Timely Wireless Flows With General Traffic Patterns: Capacity Region and Scheduling Algorithms

Lei Deng, Chih-Chun Wang, *Senior Member, IEEE*, Minghua Chen, *Senior Member, IEEE*, and Shizhen Zhao

***Abstract*— Most existing wireless networking solutions are best-effort and do not provide any delay guarantee required by important applications, such as mobile multimedia conferencing and real-time control of cyber-physical systems. Recently, Hou and Kumar provided a novel framework for analyzing and designing delay-guaranteed wireless networking solutions. While inspiring, their idle-time-based analysis applies only to flows with a special traffic pattern called *the frame-synchronized setting*. The problem remains largely open for *general traffic patterns*. This paper addresses this challenge by proposing a general framework that characterizes and achieves the complete delay-constrained capacity region with general traffic patterns in single-hop downlink access-point wireless networks. We first show that the timely wireless flow problem is fundamentally an infinite-horizon Markov decision process (MDP). Then, we judiciously combine different simplification methods to prove that the timely capacity region can be characterized by a finite-size convex polygon. This for the first time allows us to characterize the timely capacity region of wireless flows with general traffic patterns. We then design three scheduling policies to optimize network utility and/or support feasible timely throughput vectors for general traffic patterns. The first policy achieves the optimal network utility and supports any feasible timely throughput vector but suffers from the curse of dimensionality. The second and third policies are inspired by our MDP framework and are of much lower complexity. Simulation results show that both achieve near-optimal performance and outperform other existing alternatives.**

***Index Terms*— Delay-constrained wireless communications, general traffic pattern, timely capacity region, network utility maximization, Markov decision process (MDP).**

Manuscript received July 6, 2016; revised January 11, 2017; accepted August 11, 2017; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor T. Hou. Date of publication September 25, 2017; date of current version December 15, 2017. This work was supported in part by the NSF under Grant ECCS-1407603, Grant CCF-1422997, and Grant CCF-1618475, in part by the National Basic Research Program of China under Grant 2013CB336700, in part by the University Grants Committee, Hong Kong Special Administrative Region, China, under Grant 2150871, and in part by the Innovation and Technology Committee, Hong Kong Special Administrative Region, China, under Grant 14201014. Part of this work has been presented at the 35th IEEE International Conference on Computer Communications (INFOCOM), San Francisco, USA, April 10–15, 2016 [1]. *(Corresponding author: Lei Deng.)*

L. Deng and M. Chen are with the Department of Information Engineering, The Chinese University of Hong Kong, Hong Kong (e-mail: dl013@ie.cuhk.edu.hk; minghua@ie.cuhk.edu.hk).

C.-C. Wang is with the School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47906 USA (e-mail: chihw@purdue.edu).

S. Zhao is with the Google Platform Networking Team, Google Inc., Mountain View, CA 94043 USA (e-mail: shizhenzhao1988@gmail.com).

This paper has supplementary downloadable material available at http://ieeexplore.ieee.org, provided by the author.

Digital Object Identifier 10.1109/TNET.2017.2749513

I. INTRODUCTION

A. Background

REAL-TIME wireless communication systems that require delay guarantee have become prevalent. Typical systems of this kind include multimedia communication systems such as real-time streaming and video conferencing over cellular networks, and cyber-physical systems (CPSs) such as real-time surveillance and control over wireless sensor networks. As a consequence, real-time wireless traffic has experienced a phenomenal growth in recent years [7], and is predicted to increase its volume by another 11-fold in 2015-2020 [8].

A common characteristic of these systems is that they have a strict deadline for packet delivery. Packets traversing the wireless network need to be delivered before their deadlines, otherwise they expire and deem useless. For example, mobile video conferencing may require bounded delay on video delivery [9]. Similarly, in CPSs, time-critical applications impose latency constraints within which data or control messages must reach their targeting entities [10]. Additionally, real-time wireless communication systems often require performance on the *timely throughput*, defined as the throughput of packets that are delivered on time [2].

B. Challenges

Serving delay-constrained traffic over wireless networks is uniquely challenging due to the inherent coupling of space, time, and unreliable transmission.

Space: Wireless networks differ from wired networks in the presence of spatial interference, wherein the transmission over a link can upset other transmissions in its neighborhood. An optimal scheduler needs to carefully decide which link/flow to serve at a given time slot.

Time: To ensure timely packet delivery, one also has to keep track of deadlines of individual packets and properly account for delivery urgency in scheduling link transmissions. Such unique feature in the time domain often introduces high-dimensional system state.

Unreliable Transmission: Wireless transmissions are unreliable because of shadowing and fading. The channel quality may also differ from link to link. This could result in significant delay when a delay-oblivious scheduling scheme is used.

C. Fundamental Problems

In delay-constrained wireless communications, there are three fundamental problems.

TABLE I
OUR CONTRIBUTIONS AND COMPARISONS WITH EXISTING WORKS. ALL OF THEM CONSIDER A SINGLE-HOP DOWNLINK AP SCENARIO

Traffic Pattern	Capacity Region	Scheduling Policy for Network Utility Maximization	Feasibility-Optimal Scheduling Policy Design
Frame-Synchronized ([2], [3])	[2]	[3]	[2]
General (this work)	A convex polygon (Sec. V)	RAC, optimal, high complexity (Sec. V)	RAC, optimal, high complexity (Sec. V)
	A fast outer bound (Sec. VI)	RAC-Approx, heuri., low compl. (Sec.VI)	RAC-Approx, heuri., low compl. (Sec. VI)
			L-LDF, heuristic, low complexity (Sec. VI)

Capacity Region Problem: How to characterize the capacity region in terms of timely throughput? This problem is important because: (i) it provides the fundamental benchmark to evaluate any scheduling policy, and (ii) it lays down the necessary foundation for network utility maximization in terms of timely throughput.

Network Utility Maximization (NUM) Problem: How to design scheduling policies to maximize network utility in terms of timely throughput? This is the *delay-constrained* counterpart of the celebrated NUM framework for *delay-unconstrained* wireless flows, which has been widely used as both a modeling language and solution tools [11]–[13].

Feasibility-Optimal Policy Design Problem: A common by-product of solving the capacity region problem is that one can obtain a scheduling policy to support *one* feasible throughput vector in the region, by solving an optimization problem (a linear one if the capacity region is a polyhedron). This approach, however, may result in using (many) different policies to support different feasible rate vectors, one policy for each or multiple rate vectors. In practice, it is more desirable to implement only one policy that can support any feasible rate vectors.

For the delay-unconstrained scenario, the celebrated back-pressure algorithm [14] can support *any* feasible rate vector within the delay-unconstrained capacity region, and it is termed *throughput-optimal*. For the delay-constrained scenario studied in this paper, following the terminology coined in [2], we call a policy *feasibility-optimal* if it can support any feasible throughput vector within the timely capacity region. A central problem of scheduling delay-constrained traffic is to design a feasibility-optimal policy.[1]

Systematically solving these three fundamental problems calls for a framework that both captures the challenges in Sec. I-B of delay-constrained wireless communications and offers tractable solutions.

D. Our Contributions

Recently, researchers devoted much effort to studying real-time wireless communications [2]–[6], [15]–[19]. Among them, Hou *et al.* [2] and Hou and Kumar [3]–[6] developed an elegant idle-time-based framework to solve all the three fundamental problems in Sec. I-C for a special frame-synchronized traffic pattern, over single-hop downlink access-point (AP) wireless networks. Inspiring as it is, their idle-time-based framework apparently only applies to flows with the special traffic pattern, which can only capture a limited number of practical scenarios. Overall, the three fundamental problems in Sec. I-C remain largely open for general traffic patterns.

In this paper, we take a first step towards solving these three fundamental problems for *general traffic patterns* by establishing a framework based on Markov Decision Process (MDP). The structure of the timely wireless flow problem makes MDP a natural candidate for establishing such framework. We summarize our contributions about how we solve these problems and compare them with existing works in Tab. I. Specifically, we make the following contributions:

▷ In Sec. III, we model general traffic patterns. Then in Sec. IV, we show that the timely wireless flow problem with general traffic patterns is fundamentally an MDP problem. This new observation allows us to systematically explore the *full* design space, beyond those in previous studies [2]–[6].

▷ The MDP formulation is challenging to solve. In particular, it is of infinite-horizon, infinite state space, and time-heterogeneous. In Sec. V, by leveraging the underlying structure of the MDP formulation, we apply two simplification methods to show that the timely capacity region is a finite-size convex polygon. Our results build upon the rich literature of MDP to *judiciously formulate the problem and adapt several existing techniques of MDP in a coherent way so as to fully answer the fundamental problem: "What is the capacity region for timely flows with general traffic patterns?"* As a by-product of the capacity region analysis, we obtain a provably optimal scheduling policy, called RAC, for network utility maximization. We also show that RAC is feasibility-optimal.

▷ Our capacity region characterization and the optimal RAC scheduler suffer from the curse of dimensionality rooted in the MDP approach. To address this issue, in Sec. VI, we first propose a relaxed but computationally-efficient convex polygon characterization, serving as a fast outer bound of the capacity region. Based on the outer bound analysis, we propose a low-complexity heuristic scheduling policy, called RAC-Approx, for optimizing network utility and supporting feasible timely throughput vectors. Motivated by our model for system state, we also propose another low-complexity heuristic scheduling policy, called L-LDF, for supporting feasible timely throughput vectors.

▷ In Sec. VII, we carry out extensive simulations to verify the optimality of our RAC scheduler and show that our proposed heuristic scheduler are near-optimal and outperform other conceivable alternatives.

II. RELATED WORK

Supporting delay-constrained traffic over wireless networks has been a very active research area and we review the most relevant works in the following by categorizing them according to the three fundamental problems in Sec. I-C.

[1]Note that largest-deficit-first (LDF) policy proposed by Hou *et al.* in [2] is feasibility-optimal, for a special traffic (arrival and expiration) pattern. In this paper, we design a feasibility-optimal policy for general traffic patterns.

Capacity Region Problem: For the special frame-synchronized traffic pattern, Hou *et al.* in the seminal paper [2] proposed an idle-time based approach to characterize the capacity region of timely flows over a single-hop downlink AP scenario. The approach has been further extended to variable-bit-rate applications in [4] and time-varying channels in [6]. However, to the best of our knowledge, there are no results to characterize the capacity region for general traffic patterns beyond the special frame-synchronized traffic pattern. In this work, we fill this gap and give a complete characterization of the capacity region for general traffic patterns.

NUM Problem: Still for the special frame-synchronized traffic pattern, Hou and Kumar in [3] solved the NUM problem efficiently for the single-hop downlink AP scenario where each user has a general and valid utility function in terms of the achieved timely throughput. Later, Lashgari and Avestimehr [15] generalized the single-AP scenario to a multi-AP scenario, but still focused on the frame-synchronized traffic pattern and only considered a linear utility function for each user. They proposed a relaxed bin-packing problem with elegant insights for the original complicated network utility maximization problem, and provided some theoretical guarantees for such relaxation. However, there is little result on network utility maximization beyond the special frame-synchronized traffic pattern. Our work addresses this open issue.

Feasibility-Optimal Scheduling Policy Design Problem: For the special frame-synchronized traffic pattern, Hou *et al.* in [2] proposed the celebrated largest-deficit-first (LDF) scheduling policy and proved that it is feasibility-optimal in the sense that it can support any feasible timely throughput vectors. However, it turns out that LDF is not feasibility-optimal for general traffic patterns [16], [19]. There have been two lines of efforts to study the feasibility-optimal policy design problem for general traffic patterns. One is to study the performance of LDF. Kang *et al.* in [19] proposed a theoretical lower bound for the quality of service (QoS) efficiency ratio, i.e., the fraction of capacity region that can be achieved by LDF. Further, in [20], Kang *et al.* derived theoretical upper and lower bounds for the capacity efficiency ratio, i.e., the minimum link capacity required by LDF to achieve the capacity region in the same network with unit-capacity links. Both [19] and [20] considered an "i.i.d." traffic pattern. The other line is to propose new scheduling algorithms to support feasible timely throughput vectors. Hou and Singh [16] proposed the *Earliest-Positive-deficit-Deadline-First* (EPDF) scheduling policy, which is shown to outperform LDF numerically for the frame-based heterogenous-delay traffic pattern. However, currently there is not yet feasibility-optimal scheduling policy for general traffic patterns. This work fills in this gap and proposes a feasibility-optimal policy for general traffic patterns.

