMax(p). The priority indexes themselves are assigned based on the metric in (22).

The average delay resulting from the EDCA scheduling is denoted D_{tot}(EDCA), whereas that of TSCP is denoted by D_{tot}(TSCP). We consider the delay reduction gain of TSCP over EDCA as our performance metric and plot it in Fig. 7.

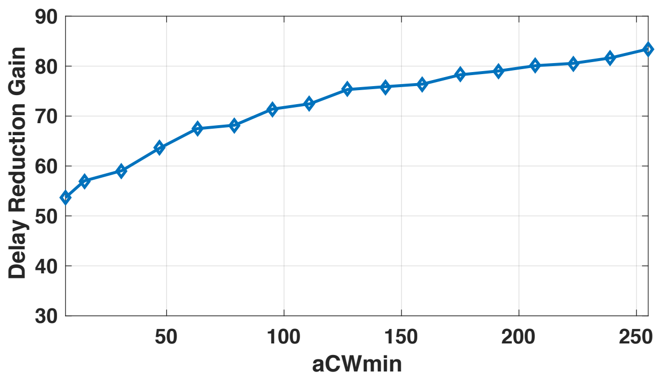


Fig. 7: Delay improvement of the proposed TSCP action-based scheduling over random access EDCA vs aCWmin.

With one simulation run spanning 10^6 time slots and averaging over a 100 simulation runs, we plot in Fig. 7 the average delay reduction gain realized by TSCP over that of EDCA as the parameter aCWmin varies between 7 and 255. The figure clearly highlights the QoE improvement of at least 53% that TSCP can achieve. The TSCP harnesses app-aware centralized scheduling capabilities that not only rank the service of app traffic for reduced delay, but also it does not suffer the delays from the random back-off times that EDCA incurs.

VI. Conclusion

In this work, we have considered the problem of serving multiple smart-phone app sessions involving human interactions and sharing the same bottleneck wireless scheduler. We have built on top of recent findings on the characteristics of human-app interactions in the timescale of seconds where human's influence on traffic patterns is pronounced. In particular, we have utilized an action-based model where each app session is composed of a sequence of actions, each is formed by uplink and downlink transmissions constituting the query-response pair of such action. In the light of the action-based model, we have accounted for two specific properties of user-server interactivity. Namely, (1) the time taken by users to process the server's response and generate a new action spans a few seconds, on average. Such human thinking time creates periods of silence in the packet flow of the app session. (2) The generation of a new action for a given session is contingent upon the service of the current action. That is, no new action will be generated unless all packets of the current action have been served, thus future arrivals depend on the service strategy at the wireless scheduler.

For the adopted model, we have formulated the problem of maximizing end-users' QoE in the form of minimizing the total time average expected delay. While the problem is generally of infinite dimension, we have proved that all locally optimal solutions to the problem are non-randomized, i.e., *action-based*, which reduces the problem complexity into a finite search problem. In action-based scheduling, the scheduler decides to serve the packets of an action from one session as a whole rather than switching service between packets of different sessions. As the optimal action-based policy requires an exhaustive search with formidable complexity, we have developed reduced complexity policies that are shown analytically and validated numerically to attain improved delay performance over traditional non-action-based or packet-based policies such as equal processor sharing.

References

[1] M. Feknous, T. Houdoin, B. Le Guyader, J. De Biasio, A. Gravey, and J. Torrijos Gijon, "Internet traffic analysis: A case study from two major european operators," *Proc. of Symposium on Computers and Communication (ISCC)*, pp. 1–7, June 2014.

[2] J. Li, R. Bhattacharyya, S. Paul, S. Shakkottai, and V. Subramanian, "Incentivizing sharing in realtime D2D streaming networks: A mean field game perspective," in *2015 IEEE Conference on Computer Communications (INFOCOM)*, pp. 2119–2127, April 2015.

[3] P. C. Hsieh and I. H. Hou, "Heavy-traffic analysis of qoe optimality for on-demand video streams over fading channels," in *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications*, pp. 1–9, April 2016.

[4] H. Wu, X. Lin, X. Liu, and Y. Zhang, "Application-level scheduling with probabilistic deadline constraints," vol. 24, pp. 1504–1517, June 2016.

[5] A. ParandehGheibi, M. Medard, A. Ozdaglar, and S. Shakkottai, "Avoiding interruptions- a qoe reliability function for streaming media applications," vol. 29, pp. 1064–1074, May 2011.

[6] L. S. Muppirisetty, J. Tadrous, A. Eryilmaz, and H. Wymeersch, "On proactive caching with demand and channel uncertainties," *2015 53rd Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pp. 1174–1181, Sept 2015.

[7] J. Tadrous and A. Eryilmaz, "On optimal proactive caching for mobile networks with demand uncertainties," *IEEE/ACM Transactions on Networking*, vol. PP, no. 99, pp. 1–1, 2015.

[8] E. Bastug, M. Bennis, and M. Debbah, "Living on the edge: The role of proactive caching in 5G wireless networks," *IEEE Communications Magazine*, vol. 52, pp. 82–89, Aug 2014.

[9] L. Huang, S. Zhang, M. Chen, and X. Liu, "When backpressure meets predictive scheduling," *Proceedings of the 15th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, pp. 33–42, 2014.

[10] S. Zhang, L. Huang, M. Chen, and X. Liu, "Effect of proactive serving on user delay reduction in service systems," *SIGMETRICS Perform. Eval. Rev.*, vol. 42, pp. 573–574, June 2014.