III. System Model and Problem Formulation

A. The Communication Model

Network Topology and Scheduling Model: We consider a single-hop downlink access-point (AP) scenario where the AP aims to transmit K independent timely traffic to K users,[2] one for each user. The traffic (resp. channel) between the AP and user $k \in [1, K]$ is denoted as flow (resp. link) k. Assume slotted transmission. In each slot, only one link can be scheduled and can only send one packet. At the beginning of slot t, the action of the AP, denoted by A_t, thus decides which flow/link to schedule.[3] At the beginning of slot $(t+1)$, the AP can choose a different A_{t+1} and the process starts over. For easier reference, we use "at time (slot) t" to refer to "at the beginning of slot t" and use "in time (slot) t" to refer to "in the time span of slot t."

Propagation Delay and Random Erasure: To model propagation delay, we assume that if link k is scheduled at time t, then the transmitted packet can be received by user k at the *end* of time t. To model unreliable transmission of wireless channels, we assume that along any link k successful delivery happens with some probability $p_k \in (0, 1]$, the random delivery events are independently and identically distributed (i.i.d.) over time, and the events for different links are independent.

We also assume that at the end of time t, the scheduled user will inform the AP through a separate control channel whether it has received the transmitted packet or not (ACK/NACK). The information will then be used for scheduling at time $(t+1)$ and onward.

The above model captures the practical Wi-Fi networks and is also widely adopted in the real-time wireless communications literature, e.g., [2], [3], [5], [6], [15]. We also remark that although we consider an ON-OFF channel model in this paper, our results can be extended to the general multi-state channel model [21].

B. Traffic Pattern

We assume *periodic-i.i.d.* packet arrivals with hard delay constraints for each flow k, which can be best described by the following concept of "arrival and expiration (A&E) profile." For any flow k, its A&E profile can be described by a 4-dim. vector

$$(\mathsf{offset}_k, \mathsf{prd}_k, \mathsf{D}_k, \mathsf{B}_k),$$

where offset_k denotes the time offset for the start of the arrival process of flow k; prd_k is the inter-arrival period of flow k; D_k is the deadline for each flow-k packet; and $\mathsf{B}_k \in (0, 1]$ is the arrival probability of each flow-k packet. For flow k with an A&E profile $(\mathsf{offset}_k, \mathsf{prd}_k, \mathsf{D}_k, \mathsf{B}_k)$, we denote the arrival time of the m-th[4] flow-k packet as $t_{\text{arr}}^{[k]}(m)$, which can

[2] In this work, each user represents one delay-constrained application, e.g., video streaming, video conferencing, etc. A physical user/device can simultaneously run multiple delay-constrained applications and thus host multiple users.

[3] After scheduling which flow to transmit, one also needs to choose which packet of the selected flow to transmit if there are multiple packets in the current queue. However, as one can show by a realization-based argument, it is optimal to always transmit the packet of the selected flow that is of the earliest deadline. If there is no packet of the selected flow in the AP's data queue, the AP just remains idle (or equivalently transmits nothing), which will not contribute to the timely throughput. In this work, we assume that the AP always chooses one flow to transmit while implicitly allowing the AP to remain idle when the queue of the chosen flow is empty.

[4] We slightly abuse the notation and still call the packet arriving at $t_{\text{arr}}^{[k]}(m)$ the m-th packet, even though on average only $(m-1)\mathsf{B}_k$ out of the first $(m-1)$ packets have actually "arrived."

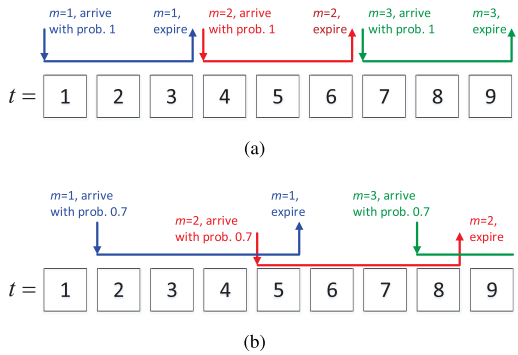


Fig. 1. The illustration of two arrival and expiration (A&E) profiles. (a) $(\mathsf{offset}_1, \mathsf{prd}_1, \mathsf{D}_1, \mathsf{B}_1) = (0, 3, 3, 1)$. (b) $(\mathsf{offset}_2, \mathsf{prd}_2, \mathsf{D}_2, \mathsf{B}_2) = (1, 3, 4, 0.7)$.

be computed as

$$t_{\text{arr}}^{[k]}(m) = \mathsf{offset}_k + (m-1)\mathsf{prd}_k + 1. \tag{1}$$

The m-th packet arrives with probability B_k. If it indeed arrives, it expires after D_k slots, and the expiration time is denoted as $t_{\text{exp}}^{[k]}(m)$, which can be computed as

$$t_{\text{exp}}^{[k]}(m) = t_{\text{arr}}^{[k]}(m) + \mathsf{D}_k. \tag{2}$$

The expired packets are removed from the system as they are no longer useful to the application.

An illustration of two A&E profiles is provided in Fig. 1. For example, the first flow-1 packet will always arrive at time 1 since $\mathsf{offset}_1 = 0$ and $\mathsf{B}_1 = 1$ and will expire at time 4. The second flow-2 packet will arrive at time 5 with probability 0.7 and expire at time 9.

We then define the *traffic pattern* as the collection of A&E profiles of all K flows. We remark that our traffic model is quite general because it captures not only the special frame-synchronized traffic pattern, but also other practical traffic patterns, as shown in the following.

The frame-synchronized traffic pattern can be captured by our traffic model as

$$(\mathsf{offset}_k, \mathsf{prd}_k, \mathsf{D}_k, \mathsf{B}_k) = (0, T, T, 1), \quad \forall k \in [1, K]$$

where T is called the *frame length*. As we can see, all K flows start at slot 1 and have the same arrival period T, and the same deadline T. Thus, every T slots, all flows have a packet arrival simultaneously, and all these packets will expire simultaneously after T slots. All three fundamental problems in Sec. I-C have been solved by [2] and [3] for this special traffic pattern.

However, the frame-synchronized traffic pattern cannot model many important practical scenarios. For example, consider a typical mobile video conferencing scenario. Suppose that packets arrive every 20ms with a hard delay of 200ms.[5] Since the delay is larger than the period, such flow cannot be modeled by the frame-synchornized traffic pattern. However, our traffic model can capture such traffic profile: if we assume that each slot spans 20ms and the first packet arrives at time 1, the A&E profile would be $(\mathsf{offset}, \mathsf{prd}, \mathsf{D}, \mathsf{B}) = (0, 1, 10, 1)$.

[5]If we assume that every packet has 1000 bytes (1kB), such arriving process means a sending rate of 400kbps. All settings, including packet size, sending rate and delay, are in line with practical video conferencing systems [22].

Note that our traffic pattern also suggests that we do not need an infinite-size data queue in the AP. Since all expired flow-k packets will be removed from the system, there exist at most $\left\lceil \frac{\mathsf{D}_k}{\mathsf{prd}_k} \right\rceil$ flow-k packets in the data queue of the AP. The total number of packets in the queue (from all K flows) is thus at most $\sum_{k=1}^{K} \left\lceil \frac{\mathsf{D}_k}{\mathsf{prd}_k} \right\rceil$. To avoid overflow, we therefore require that the data queue of the AP can at least hold $\sum_{k=1}^{K} \left\lceil \frac{\mathsf{D}_k}{\mathsf{prd}_k} \right\rceil$ packets. Again consider a video-conferencing scenario with ten flows where each flow's packets arrive every 20ms with a hard delay of 200ms. If every packet has a size of 1000 bytes (1kB), then the minimal data queue size requirement is $1\text{kB} \times 10 \times \left\lceil \frac{200\text{ms}}{20\text{ms}} \right\rceil = 100\text{kB}$.

Remark: Although our work is described only for the periodic-i.i.d. traffic patterns, the same principle can be readily extended to the much more general cyclostationary Markovian arrivals with observable states. Moreover, although we assume that any flow has at most one packet arrival in each period, our work can be generalized to the case that a flow may have multiple packet arrivals in a batch [23]. Our analysis can also be generalized to the case that different flows could have different packet lengths by treating a large packet as multiple sub-packets of the same length.

C. The Objective

The timely throughput R_k of flow k is defined as

$$R_k \triangleq \liminf_{\mathsf{T}\to\infty} \frac{\mathsf{E}\{\#\text{ of flow-}k\text{ pkts delivered before expiration in }[1,\mathsf{T}]\}}{\mathsf{T}}, \tag{3}$$

which computes the long-term average number of flow-k packets delivered before expiration per slot. Obviously, R_k depends on how to schedule the links/flows for $t = 1$ to ∞.

As we introduced in Sec. I-C, our objective is to solve the following three fundamental problems.

Capacity Region Problem: Characterize the capacity region,

$$\mathcal{R} \triangleq \{\vec{R} = (R_1, R_2, \cdots, R_K) | \text{ there exists a scheduling policy achieving timely throughput } R_k, \forall k \in [1, K]\} \tag{4}$$

NUM Problem: Design scheduling policies such that the resulting timely throughput vector solves the NUM problem,

$$(\mathbf{P1}) \quad \max_{\vec{R}\in\mathcal{R}} \sum_{k=1}^{K} U_k(R_k)$$

where $U_k(\cdot)$ is the utility function for flow k, which is assumed to be increasing, concave, and continuously differentiable.

Feasibility-Optimal Scheduling Policy Design Problem: Design one scheduling policy, so that for any feasible timely throughput vector $\vec{R} = (R_1, R_2, \cdots, R_K) \in \mathcal{R}$, the achieved flow-$k$ timely throughput under this policy is at least R_k for any $k \in [1, K]$.

D. Complexity Hardness

It is valuable to first examine the computational complexity of our problems. Since only the NUM problem $(\mathbf{P1})$ is a well-defined optimization problem, we show its hardness.

Theorem 1: $(\mathbf{P1})$ is co-NP-hard in the strong sense.

Proof: Please see Appendix A in the supplementary materials. ■

This shows that it is computationally prohibitive to get the exact solution unless P=co-NP. Note that it only shows the hardness but does not suggest any exact solution to solve $(\mathbf{P1})$.

In next two sections (Sec. IV and Sec. V), we will propose an exact solution based on MDP to $(\mathbf{P1})$ (and also to capacity region problem and feasibility-optimal scheduling policy design problem) with exponential complexity, meaning that all these three problems are in principle solvable. Theorem 1 shows that this could be the best that we can achieve if we desire an exact solution. To address the complexity issue, we must sacrifice the optimality. In Sec. VI, we therefore propose some heuristic solutions with much lower complexity.

IV. An Infinite-Dim. Infinite-Horizon MDP

This section explains how to cast our timely wireless flow problem as an infinite-dimension infinite-horizon multi-reward MDP problem. In Sec. V, we will further simplify the infinite-size MDP and characterize the complete timely capacity region and propose a scheduling policy that is feasibility-optimal and maximizes network utility.

An MDP problem [24], [25] can be described in many different forms. The MDP used in this work is described by a tuple $(\mathcal{S}, \{\mathcal{A}_s : s \in \mathcal{S}\}, \{P_t\}, \{r_k\})$ where $\mathcal{S}$ is the state space, $\mathcal{A}_s$ is the set of possible actions when the state is $s \in \mathcal{S}$, and P_t is the transition probabilities in time t:

$$P_t(S_{t+1} = s' | S_t = s, A_t = a), \forall t, \forall s, s' \in \mathcal{S}, \forall a \in \mathcal{A}_s, \quad (5)$$

and r_k is the flow-k reward function, i.e., $r_k(s, a)$ denotes the per-slot (additive) flow-k reward of taking the action $A_t = a$ when the system state is $S_t = s$. In our problem, since we should characterize the capacity region in terms of all K flows' timely throughput, we thus define K reward functions. This is called an MDP with multiple rewards [25]. We now describe how the timely wireless flow problem can be cast as an MDP by describing the corresponding $(\mathcal{S}, \{\mathcal{A}_s : s \in \mathcal{S}\}, \{P_t\}, \{r_k\})$.

Definition of the State: We define the (network) state S_t of the MDP as the *snap-shot* of all the network queues at time t. More specifically, define

$$S_t \triangleq (S_t^1, S_t^2, \cdots, S_t^K),$$

where S_t^k, the state of flow k at time t, is the collection of all non-expired flow-k packets in the AP's queue.

For example, suppose that there are only $K = 2$ flows with the corresponding A&E profiles depicted in Figs. 1(a) and 1(b), respectively. Then a possible network state at slot 8 is illustrated in Fig. 2(a). Specifically, for flow 1, at slot 8, packets $m = 1$ and $m = 2$ have expired and packet $m = 3$ has arrived at the AP. If the packet $m = 3$ has not been delivered successfully, it will remain in the queue and the state of flow 1 is $S_8^1 = \{X_3\}$. For flow 2, at slot 8, packet $m = 1$ has expired (no matter whether it showed up at the AP or not), and thus it does not appear in the queue. Packets $m = 2$ and $m = 3$ could have arrived at the AP. Suppose that these two packets have not been delivered successfully. The state of flow 2 is thus $S_8^2 = \{Y_2, Y_3\}$. The network state at slot 8 is $S_8 = (S_8^1, S_8^2) = (\{X_3\}, \{Y_2, Y_3\})$. Clearly, this is just one of many possibilities. Fig. 2(b) depicts another possible network state $S_9 = (S_9^1, S_9^2) = (\emptyset, \{Y_3\})$ at slot 9.

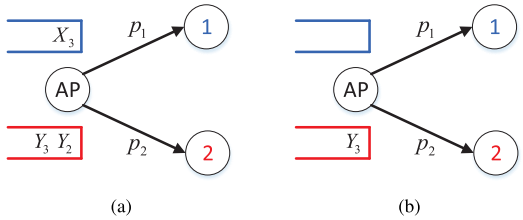


Fig. 2. Two examples for network states in slots 8 and 9, respectively. (a) $S_8 = (\{X_3\}, \{Y_2, Y_3\})$. (b) $S_9 = (\emptyset, \{Y_3\})$.

By enumerating all possible network states, we can explicitly construct the state space $\mathcal{S}$.

Definition of the Action: As mentioned in Sec. III-A, an action A_t represents the selection of which flow to transmit in time t. After selecting the flow, say flow k, the AP will transmit the oldest flow-k packet if there exist packet(s) in the data queue or remain idle otherwise. For example, if the network state at slot 8 is as Fig. 2(a), then there are 2 possible actions:

- Action 1: schedule link 1 (and then transmit the oldest packet of flow 1, which is X_3);
- Action 2: schedule link 2 (and then transmit the oldest packet of flow 2, which is Y_2).

One can quickly see that there are at most K actions for any state s. Even though not all K actions will contribute to timely throughput, for ease of exposition, in this paper we denote the action set for any state s as $\mathcal{A}_s = \{1, 2, \cdots, K\}$ and then simply write $\mathcal{A} = \{1, 2, \cdots, K\}$ by omitting the subscript s. Thus we get the action space $\mathcal{A} = \{1, 2, \cdots, K\}$.

Definition of the Transition Probabilities: We observe that the transition probability P_t from $S_t = s$ to $S_{t+1} = s'$ if taking action $A_t = a$ in slot t depends on (i) the channel success probabilities $\{p_k : k = 1, \cdots, K\}$, and (ii) the arrival and expiration events at the end of time t (or equivalently at the beginning of time $(t + 1)$). For example, at slot $(t + 1)$, some packet may be successfully delivered in time t, some old packets may expire and no longer remain in the queue, and some new packets may arrive, all of which will affect the network state S_{t+1}. By carefully examining (i) and (ii), we can explicitly construct the transition probability P_t in (5) for all t, s, s' and a. For example, considering the scenario in Fig. 2, we have

$$P_8(S_9 = (\emptyset, \{Y_3\}) | S_8 = (\{X_3\}, \{Y_2, Y_3\}), A_8 = \text{Action 1}) = p_1.$$

The reason is as follows. When the AP takes "Action 1: schedule link 1 (and transmit packet X_3)" in slot 8, if the transmission is successful, then X_3 will arrive at user 1 and will thus be removed from the queue. At the same time, since Y_2 will always expire at slot 9, it will also be removed from the queue. The network state at slot 9 thus becomes $S_9 = (\emptyset, \{Y_3\})$. The probability of this transition is thus p_1.

Note that since the packet arrival/expiration event depends on the time index t, the transition probabilities are time-inhomogeneous.

Definition of the Reward: In our problem, we care about the timely throughput of all K flows. Thus, we associate K reward functions for each state $s \in \mathcal{S}$ and action $a \in \mathcal{A}$ [25].

More specifically, for any flow $k \in [1, K]$, we define a reward function

$$r_k(s,a) \triangleq p_k \cdot 1_{\{\text{a flow-}k \text{ pkt is transmitted under state } s \text{ \& action } a\}}. \quad (6)$$

The indicator function $1_{\{\cdot\}}$ returns value 1 if the action a schedules a flow-k packet and the corresponding queue, specified in s, is not empty, and returns 0 otherwise. Notation p_k is the probability that the scheduled packet is successfully delivered. Eq. (6) calculates the expected value of the flow-k contribution for a given (s, a). Note that since our definition of state s only keeps those *unexpired* packets, any successful transmission is always unexpired and will contribute to $r_k(s, a)$.

For example, at slot 8, if $S_8 = (\{X_3\}, \{Y_2, Y_3\})$ (see Fig. 2(a)) and A_8 is "Action 1: schedule link 1 (and transmit packet X_3)", then the respective flow-1/flow-2 rewards are

$$r_1(S_8, A_8) = p_1, \quad r_2(S_8, A_8) = 0,$$

MDP-based Equivalence: From our MDP formulation, we can see that the timely throughput of any flow k is exactly the flow-k (long-term) average reward, i.e.,

$$R_k = \liminf_{\mathsf{T}\to\infty} \frac{\sum_{t=1}^{\mathsf{T}} \mathsf{E}\{r_k(S_t, A_t)\}}{\mathsf{T}}. \quad (7)$$

The capacity region $\mathcal{R}$ defined in (4) can then be rewritten as the following reward region of our MDP formulation, i.e.,

$$\mathcal{R} = \{\vec{R} = (R_1, R_2, \cdots, R_K) | \text{ there exists a scheduling policy of the MDP achieving average reward } R_k, \forall k\}$$

It is straightforward to verify that the NUM problem and the feasibility-optimal scheduling policy design problem can also be rewritten as an MDP-based formulation in a similar manner.

Remark: Note that the essence/difficulty of the timely throughput problem is that when we schedule a particular flow k at time t, the remaining packets in the queues are getting "older" and some may even expire. Therefore, the decision of sending which flow not only affects the instantaneous "reward" in time t, but it will also change the subsequent network state at time $(t + 1)$. The effect of a decision at time t can even propagate over multiple time slots, which makes it difficult to find the optimal solution. Such a phenomenon is captured naturally by our new MDP formulation where the action A_t not only affects $r_k(S_t, A_t)(\forall k \in [1, K])$ but also affects the next network state S_{t+1} through the transition probability (5).

V. Simplification and Optimal Solutions

The first contribution of this work is to observe that the timely wireless flow problem is fundamentally an MDP problem. However, our MDP formulation in Sec. IV is difficult to handle, because it has an infinite number of states, and it is time-inhomogeneous with infinite horizon. In this section, we demonstrate two simplification methods to reduce the state space $\mathcal{S}$, and address the time-inhomogeneity by observing that our MDP is actually *almost cyclostationary*. We then prove that the timely capacity region is a convex polygon which is characterized by a finite set of linear constraints. Our analytical results also allow us to design a scheduling policy, by solving a convex program, which is feasibility-optimal and maximizes network utility.

A. Reduce the State Space

Define the *lead time* (see [26] for further discussion) of the m-th flow-k packet at slot $\tau \in [t_{\text{arr}}^{[k]}(m), t_{\text{exp}}^{[k]}(m) - 1]$ as

$$t_{\text{lead}}^{[k]}(m) = t_{\text{exp}}^{[k]}(m) - \tau. \quad (8)$$

Clearly, we have $t_{\text{lead}}^{[k]}(m) \in [1, \mathsf{D}_k]$, which can be interpreted as the remaining time before expiration. Moreover, at any slot t, there exists at most one flow-k packet in the queue whose lead time is τ, for any $\tau \in [1, \mathsf{D}_k]$. Therefore, the state of flow k, which was originally defined as the set of unexpired flow-k packets in the queue, can be rewritten as an equivalent binary string, $S_t^k \triangleq l_1^k l_2^k, \cdots l_{\mathsf{D}_k}^k$ where

$$l_i^k = \begin{cases} 1, \text{ if } \exists \text{ a flow-}k \text{ packet with lead time } i \text{ at } t; \\ 0, \text{ otherwise.} \end{cases}$$

For example, for flow 2 in Fig. 2(a), the state at slot 8 is $S_8^2 = 1001$. The reason is that both Y_2 and Y_3 are in the queue. The lead time of Y_2 is $t_{\text{lead}}^{[2]}(2) = t_{\text{exp}}^{[2]}(2) - 8 = 1$ and the lead time of Y_3 is $t_{\text{lead}}^{[2]}(3) = t_{\text{exp}}^{[2]}(3) - 8 = 4$. At slot 9, the state becomes $S_9^2 = 0010$ since only Y_3 remains and its lead time is now changed to $t_{\text{lead}}^{[2]}(3) = t_{\text{exp}}^{[2]}(3) - 9 = 3$. For Fig. 2, similar reasoning can be used to show $S_8^1 = 010$ and $S_9^1 = 000$. The network state thus becomes $S_8 = (S_8^1, S_8^2) = (010, 1001)$ and $S_9 = (S_9^1, S_9^2) = (000, 0010)$.

The new binary-string-based representation is equivalent to the original set-based representation since for any time t, we can use (2) and (8) to infer whether the m-th flow-k packet is in the queue or not.

Since each state s^k is a binary string of length D_k, if we denote $\mathcal{S}^k$ as the set of all possible s^k, then we have $|\mathcal{S}^k| \leq 2^{\mathsf{D}_k}$. The total number of network states is thus

$$|\mathcal{S}| = |\mathcal{S}^1| \cdot |\mathcal{S}^2| \cdot \cdots \cdot |\mathcal{S}^K| \leq 2^{\mathsf{D}_1 + \mathsf{D}_2 + \cdots + \mathsf{D}_K} < \infty. \quad (9)$$

The new lead-time-based state space $\mathcal{S}$ is therefore bounded. Note that even with the lead-time-based $\mathcal{S}$, the MDP is still of infinite horizon.