[11] S. Dimatteo, P. Hui, B. Han, and V. O. K. Li, "Cellular traffic offloading through wifi networks," *2011 IEEE Eighth International Conference on Mobile Ad-Hoc and Sensor Systems*, pp. 192–201, Oct 2011.

[12] L. Xiaofeng, H. Pan, and P. Lio, "Offloading mobile data from cellular networks through peer-to-peer wifi communication: A subscribe-and-send architecture," *Communications, China*, vol. 10, pp. 35–46, June 2013.

[13] J. Lee, Y. Yi, S. Chong, and Y. Jin, "Economics of wifi offloading: Trading delay for cellular capacity," *Computer Communications Workshops (INFOCOM WKSHPS), 2013 IEEE Conference on*, pp. 357–362, April 2013.

[14] M. H. Cheung, R. Southwell, and J. Huang, "Congestion-aware network selection and data offloading," *Information Sciences and Systems (CISS), 2014 48th Annual Conference on*, pp. 1–6, March 2014.

[15] L. Zhang, W. Wu, and D. Wang, "Time dependent pricing in wireless data networks: Flat-rate vs. usage-based schemes," *INFOCOM, 2014 Proceedings IEEE*, pp. 700–708, April 2014.

[16] S. Sen, C. Joe-Wong, S. Ha, and M. Chiang, "Incentivizing time-shifting of data: a survey of time-dependent pricing for internet access," *Communications Magazine, IEEE*, vol. 50, pp. 91–99, November 2012.

[17] J. Tadrous, H. El Gamal, and A. Eryilmaz, "Can carriers make more profit while users save money?," *Information Theory (ISIT), 2014 IEEE International Symposium on*, pp. 1757–1761, June 2014.

[18] J. Tadrous, A. Eryilmaz, and H. El Gamal, "Joint smart pricing and proactive content caching for mobile services," *Networking, IEEE/ACM Transactions on*, vol. PP, no. 99, pp. 1–1, 2015.

[19] J. Tadrous and A. Sabharwal, "Interactive app traffic: Action-based model and data driven analysis," *14th International Symposium on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks,*, pp. 1–8, May 2016.

[20] S. Miskovic, G. M. Lee, Y. Liao, and M. Baldi, *AppPrint: Automatic Fingerprinting of Mobile Applications in Network Traffic*, pp. 57–69. Cham: Springer International Publishing, 2015.

[21] S. Dai, A. Tongaonkar, X. Wang, A. Nucci, and D. Song, "Networkprofiler: Towards automatic fingerprinting of android apps," *INFOCOM, 2013 Proceedings IEEE*, pp. 809–817, April 2013.

[22] H. Wu, X. Wang, Q. Zhang, and X. Shen, "Ieee 802.11e enhanced distributed channel access (edca) throughput analysis," in *2006 IEEE International Conference on Communications*, vol. 1, pp. 223–228, June 2006.

[23] Z. Tao and S. Panwar, "Throughput and delay analysis for the ieee 802.11e enhanced distributed channel access," *IEEE Transactions on Communications*, vol. 54, pp. 596–603, April 2006.

[24] S. Choi, J. del Prado, S. S. N, and S. Mangold, "Ieee 802.11 e contention-based channel access (edcf) performance evaluation," in *Communications, 2003. ICC '03. IEEE International Conference on*, vol. 2, pp. 1151–1156 vol.2, May 2003.

[25] K. Ross and B. Chen, "Optimal scheduling of interactive and non-interactive traffic in telecommunication systems," *Automatic Control, IEEE Transactions on*, vol. 33, pp. 261–267, Mar 1988.

[26] L. Almajano and J. Perez-Romero, "Packet scheduling algorithms for interactive and streaming services under qos guarantee in a cdma system," *Vehicular Technology Conference, 2002. Proceedings. VTC 2002-Fall. 2002 IEEE 56th*, vol. 3, pp. 1657–1661 vol.3, 2002.

[27] T. Li, D. Leith, and L. Qiu, "Opportunistic routing for interactive traffic in wireless networks," *Distributed Computing Systems (ICDCS), 2010 IEEE 30th International Conference on*, pp. 458–467, June 2010.

[28] W. Song and W. Zhuang, "Multi-class resource management in a cellular/wlan integrated network," *Proc. of WCNC*, 2007.

[29] W. Song and W. Zhuang, "Resource allocation for conversational, streaming, and interactive services in cellular/wlan interworking," *Proc. of GLOBECOM*, 2007.

[30] Y. Zheng, B. Ji, N. Shroff, and P. Sinha, "Forget the deadline: Scheduling interactive applications in data centers," *2015 IEEE 8th International Conference on Cloud Computing*, pp. 293–300, June 2015.

[31] S. M. Ross, *Applied Probability Models with Optimization Applications*. Dover Publications, INC., New York, 1972.

[32] G. P. W. Barry R. Marks, "A general inner approximation algorithm for nonconvex mathematical programs," *Operations Research*, vol. 26, no. 4, pp. 681–683, 1978.

[33] S. Boyd and L. Vandenberghe, *Convex Optimization*. New York, NY, USA: Cambridge University Press, 2004.

[34] A. B. MacKenzie and L. A. DaSilva, *Game Theory for Wireless Engineers*. Morgan & Claypool, 2006.

[35] J. Tadrous and A. Sabharwal, "A guide to inetaractive smartphone app traffic dataset." https://dl.dropboxusercontent.com/u/16856629/Guidetointeractivedataset/GuideInteractiveDataset.pdf.