The reason that (9) is only an upper bound is that for any given traffic pattern, some binary strings do not represent any state. This fact can be used to further reduce the state space for some special traffic patterns. For example, for the *frame-synchronized traffic pattern* in Sec. III-B, at each time t, the flow-k state $S_t^k = l_1^k l_2^k, \cdots l_T^k$, where

$$\begin{cases} l_i^k = 0, \forall i \in [1, T], & \text{if no flow-}k \text{ packet;} \\ l_{g(t)}^k = 1, l_i^k = 0, \forall i \neq g(t), & \text{if } \exists \text{ a flow-}k \text{ packet.} \end{cases} \quad (10)$$

and $g(t) = T - ((t-1) \mod T)$. Since there are only two possible states for each flow-k at any slot, we can perform a "lossless compression" and use $S_t^k = 0$ to represent the first case, and use $S_t^k = 1$ for the second case in (10). In this way, the state space is further reduced and we have $|\mathcal{S}^k| = 2$. The number of network states is then equal to $|\mathcal{S}| = 2^K$, much smaller than the upper bound (9). A similar compression method can be used to reduce the bound to $|\mathcal{S}| \leq 2^{\sum_{k=1}^K \lceil \mathsf{D}_k / \mathsf{prd}_k \rceil}$ when the traffic pattern is not frame-synchronized.

B. Reduce the Horizon

With the simplification in Sec. V-A, the new MDP is of finite dimension now. However, it is still time inhomogeneous with infinite horizon, which makes it difficult to apply the existing techniques that solve time-homogeneous infinite-horizon average-reward MDP [24]. To circumvent this difficulty, we make another critical observation:

Lemma 1: Using the new network state representation, the transition probabilities P_t are *almost cyclostationary*. Namely, define

$$\mathsf{Prd} \triangleq \text{Least.Common.Multiple}(\mathsf{prd}_1, \mathsf{prd}_2, \cdots, \mathsf{prd}_K),$$

i.e., Prd is is the smallest positive integer that is divisible by all prd_k, and choose L as a constant positive integer such that

$$L \cdot \mathsf{Prd} \geq \max_{k\in[1,K]} (\mathsf{offset}_k + \mathsf{D}_k).$$

Then, for any $\tau \in [1, \mathsf{Prd}], l \geq L$, the transition probability $P_{l\cdot\mathsf{Prd}+\tau}$ for slot $t = l \cdot \mathsf{Prd} + \tau$ is identical to the transition probability $P_{(l+1)\mathsf{Prd}+\tau}$ for slot $t' = (l+1) \cdot \mathsf{Prd} + \tau$.

Proof: Please see Appendix B in the supplementary materials. ∎

The reason behind is that when $l \geq L$, then at time $t = (l\cdot\mathsf{Prd}+\tau)$, the first packet of flow k has expired for all k. Therefore, all K flows have left their transient "initialization phase" and entered their "steady state". Also, since Prd is the least common multiple of all prd_k, then after every Prd time slots the arrival and expiration patterns of all K flows will repeat themselves. Since the inhomogeneity of the transition probability P_t is only caused by different arrival and expiration events for each time t, the transition probability P_t will also repeat itself after every Prd time slots since the traffic pattern are periodic. For future reference, we define $\tau_{\text{trans}} = L \cdot \mathsf{Prd}$ and call the time interval $[1, \tau_{\text{trans}}]$ the *transient duration*.

The fact that the transition probability P_t is almost periodic prompts the following intuition. If we can focus on the those time slots after the transient duration, then the network controller faces a periodic environment. As a result, in terms of finding the asymptotic average reward, we can consider a single period instead of the infinite horizon from 1 to ∞, as long as the period of interest is beyond the transient duration. For example, one such period could be the interval $[L \cdot \mathsf{Prd}+1, (L+1)\mathsf{Prd}]$. We will make this intuition rigorous in the next subsection.

C. Optimal Solutions

In this subsection, we make use of the two simplification methods in Sec. V-A and Sec. V-B to characterize the capacity region, based on which we further propose a scheduling policy that is feasibility-optimal and maximizes network utility.

Towards that end, we first define the randomized almost cyclostationary (RAC) policy as follows.

Definition 1: A scheduling policy π is *randomized* if for every time t under state $S_t = s$, the action A_t is chosen randomly according to a probability mass function

$$\mathsf{Prob}_{A_t|S_t}(a|s) = \mathsf{Prob}(A_t = a|S_t = s), \quad \forall a \in \mathcal{A}$$

For our given MDP, a randomized policy π is *almost cyclostationary* if the following two conditions hold: (i)

$$\mathsf{Prob}_{A_t|S_t}(a|s) = \mathsf{Prob}_{A_{t+\mathsf{Prd}}|S_{t+\mathsf{Prd}}}(a|s),$$

for all $s \in \mathcal{S}, a \in \mathcal{A}, t > \tau_{\text{trans}}$, that is, for all $t > \tau_{\text{trans}}$, the conditional probabilities repeat themselves after Prd slots, and (ii) the random process[6] of the MDP state after time τ_{trans}, $\{S_{\tau_{\text{trans}}+t} : \forall t \geq 1\}$, is cyclostationary with period Prd.

We now present our main result.

Theorem 2: (i) Any feasible timely throughput vector $\vec{R} \in \mathcal{R}$ can be achieved by an RAC policy.

(ii) The capacity region $\mathcal{R}$ can be characterized by the following convex polygon,

$$\mathcal{R} = \{\vec{R} = (R_1, R_2, \cdots, R_K) | \text{ there exists an } \vec{x}$$
$$\text{such that the following conditions (11a) – (11e) hold}\}$$

where the conditions are

$$\sum_{a\in\mathcal{A}} x_{t+1}(s', a) = \sum_{s\in\mathcal{S}}\sum_{a\in\mathcal{A}} P_t(s'|s, a)x_t(s, a), \quad \forall s' \in \mathcal{S}, \quad t \in [T_1, T_2 - 1] \tag{11a}$$

$$\sum_{a\in\mathcal{A}} x_{T_1}(s', a) = \sum_{s\in\mathcal{S}}\sum_{a\in\mathcal{A}} P_{T_2}(s'|s, a)x_{T_2}(s, a), \quad \forall s' \in \mathcal{S} \tag{11b}$$

$$R_k \leq \sum_{t=T_1}^{T_2}\sum_{s\in\mathcal{S}}\sum_{a\in\mathcal{A}} \frac{r_k(s, a)x_t(s, a)}{\mathsf{Prd}}, \quad \forall k\in[1, K] \tag{11c}$$

$$\sum_{s\in\mathcal{S}}\sum_{a\in\mathcal{A}} x_t(s, a) = 1, \quad \forall t \in [T_1, T_2] \tag{11d}$$

$$\vec{x} \geq 0, \quad \vec{R} \geq 0 \tag{11e}$$

with $[T_1, T_2] \triangleq [L \cdot \mathsf{Prd} + 1, (L + 1)\mathsf{Prd}]$.

(iii) For any $\vec{x}$ and $\vec{R} = (R_1, R_2, \cdots, R_K)$ that satisfy (11a) – (11e), the RAC policy[7] with conditional probability,

$$\begin{cases} \mathsf{Prob}_{A_t|S_t}(a|s) = \frac{x_t(s,a)}{\sum_{a'\in\mathcal{A}} x_t(s,a')}, & \forall t\in[T_1, T_2]; \\ \mathsf{Prob}_{A_t|S_t}(a|s) = \mathsf{Prob}_{A_{t-\mathsf{Prd}}|S_{t-\mathsf{Prd}}}(a|s), & \forall t > T_2, \end{cases} \tag{12}$$

and state distribution,

$$\mathsf{Prob}_{S_t}(s) = \sum_{a\in\mathcal{A}} x_t(s, a), \quad \forall t \in [T_1, T_2]. \tag{13}$$

achieves the timely throughput R_k, for any $k \in [1, K]$.

Proof: Please see Appendix C in the supplementary materials. ∎

Note that here $[T_1, T_2] = [L \cdot \mathsf{Prd} + 1, (L + 1)\mathsf{Prd}]$ is the period that we mentioned in Sec. V-B.

Part (i) of Theorem 2 shows that RAC polices achieve any feasible timely throughput vector. This greatly reduces the policy space since, if we ignore the transient phase, an RAC scheme can be specified by the conditional probability $\mathsf{Prob}_{A_t|S_t}(a|s)$ and the resulting state distribution $\mathsf{Prob}_{S_t}(s)$ for one period of Prd slots. The design space is now bounded

[6]Condition (i) requires that the way we make the decision is periodic and condition (ii) requires that the resulting state distribution is periodic. Although condition (i) generally means that condition (ii) is satisfied asymptotically when $t \to \infty$, here we require condition (ii) to be satisfied for small finite t.

[7]For ease of exposition, we omit the design of the transient state $t \leq \tau_{\text{trans}} = T_1 - 1$. The complete RAC design can be found in the proof, i.e., Appendix C in the supplementary materials.

and the resulting RAC policy can fully solve an infinite-horizon MDP problem. This justifies our intuition in Sec. V-B.

Part (ii) of Theorem 2 shows that the complete timely capacity region can be characterized by a finite-size convex polygon in (11). The intuition of (11) is as follows. The variable $x_t(s,a) = \mathsf{Prob}_{S_t,A_t}(s,a)$ is the probability that the system state is s and the action is a at slot t under an RAC policy, which is why the total sum of $x_t(s,a)$ is 1 (see (11d)). The right-hand side of (11c) computes the average reward of flow k under such an RAC policy. Eq. (11a) is the consistency condition for time $[T_1, T_2-1]$. The left-hand side of (11a) is the marginal probability $\mathsf{Prob}_{S_{t+1}}(s')$. The right-hand side of (11a) starts from the joint distribution Prob_{S_t,A_t} and uses the transition probability $P_t(S_{t+1}|S_t,A_t)$ to compute $\mathsf{Prob}_{S_{t+1}}(s')$. Similarly, (11b) is the consistency condition from time T_2 back to T_1 since we require the periodicity condition $\mathsf{Prob}_{S_{T_2+1}}(s') = \mathsf{Prob}_{S_{T_1}}(s')$.

Part (iii) of Theorem 2 gives the corresponding RAC policy to achieve any feasible timely throughput vector based on the solution of (11).

Theorem 2 solves the first fundamental problem in Sec. I-C, i.e., characterizing the capacity region. To the best of our knowledge, this is the first timely capacity characterization for general traffic patterns.

Based on the capacity region in (11), we can optimally solve the other two fundamental problems proposed in Sec. I-C. More specifically, the NUM problem $(\mathbf{P1})$ is equivalent to the following convex one:

$$(\mathbf{P2}) \quad \max \ \sum_{k=1}^{K} U_k(R_k) \quad \text{s.t.} \ \ (11a)-(11e) \quad \text{var.} \ \ \vec{x}, \vec{R}$$

Similarly, to design a feasibility-optimal scheduling policy, we can solve the following linear programming (LP),[8]

$$(\mathbf{P3}) \quad \max \ \ 1 \quad \text{s.t.} \ \ (11a)-(11e) \quad \text{var.} \ \ \vec{x}$$

Note that in $(\mathbf{P2})$, the achieved timely throughput R_k is an optimization variable while in $(\mathbf{P3})$, R_k is given as an input.

RAC (Optimal) Scheduling Policy: Once we solve $(\mathbf{P2})$ or $(\mathbf{P3})$, we obtain the optimal state-action frequency $\vec{x}^*$. We then replace $\vec{x}$ in (12) and (13) by $\vec{x}^*$. This gives the optimal RAC policy that is feasibility-optimal and maximizes network utility, as shown in part (iii) of Theorem 2.

Remark: The existing elegant framework in [2] and [3] is based on the frame-synchronized setting. In contrast, our framework applies to general traffic patterns. Further, the existing framework is based on first deriving an idle-time-based outer bound. Then a largest-deficit-first (LDF) scheme is proposed that attains any point within the outer bound. However, for general settings, how to find a tight outer bound is highly non-trivial and remains open as of today. Instead of finding an outer bound and an achievability scheme separately as in [2], our approach is fundamentally different. By proposing a new MDP framework, we first establish that any optimal point can always be achieved by an RAC policy. Then we search for the optimal RAC by solving a finite-size convex program. The solution is thus simultaneously an outer bound (no scheme can do better) and an inner bound (as it explicitly leads to an optimal design). In the broadest sense, our approach can be viewed as directly finding the *maximum flow* instead of indirectly finding the *minimum cut*.

[8]We write $(\mathbf{P3})$ as a maximization problem for consistence, but we should note that in $(\mathbf{P3})$, we only need to find a solution $\vec{x}$ such that (11a) – (11e) hold.

VI. Low-Complexity Heuristic Solutions

Thus far we have characterized the timely capacity region and designed the corresponding optimal scheduling policy that is feasibility-optimal and maximizes network utility. However, our capacity region characterization (11) suffers from high complexity, which involves $O(\mathsf{Prd} \cdot K \cdot 2^{\sum_{k=1}^{K} \mathsf{D}_k})$ variables and constraints. This makes it less appealing for practical implementation.[9]

In this section, we address the complexity issue by proposing two low-complexity heuristics, both of which are inspired by our MDP-based formulation. In the first one in Sec. VI-A, we derive a computationally-efficient outer bound for the capacity region in (11), based on which we further propose a heuristic scheduling policy to optimize network utility or support feasible timely throughput vectors. In the second one in Sec. VI-B, we improve the LDF scheduling policy [2] to support feasible timely throughput vectors by combining both deficit information and *urgency* information. Both heuristics are based on the insights derived in Theorem 2 and achieve good performance in the numerical evaluation, as shown later in Sec. VII.

A. RAC Approximation

In our original system, the AP schedules one and only one flow at each slot. We can equivalently convert this 1-to-many system to K parallel 1-to-1 systems $(\mathrm{src}_k, \mathrm{dst}_k)$ for each k in the following way. We allow each src_k to make their own decision $A_t^k \in \{1, \cdots, K\}$ and src_k transmits only when $A_t^k = k$ and remains idle whenever $A_t^k \neq k$. We further impose that $A_t^{k_1} = A_t^{k_2}$ for any two flows k_1 and k_2. This ensures that even though we have K parallel 1-to-1 systems, their decisions are strictly synchronized, and only one of them can be active in any time t. Therefore, the K parallel 1-to-1 systems are equivalent to the original 1-to-many AP network.

Now we relax this *synchronized action constraint* $A_t^{k_1} = A_t^{k_2}$ to a *common scheduling frequency constraint* $\mathsf{Prob}(A_t^{k_1} = a) = \mathsf{Prob}(A_t^{k_2} = a) \triangleq z_t(a)$. Namely, for each parallel system k, we use $z_t^k(s^k, a)$ to denote the probability that flow k is in state s^k and the action is a at slot t. The *common scheduling frequency constraint* imposes that

[9]The main complexity is due to the computation of the optimal $x_t^*(s,a)$ in $(\mathbf{P2})$ or $(\mathbf{P3})$. Once $x_t^*(s,a)$ is known, the actual RAC scheduler is simple and involves generating random variables with probability distribution in (12). Therefore, for a relatively stable system, the optimal RAC policy can still be implemented by computing the optimal $x_t^*(s,a)$ offline. For references, using off-the-shelf solvers, $(\mathbf{P2})$ or $(\mathbf{P3})$ can be solved in a few seconds with $K = 7$ flows and moderate D_k values for linear utility functions.

the K parallel systems must share a common scheduling frequency, i.e.,

$$\sum_{s^k\in\mathcal{S}^k} z_t^k(s^k,a)=z_t(a),\quad \forall k\in[1,K],\ t,\ a\in\mathcal{A}. \quad (14)$$

Clearly, the sample-path-based *synchronized action constraint* $A_t^{k_1}=A_t^{k_2}$ implies the distribution-based *common scheduling frequency constraint* (14). This motives us to define the following outer bound $\mathcal{R}^{\mathsf{outer}}$ of the capacity region $\mathcal{R}$ in (11).

$$\mathcal{R}^{\mathsf{outer}}\triangleq\{\vec{R}=(R_1,R_2,\cdots,R_K)|\text{ there exists an }\vec{z}\text{ such that the following conditions (15a)}-\text{(15f) hold}\}$$

where the conditions are

$$\sum_{a\in\mathcal{A}} z_{t+1}^k(\tilde{s}^k,a)=\sum_{s^k\in\mathcal{S}^k}\sum_{a\in\mathcal{A}} P_t^k(\tilde{s}^k|s^k,a)z_t^k(s^k,a),\quad \forall k\in[1,K],\ \tilde{s}^k\in\mathcal{S}^k,\ t\in[T_1,T_2-1] \quad (15a)$$

$$\sum_{a\in\mathcal{A}} z_{T_1}^k(\tilde{s}^k,a)=\sum_{s^k\in\mathcal{S}^k}\sum_{a\in\mathcal{A}} P_{T_2}^k(\tilde{s}^k|s^k,a)z_{T_2}^k(s^k,a),\quad \forall k\in[1,K],\ \tilde{s}^k\in\mathcal{S}^k \quad (15b)$$

$$R_k\le\sum_{t=T_1}^{T_2}\sum_{s^k\in\mathcal{S}^k}\sum_{a\in\mathcal{A}}\frac{r_k(s^k,a)z_t^k(s^k,a)}{\mathsf{Prd}},\quad \forall k\in[1,K] \quad (15c)$$

$$\sum_{s^k\in\mathcal{S}^k} z_t^k(s_k,a)=z_t(a),\quad \forall k\in[1,K],\ t\in[T_1,T_2] \quad (15d)$$

$$\sum_{s^k\in\mathcal{S}^k}\sum_{a\in\mathcal{A}} z_t^k(s^k,a)=1,\quad \forall k\in[1,K],\ t\in[T_1,T_2] \quad (15e)$$

$$\vec{z}\ge 0,\quad \vec{R}\ge 0 \quad (15f)$$

In (15), $P_t^k(\tilde{s}^k|s^k,a)$ is the transition probability from state s^k to state $\tilde{s}^k$ for flow k if taking action $A_t^k=a$ at slot t, $r_k(s^k,a)$ is the flow-k per-slot reward under state s^k and action a (defined similarly as (6)), and T_1 and T_2 are defined as the same in (11). One can see that the form of (15) is very close to that of (11) except that (15) deals with each 1-to-1 system separately and links them through the common scheduling frequency constraint (15d).

We can regard (15) as a relaxed version of (11). In return for the relaxation, we can handle it more efficiently since the state of each flow k is considered separately (rather than considered as a joint network state). The new complexity (in terms of number of variables and constraints) thus becomes,

$$O\left((2^{\mathsf{D}_1}+2^{\mathsf{D}_2}+\cdots+2^{\mathsf{D}_K})\cdot K\cdot\mathsf{Prd}\right).$$

This allows us to handle significantly large K, Prd and very reasonable practical D_k values. If we further use the lossless simplification method in (10), then the complexity can be further reduced to

$$O\left(\left(2^{\lceil\frac{\mathsf{D}_1}{\mathsf{prd}_1}\rceil}+2^{\lceil\frac{\mathsf{D}_2}{\mathsf{prd}_2}\rceil}+\cdots+2^{\lceil\frac{\mathsf{D}_K}{\mathsf{prd}_K}\rceil}\right)\cdot K\cdot\mathsf{Prd}\right), \quad (16)$$

which is quite manageable for almost all practical system parameters. If we aim to solve (15)[10] approximately rather than exactly, we can further *unwind* the arrival of packets from K flows and find an approximation solution of (15) with a complexity of $O((\sum_{k=1}^K\lceil\mathsf{D}_k/\mathsf{prd}_k\rceil)\cdot K\cdot\mathsf{Prd})$. Please see the details in Appendix E in the supplementary materials. We thus call $\mathcal{R}^{\mathsf{outer}}$ a fast outer bound of the capacity region $\mathcal{R}$.

Next we use the outer bound $\mathcal{R}^{\mathsf{outer}}$ to design a heuristic scheduling policy, called RAC-Approx, to either maximize the network utility or support feasible timely throughput vectors. More concretely, for the NUM problem, we solve the following convex program,

$$(\mathbf{P4})\quad \max\ \sum_{k=1}^K U_k(R_k)\quad \text{s.t. (15a)}-\text{(15f)}\quad \text{var. }\vec{z},\vec{R}$$

and for designing low-complexity scheduling policy for supporting feasible throughput vectors, we solve the following size-reduced LP with the timely throughput vector $\vec{R}$ as an input:

$$(\mathbf{P5})\quad \max\ 1\quad \text{s.t. (15a)}-\text{(15f)}\quad \text{var. }\vec{z}$$

We can regard $(\mathbf{P4})$ (resp. $(\mathbf{P5})$) as the relaxed problem of $(\mathbf{P2})$ (resp. $((\mathbf{P3}))$ with much lower complexity. We then use the optimal solution of $(\mathbf{P4})$ or $(\mathbf{P5})$, denoted by $\tilde{z}_t^k(s^k,a)$, to design the control probability of an RAC policy, i.e., we will replace (12) by a new formula.

RAC-Approx Scheduling Policy: At slot t, suppose that the system state is $S_t=(S_t^1,S_t^2,\cdots S_t^K)=(s^1,s^2,\cdots s^K)$, first compute the following conditional probability for each action $a\in\mathcal{A}$ and each flow $k\in[1,K]$,

$$\begin{cases}\mathsf{Prob}_{A_t|S_t^k}(a|s^k)=\frac{\tilde{z}_t^k(s^k,a)}{\sum_{a'\in\mathcal{A}}\tilde{z}_t^k(s^k,a')}, & \forall t\in[T_1,T_2];\\ \mathsf{Prob}_{A_t|S_t^k}(a|s^k)=\mathsf{Prob}_{A_{t-\mathsf{Prd}}|S_{t-\mathsf{Prd}}^k}(a^k|s), & \forall t>T_2.\end{cases} \quad (17)$$

and then select action a with probability

$$\frac{\prod_{k=1}^K\mathsf{Prob}_{A_t|S_t^k}(a|s^k)}{\sum_{a'\in\mathcal{A}}\prod_{k=1}^K\mathsf{Prob}_{A_t|S_t^k}(a'|s^k)}. \quad (18)$$

The intuition of (18) is as follows. Eq. (17) is the probability that the k-th parallel system will choose a specific action a. Since all parallel systems choose their actions independently, the numerator of (18) is the probability that all flows of the auxiliary K parallel systems choose the same action a. When all flows choose the same action a, we let the AP of our actual system take such action a. Note that it is possible that all flows choose a different action a'. By normalizing over the probability of all possible a' in the denominator of (18), it is as if we directly let the AP randomly choose an action a with probability (18).

[10]Precisely, we mean solving $(\mathbf{P4})$ or $(\mathbf{P5})$ with constraints (15) which will be mentioned soon.

Our RAC-Approx policy, using $(\mathbf{P4})$/$(\mathbf{P5})$, (17), and (18), is very efficient since all the computation can be performed on a per-flow base, as opposed to the network-wide computation in $(\mathbf{P2})$/$(\mathbf{P3})$ and (12). We further show the convergence result of our RAC-Approx policy under a mild condition.

Lemma 2: Suppose that the arrival probability is strictly less than 1, i.e., $\mathsf{B}_k < 1$, for any flow $k \in [1, K]$. Then the timely throughput of the RAC-Approx policy converges and the $\liminf$ in (3) can be replaced by $\lim$.

Proof: Please see Appendix D in the supplementary materials. ■

B. Deficit-Based Scheduling Algorithm

In this subsection, we propose a low-complexity heuristic deficit-based scheduling policy to support feasible timely throughput vectors. Our MDP formulation shows that the lead-time-based state representation is critical to finding the optimal solution. In the following, we show that by incorporating the concept of lead time, we can further improve the performance of the existing deficit-based policies.

In general, the flow-k deficit at slot t is the difference between the desired number of delivered flow-k packets[11] and the actual number of delivered flow-k packets up to slot t [2]. Intuitively, the AP should schedule the flow with largest deficit, which is the celebrated *Largest-Deficit-First* (LDF) scheduler. Hou *et al.* [2] proved that LDF is feasibility-optimal for the frame-synchronized traffic pattern. The reason why LDF is optimal in the frame-synchronized traffic pattern is that all non-expired packets are equally urgent because they have the same deadline. However, when different packets have different deadlines, they have different levels of urgency. Since the deficit does not contain any *urgency* information, the LDF policy is no longer optimal for general traffic patterns [16], [19].

To handle heterogeneous deadlines, [16] proposed the *Earliest-Positive-deficit-Deadline-First* (EPDF) scheduler. In EPDF, at any slot, the AP focuses on those flows with strictly positive deficit and among them selects the flow which has the earliest deadline. Unfortunately, when evaluated numerically, EPDF is strictly sub-optimal, see Sec. VII-C for more detailed discussion of the sub-optimality of EPDF.

Inspired by our lead-time-based MDP study, we propose the following *Lead-time-normalized-Largest-Deficit-First* (L-LDF) scheduler, which combines both deficit and urgency information.

L-LDF Scheduling Policy: Suppose flow-k timely throughput requirement is $q_k \in (0, 1]$. At each slot t, among all flows that currently have packets to send, the AP computes the *lead-time-normalized deficit* $\bar{d}_k(t)$ for each flow k:

$$\bar{d}_k(t) \triangleq \frac{d_k(t) \cdot p_k}{\text{smallest-lead-time}(k,t)},$$

where $d_k(t)$ is the flow-k deficit at slot t defined as,

$$d_k(t) \triangleq [d_k(t-1) - 1_{\{\text{a flow-}k\text{ packet is delivered at slot } t\}}]^+ + q_k,$$

with $d_k(0) \triangleq 0$, $[x]^+ \triangleq \max\{x, 0\}$, and smallest-lead-time (k,t) is the smallest lead time among all flow-k packets at slot t. Note that smallest-lead-time(k,t) is no smaller than 1 according to the definition of lead time in (8) and the remark right below the equation. Then, the AP selects the flow with the largest $\bar{d}_k(t)$.

Note that L-LDF collapses to the existing LDF in the frame-synchronized traffic pattern, since in that setting all non-expired packets at time t will have the same smallest lead time. However, the additional normalization according to the smallest lead time will better reflect the urgency of each individual flow for general traffic patterns.

[11]More specifically, the desired number of delivered flow-k packets up to t is $q_k t$, where q_k is the flow-k timely throughput requirement.

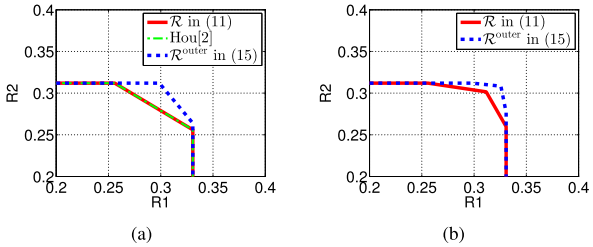


Fig. 3. Capacity regions of two examples in Sec. VII-A. (a) Two synchronized flows. (b) Two non-synchronized flows.

VII. Simulation

In this section, we demonstrate numerical performances of our solutions on characterizing timely capacity region, maximizing network utility, and supporting feasible timely throughput vectors.

A. Characterizing Capacity Region

Since the existing idle-time-based analysis [2] can only characterize the capacity region for the frame-synchronized traffic pattern, we also apply our MDP-based computation $\mathcal{R}$ in (11) to such a simple setting. Specifically, we consider the following frame-synchronized traffic pattern:

$$(\mathsf{offset}_1, \mathsf{prd}_1, \mathsf{D}_1, \mathsf{B}_1, p_1) = (0, 3, 3, 1, 0.8),$$
$$(\mathsf{offset}_2, \mathsf{prd}_2, \mathsf{D}_2, \mathsf{B}_2, p_2) = (0, 3, 3, 1, 0.6).$$

Fig. 3(a) shows the capacity region of this traffic pattern. As expected, both [2] and our MDP-based computation $\mathcal{R}$ in (11) successfully characterize the same capacity region.

Next we offset flow-2 by 2 slots. Namely, the two flows are non-synchronized now:

$$(\mathsf{offset}_1, \mathsf{prd}_1, \mathsf{D}_1, \mathsf{B}_1, p_1) = (0, 3, 3, 1, 0.8),$$
$$(\mathsf{offset}_2, \mathsf{prd}_2, \mathsf{D}_2, \mathsf{B}_2, p_2) = (2, 3, 3, 1, 0.6).$$

The idle-time-based analysis does not hold anymore. However, our MDP-based computation $\mathcal{R}$ in (11) can still characterize the capacity region, see Fig. 3(b), which contains three corner points, as opposed to only two corner points in Fig. 3(a). Such a phenomenon is observed for the first time in the literature.

In both Figs. 3(a) and 3(b), we also evaluate our fast outer bound $\mathcal{R}^{\text{outer}}$ in (15). We can see that empirically it is a reasonably tight outer bound of the capacity region. We further show the gap between our outer bound and the capacity region as follows. We consider the linear utility function $U_k(R_k) = w_k R_k$. For any particular weight vector $\vec{w} = (w_1, w_2, \cdots, w_K)$, we solve the RAC problem $(\mathbf{P2})$

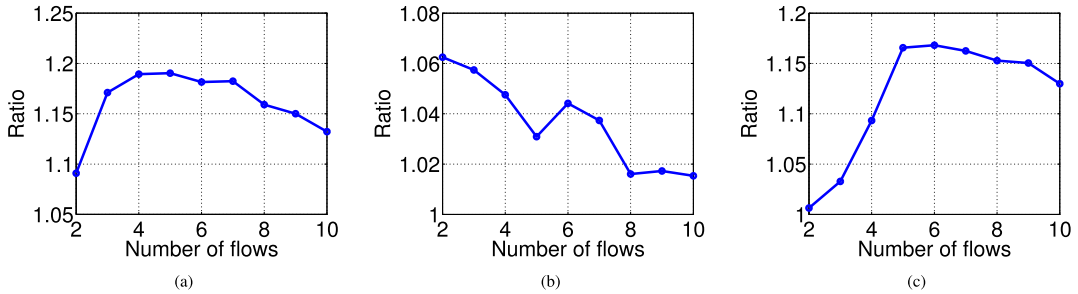


Fig. 4. Ratio of the outer bound to the capacity region, defined as $r^* \triangleq \max_{\vec{w}=(w_1,w_2,\cdots,w_K)\in\mathbb{R}_+^K} \frac{u_4^*(\vec{w})}{u_2^*(\vec{w})}$, where $u_2^*(\vec{w})$ (resp. $u_3^*(\vec{w})$) is the optimal value/utility of $(\mathbf{P2})$ (resp. $(\mathbf{P4})$) if taking utility function $U_k(R_k) = w_k R_k$ for all k. (a) Case 1: $(\mathsf{offset}_k, \mathsf{prd}_k, \mathsf{D}_k, \mathsf{B}_k, p_k) = (0, 2, 2, 0.5, 0.8)$ for all k. (b) Case 2: $(\mathsf{offset}_k, \mathsf{prd}_k, \mathsf{D}_k, \mathsf{B}_k, p_k) = (0, 3, 3, 1, p_k)$ for all k where $p_k = 0.8$ (resp. 0.6) for odd (resp. even) k. (c) Case 3: $(\mathsf{offset}_k, \mathsf{prd}_k, \mathsf{D}_k, \mathsf{B}_k, p_k) = (\mathsf{offset}_k, 3, 3, 0.5, 0.8)$ for all k where $\mathsf{offset}_k = 0$ (resp. 1) for odd (resp. even) k.

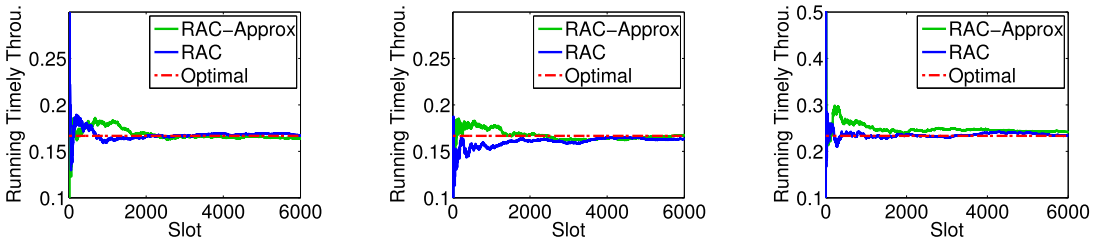


Fig. 5. Comparison of different scheduling policies for maximizing network utility for three flows in Sec. VII-B with utility functions, $U_1(R_1) = \log(R_1)$, $U_2(R_2) = \log(R_2)$, and $U_3(R_3) = \log(R_3)$. Both RAC and RAC-Approx scheduling policies converge to the optimal solution. (a) Flow 1. (b) Flow 2. (c) Flow 3.

and the RAC-Approx problem $(\mathbf{P3})$, respectively. We denote the optimal value/utility of $(\mathbf{P2})$ (resp. $(\mathbf{P4})$) as $u_2^*(\vec{w})$ (resp. $u_4^*(\vec{w})$). We then define $r(\vec{w}) \triangleq \frac{u_4^*(\vec{w})}{u_2^*(\vec{w})}$, which measures the gap between the outer bound and the capacity region *in the direction* $\vec{w}$. Now we define the *ratio of the outer bound to the capacity region* as $r^* \triangleq \max_{\vec{w}\in\mathbb{R}_+^K} r(\vec{w})$ where $\mathbb{R}_+^K$ is the set of all nonnegative directions, i.e., $\mathbb{R}_+^K \triangleq \{\vec{w} = (w_1, w_2, \cdots, w_K) : w_k \geq 0, \forall k \in [1, K]\}$. If $r^* = 1$, then the outer bound is exactly the capacity region, and smaller r^* means tighter outer bound. We thus use r^* to measure the gap between our outer bound and the actual capacity region.

We measure r^* by using Monte Carlo method, i.e., randomly generating 1000 different direction $\vec{w}$'s. We show *ratio v.s. number of flows* results in Fig. 4 for three different traffic patterns. As we can see, the ratio r^* does not monotonically change when the number of flows increases. When we compare the ratios for different cases with different flows, the traffic patterns may not change in a "monotonic" way though all flows in those different cases share similar A&E profile. Thus, we may not be able to observe the monotonicity. But we can observe the decreasing trend of r^* when the number of flows becomes larger. Note that due to the high complexity, it requires more significant computing resources than those we can access to solve the RAC problem $(\mathbf{P2})$ for more than 10 flows, and thus we only show the results up to 10 flows.

B. Maximizing Network Utility

In this subsection we evaluate the two proposed scheduling policies to maximize the network utility: one is the provably optimal RAC scheduling policy; the other is the low-complexity heuristic RAC-Approx scheduling policy. We show their performances by considering the following 3-flow traffic pattern:

$$(\mathsf{offset}_1, \mathsf{prd}_1, \mathsf{D}_1, \mathsf{B}_1, p_1) = (0, 4, 4, 1, 0.5),$$
$$(\mathsf{offset}_2, \mathsf{prd}_2, \mathsf{D}_2, \mathsf{B}_2, p_2) = (2, 4, 4, 1, 0.5),$$
$$(\mathsf{offset}_3, \mathsf{prd}_3, \mathsf{D}_3, \mathsf{B}_3, p_3) = (0, 1, 3, 0.9, 0.7).$$

We first set the utility functions as $U_k(R_k) = \log R_k$, $\forall k \in [1, 3]$. Note that here flow 3 is simply the traditional i.i.d. arrival since $\mathsf{prd}_3 = 1$. By solving $(\mathbf{P2})$, we get the optimal timely throughput vector $(R_1^*, R_2^*, R_3^*) = (0.1667, 0.1667, 0.2333)$, which maximizes the network utility. To see concretely how our optimal RAC policy works, in Appendix F in the supplementary materials, we also show the conditional probability $\mathsf{Prob}_{A_t|S_t^k}(a|s^k)$ of the optimal RAC policy (see (12)). Fig. 5 shows that both RAC and RAC-Approx converge to the optimal solution. This verifies the optimality of RAC scheduling policy and demonstrates the good empirical performance of RAC-Approx scheduling policy. Note that in Fig. 5 and later figures in this section, we define the flow-k *running timely throughput* at slot t as

$$\frac{1}{t} \cdot \{\#\text{ of flow-}k\text{ pkts delivered before expiration in } [1, t]\}.$$

We also evaluate RAC and RAC-Approx policies with different utility functions. Specifically, we set $U_1(R_1) = 2\sqrt{R_1}$, $U_2(R_2) = \sqrt{R_2}$, and $U_3(R_3) = \sqrt{R_3}$. The results are shown in Fig. 6. As we can see, our provably optimal RAC policy again converges to the optimal timely throughput vector $(R_1^*, R_2^*, R_3^*) = (0.2344, 0.1107, 0.2169)$ with

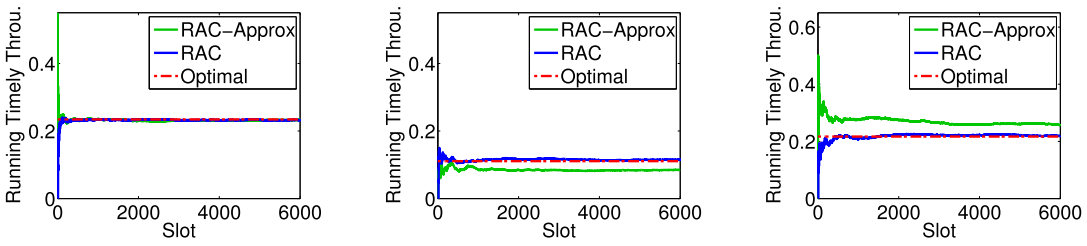


Fig. 6. Comparison of different scheduling policies for maximizing network utility for three flows in Sec. VII-B with utility functions, $U_1(R_1) = 2\sqrt{R_1}$, $U_2(R_2) = \sqrt{R_2}$, and $U_3(R_3) = \sqrt{R_3}$. RAC scheduling policy converges to the optimal solution and RAC-Approx scheduling policy converges to a near-optimal solution which achieves 99.81% of the optimal utility. (a) Flow 1. (b) Flow 2. (c) Flow 3.

optimal utility $u^* = 2\sqrt{R_1^*} + \sqrt{R_2^*} + \sqrt{R_3^*} = 1.7667$. Our proposed low-complexity RAC-Approx policy converges to a sub-optimal timely throughput vector $(\bar{R}_1, \bar{R}_2, \bar{R}_3) = (0.2328, 0.0848, 0.2573)$ with utility $\bar{u} = 2\sqrt{\bar{R}_1} + \sqrt{\bar{R}_2} + \sqrt{\bar{R}_3} = 1.7634$. Though RAC-Approx does not achieve the optimal utility, it has quite good performance with a utility ratio $\bar{u}/u^* = 99.81\%$.

C. Supporting Feasible Timely Throughput Vectors

In this subsection we evaluate the three proposed scheduling polices to support feasible throughput vectors: the first is the provably optimal RAC scheduling policy; the second is the low-complexity heuristic RAC-Approx scheduling policy; the last is the low-complexity deficit-based L-LDF scheduling policy. We will compare them to existing LDF [2] and EPDF [16] scheduling policies.

Since all these scheduling policies require a feasible timely throughput vector as an input, we will also set a utility function $U_k(R_k)$ for each flow k and solve $(\mathbf{P2})$ to get a timely throughput vector on the boundary of the capacity region. We then use it as the input to the scheduling policies. In the following, we use two simple scenarios to show that both LDF and EPDF can be strictly suboptimal while our proposed RAC policy is guaranteed to achieve optimality in all scenarios. Our heuristic solutions RAC-Approx and L-LDF also outperform LDF and EPDF in these two examples.

LDF is Sub-optimal: Consider a 2-flow case with

$$(\mathsf{offset}_1, \mathsf{prd}_1, \mathsf{D}_1, \mathsf{B}_1, p_1) = (0, 4, 4, 1, 0.5), U_1(R_1) = R_1,$$
$$(\mathsf{offset}_2, \mathsf{prd}_2, \mathsf{D}_2, \mathsf{B}_2, p_2) = (2, 4, 4, 1, 0.5), U_2(R_2) = R_2.$$

By solving $(\mathbf{P2})$, the optimal timely throughput vector is $(R_1^*, R_2^*) = (0.2187, 0.2187)$. We use (R_1^*, R_2^*) as the timely throughput requirements for the scheduling policies to be evaluated. Fig. 7 shows their performances. As we can see, RAC converges to (R_1^*, R_2^*) as proven in Theorem 2. Our proposed heuristics, RAC-Approx and L-LDF, also converge to (R_1^*, R_2^*). Meanwhile, the LDF algorithm cannot support this particular timely throughput vector; indeed, the achieved rates for both flows are strictly smaller than (R_1^*, R_2^*).

EPDF is Sub-optimal: Consider a 2-flow case with

$$(\mathsf{offset}_1, \mathsf{prd}_1, \mathsf{D}_1, \mathsf{B}_1, p_1) = (0, 4, 4, 1, 0.5),$$
$$(\mathsf{offset}_2, \mathsf{prd}_2, \mathsf{D}_2, \mathsf{B}_2, p_2) = (0, 4, 3, 1, 0.5).$$

We set the utility function as $U_k(R_k) = w_k R_k$ with weights $(w_1, w_2) = (1, 10^{-5})$. Choosing $w_2 = 10^{-5}$ means that we give absolute priority to flow 1. The optimal rate is $(R_1^*, R_2^*) = (0.2344, 0.1250)$, which we input to all scheduling policies. Fig. 8(a) and Fig. 8(b) show that the achieved timely throughputs of our proposed RAC, RAC-Approx, and L-LDF scheduling policies all converge to (R_1^*, R_2^*); hence, they can all support this timely throughput vector. On the contrary, EPDF cannot support this particular timely throughput vector, and thus is strictly sub-optimal. The observation holds for a wide range of different M values as shown in Fig. 8(c) and Fig. 8(d) where M is a tuning parameter of EPDF [16]. The reason is as follows. The choice of $U_1(R_1)$ and $U_2(R_2)$ implies that to achieve the optimal (R_1^*, R_2^*), we must always give priority to flow 1. However, in EPDF, the *periodic virtual injection of every M time slots* ensures that for a constant fraction of time slots, the deficit of flow 2 will be strictly positive. Since flow 2 has an earlier deadline, EPDF will favor flow 2 for a constant fraction of time slots. This is strictly sub-optimal since an optimal policy must always give precedence to flow 1.

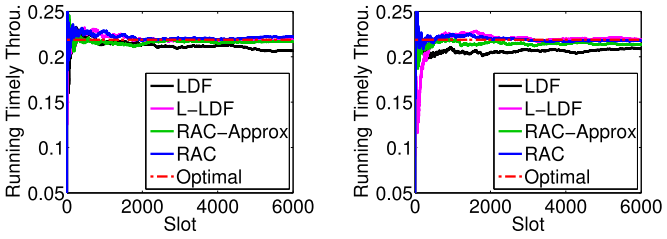


Fig. 7. An example showing that LDF is strictly sub-optimal. (a) Flow 1. (b) Flow 2.

In both Fig. 7 and Fig. 8, we verify that our RAC scheduling policy is feasibility-optimal. We also show that, for the instances considered in this set of simulations, our proposed low-complexity heuristic RAC-Approx and L-LDF scheduling polices support the given timely throughput vector and thus outperform existing alternatives.

D. Average Performance Comparison of Scheduling Policies Over A Large Number of Problem Instances

In this subsection, we compare the performance of the two scheduling policies for maximizing network utility, RAC and RAC-Approx, and five scheduling policies for supporting feasible timely throughput vectors, RAC, RAC-Approx, LLDF, LDF and EPDF, over a large number of randomly generated problem instances with up to 10 flows.

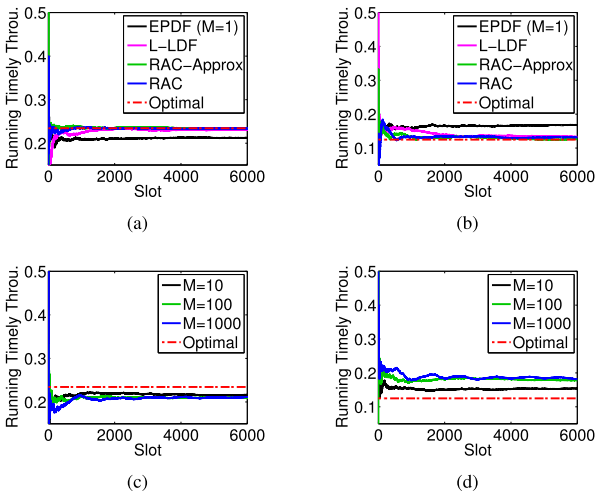


Fig. 8. An example showing that EPDF is strictly sub-optimal. (a) Flow 1. (b) Flow 2. (c) Flow 1 under EPDF. (d) Flow 1 under EPDF.

Our experiments consider K flows where $K \in \{2, 4, 6, 8, 10\}$ and we randomly generate the A&E profile and the successful delivery probability for any flow $k \in [1, K]$, i.e., $(\mathsf{offset}_k, \mathsf{prd}_k, \mathsf{D}_k, \mathsf{B}_k, p_k)$, where offset_k and prd_k are integers uniformly drawn from $[1, 5]$, D_k is an integer uniformly drawn from $[1, \mathsf{prd}_k]$, and B_k and p_k are real numbers uniformly drawn from $[0.5, 1]$. For each flow k, we choose the utility function $U_k(R_k) = \log R_k$.

For the utility-maximization problem, we first solve the utility-maximization problem $(\mathbf{P2})$ and get the optimal network utility u^*. We then evalute the empirical network utility u for RAC and RAC-Approx scheduling polices. We measure the utility gap by

$$\delta_1(u, u^*) \triangleq \frac{u^* - u}{|u^*|}.$$

To evaluate whether a scheduling policy is feasibility-optimal, i.e., capable of supporting any feasible throughput vector in the timely capacity region, we first solve the utility-maximization problem $(\mathbf{P2})$ and get the optimal rate vector $\vec{R}^* = (R_1^*, R_2^*, \cdots, R_K^*)$. We input $\vec{R}^*$ as the timely throughput requirements for RAC-Approx (see $(\mathbf{P3})$), LDF, EPDF, and LLDF scheduling policies. We then evaluate the empirical timely throughput vector $\vec{R}$ for RAC, RAC-Approx, LDF, EPDF, and LLDF scheduling policies. We measure the throughput gap by

$$\delta_2(\vec{R}, \vec{R}^*) \triangleq \frac{\sum_{k=1}^K [R_k^* - R_k]^+}{\sum_{k=1}^K R_k^*},$$

where $[x]^+ \triangleq \max\{x, 0\}$.

For each $K \in \{2, 4, 6, 8, 10\}$, the empirical performance of each instance is measured over 1000000 time slots and we repeat the experiment for 1000 problem instances. We run all evaluations in MATLAB in a cluster of 40 Linux servers, each of which has an 8-core Intel Core-i7 3770 3.4Ghz CPU and up to 61GB memory, running CentOS 6.4. We compute the average utility gap $\delta_1(u, u^*)$ for the two scheduling policies for maximizing network utility, RAC and RAC-Approx, as shown in Tab. II. We compute the average rate gap $\delta_2(\vec{R}, \vec{R}^*)$ for the five scheduling policies for supporting feasible timely throughput vectors, RAC, RAC-Approx, LDF, EPDF, and LLDF, as shown in Tab. III.

TABLE II

PERFORMANCE COMPARISON OF TWO SCHEDULING POLICES FOR MAXIMIZING NETWORK UTILITY, IN TERMS OF THE AVERAGE UTILITY GAP $\delta_1(u, u^*)$ OVER 1000 PROBLEM INSTANCES

Policy \ K	2	4	6	8	10
RAC	0.00%	0.00%	0.01%	0.01%	0.02%
RAC-Approx	0.82%	1.36%	1.76%	2.82%	3.31%

TABLE III

PERFORMANCE COMPARISON OF FIVE SCHEDULING POLICES FOR SUPPORTING FEASIBLE TIMELY THROUGHPUT VECTOR, IN TERMS OF THE AVERAGE THROUGHPUT GAP $\delta_2(\vec{R}, \vec{R}^*)$ OVER 1000 PROBLEM INSTANCES

Policy \ K	2	4	6	8	10
RAC	0.04%	0.06%	0.09%	0.14%	0.16%
LLDF	0.85%	1.31%	0.80%	1.13%	1.20%
LDF	1.91%	3.34%	1.95%	1.76%	1.76%
RAC-Approx	1.77%	3.94%	5.25%	6.24%	6.43%
EPDF	1.81%	6.02%	6.27%	9.38%	8.99%

For maximizing network utility, Tab. II verifies that our proposed high-complexity RAC policy achieves the optimal network utility, and also shows that our proposed low-complexity RAC-Approx policy achieves near-optimal performance. For supporting feasible timely throughput vector, Tab. III verifies that our proposed high-complexity RAC policy is feasibility-optimal. Our proposed L-LDF policy achieves the smallest throughput gap among the remaining four low-complexity scheduling policies.

VIII. CONCLUSION AND FUTURE WORK

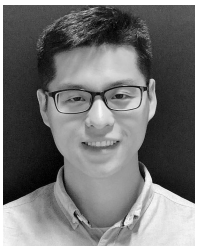
In this paper, we study three fundamental problems of timely wireless flows under *general traffic patterns*: capacity region problem, network utility maximization problem and feasibility-optimal policy design problem. All of them remained largely open. We propose a new MDP-based framework to formulate the timely wireless flow problem with general traffic patterns, which allows us to systematically explore the full design space beyond the existing synchronized-frame-based studies. By applying two problem-structure-inspired simplification methods, *for the first time* we show that all these three fundamental problems can be solved in principle though suffering the curse of dimensionality. Therefore, this paper serves as the ultimate benchmark to evaluate any scheduling policies for timely wireless flows under general traffic patterns. We also take a first step toward addressing the curse of dimensionality by proposing two low-complexity heuristic solutions. Simulation results show that they achieve near-optimal performance and outperform existing alternatives. An interesting and important future direction is to design efficient scheduling algorithms with performance guarantee for general traffic patterns.

REFERENCES

[1] L. Deng, C.-C. Wang, M. Chen, and S. Zhao, "Timely wireless flows with arbitrary traffic patterns: Capacity region and scheduling algorithms," in *Proc. IEEE INFOCOM*, Apr. 2016, pp. 1–9.

[2] I.-H. Hou, V. Borkar, and P. R. Kumar, "A theory of QoS for wireless," in *Proc. IEEE INFOCOM*, Apr. 2009, pp. 486–494.

[3] I.-H. Hou and P. R. Kumar, "Utility maximization for delay constrained QoS in wireless," in *Proc. IEEE INFOCOM*, Mar. 2010, pp. 1–9.
[4] I.-H. Hou and P. R. Kumar, "Admission control and scheduling for QoS guarantees for variable-bit-rate applications on wireless channels," in *Proc. ACM MobiHoc*, 2009, pp. 175–184.
[5] I.-H. Hou and P. R. Kumar, "Utility-optimal scheduling in time-varying wireless networks with delay constraints," in *Proc. ACM MobiHoc*, Mar. 2010, pp. 31–40.
[6] I.-H. Hou and P. R. Kumar, "Scheduling heterogeneous real-time traffic over fading wireless channels," in *Proc. IEEE INFOCOM*, Mar. 2010, pp. 1–9.
[7] *Global Internet Phenomena Report*, Sandvine, Waterloo, ON, Canada, 2015.
[8] Cisco, "Cisco visual networking index: Global mobile data traffic forecast update, 2015–2020," Cisco, San Jose, CA, USA, Tech. Rep. 1454457600805266, 2016.
[9] T. Szigeti and C. Hattingh, "Quality of service design overview," Cisco, San Jose, CA, USA, Tech. Rep., 2004.
[10] F. Bai, T. Elbatt, G. Hollan, H. Krishnan, and V. Sadekar, "Towards characterizing and classifying communication-based automotive applications from a wireless networking perspective," in *Proc. IEEE AutoNet*, Dec. 2006, pp. 1–25.
[11] L. Jiang and J. Walrand, "A distributed CSMA algorithm for throughput and utility maximization in wireless networks," *IEEE/ACM Trans. Netw.*, vol. 18, no. 3, pp. 960–972, Jun. 2010.
[12] J. Ni, B. Tan, and R. Srikant, "Q-CSMA: Queue-length-based CSMA/CA algorithms for achieving maximum throughput and low delay in wireless networks," *IEEE/ACM Trans. Netw.*, vol. 20, no. 3, pp. 825–836, Jun. 2012.
[13] M. Chen, S. C. Liew, Z. Shao, and C. Kai, "Markov approximation for combinatorial network optimization," *IEEE Trans. Inf. Theory*, vol. 59, no. 10, pp. 6301–6327, Oct. 2013.
[14] L. Tassiulas and A. Ephremides, "Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks," *IEEE Trans. Autom. Control*, vol. 37, no. 12, pp. 1936–1948, Dec. 1992.
[15] S. Lashgari and A. S. Avestimehr, "Timely throughput of heterogeneous wireless networks: Fundamental limits and algorithms," *IEEE Trans. Inf. Theory*, vol. 59, no. 12, pp. 8414–8433, Dec. 2013.
[16] I. Hou and R. Singh, "Scheduling of access points for multiple live video streams," in *Proc. ACM MobiHoc*, Jul. 2013, pp. 267–270.
[17] J. J. Jaramillo and R. Srikant, "Optimal scheduling for fair resource allocation in ad hoc networks with elastic and inelastic traffic," in *Proc. IEEE INFOCOM*, Mar. 2010, pp. 1–9.
[18] J. J. Jaramillo, R. Srikant, and L. Ying, "Scheduling for optimal rate allocation in ad hoc networks with heterogeneous delay constraints," *IEEE J. Sel. Areas Commun.*, vol. 29, no. 5, pp. 979–987, May 2011.
[19] X. Kang, W. Wang, J. J. Jaramillo, and L. Ying, "On the performance of largest-deficit-first for scheduling real-time traffic in wireless networks," in *Proc. ACM MobiHoc*, Jul. 2013, pp. 99–108.
[20] X. Kang, I.-H. Hou, and L. Ying, "On the capacity requirement of largest-deficit-first for scheduling real-time traffic in wireless networks," in *Proc. ACM MobiHoc*, Jun. 2015, pp. 217–226.
[21] S. Bodas, S. Shakkottai, L. Ying, and R. Srikant, "Scheduling for small delay in multi-rate multi-channel wireless networks," in *Proc. IEEE INFOCOM*, Apr. 2011, pp. 1251–1259.
[22] Y. Xu, C. Yu, J. Li, and Y. Liu, "Video telephony for end-consumers: Measurement study of Google+, iChat, and Skype," in *Proc. ACM IMC*, 2012, pp. 371–384.
[23] C.-C. Wang, "Delay-constrained capacity for broadcast erasure channels: A linear-coding-based study," in *Proc. IEEE ISIT*, Jul. 2016, pp. 2903–2907.
[24] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Hoboken, NJ, USA: Wiley, 1994.
[25] K. Chatterjee, "Markov decision processes with multiple long-run average objectives," in *Foundations of Software Technology and Theoretical Computer Science*. Berlin, Germany: Springer, 2007.
[26] L. Kruk, J. Lehoczky, and S. Shreve, "Accuracy of state space collapse for earliest-deadline-first queues," *Ann. Appl. Probab.*, vol. 16, no. 2, pp. 516–561, 2006.
[27] S. K. Baruah, L. E. Rosier, and R. R. Howell, "Algorithms and complexity concerning the preemptive scheduling of periodic, real-time tasks on one processor," *Real-Time Syst.*, vol. 2, no. 4, pp. 301–324, 1990.
[28] R. G. Gallager, *Stochastic Processes: Theory for Applications*. Cambridge, U.K.: Cambridge Univ. Press, 2013.



Lei Deng received the B.Eng. degree from the Department of Electronic Engineering, Shanghai Jiao Tong University, Shanghai, China, in 2012, and the Ph.D. degree from the Department of Information Engineering, The Chinese University of Hong Kong, Hong Kong, in 2017. In 2015, he was a Visiting Scholar with the School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN, USA. His research interests are timely network communications, energy-efficient timely transportation, and spectral-energy efficiency in wireless networks.

Chih-Chun Wang (S'02–M'06–SM'14) received the B.E. degree in electrical engineering from National Taiwan University, Taipei, Taiwan, in 1999, and the M.S. and Ph.D. degrees in electrical engineering from Princeton University in 2002 and 2005, respectively. He was with Comtrend Corporation, Taipei, Taiwan, as a Design Engineer in 2000 and with Flarion Technologies, NJ, USA, in 2004. In 2005, he held a post-doctoral researcher position at the Department of Electrical Engineering, Princeton University. He joined Purdue University in 2006, and became a Professor in 2017, where he is currently with the School of Electrical and Computer Engineering. His current research interests are in the delay-constrained information theory, network coding, networking, optimal control, information theory, detection theory, and coding theory. He has been an Associate Editor of the IEEE TRANSACTIONS ON INFORMATION THEORY since 2014 and the Technical Co-Chair of the 2017 IEEE Information Theory Workshop.

Dr. Wang received the National Science Foundation Faculty Early Career Development (CAREER) Award in 2009.

Minghua Chen (S'04–M'06–SM'13) received the B.Eng. and M.S. degrees from the Department of Electronic Engineering, Tsinghua University, in 1999 and 2001, respectively, and the Ph.D. degree from the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, in 2006. He spent one year visiting Microsoft Research Redmond as a Post-Doctoral Researcher. He joined the Department of Information Engineering, The Chinese University of Hong Kong, in 2007, where he is currently an Associate Professor. He is also an Adjunct Associate Professor with the Institute of Interdisciplinary Information Sciences, Tsinghua University. His current research interests include energy systems (e.g., smart power grids and energy-efficient data centers), intelligent transportation system, wireless networking, multimedia networking, online competitive optimization, distributed optimization, and delay-constrained network information flow. He received the Eli Jury Award from UC Berkeley in 2007 (presented to a graduate student or recent alumnus for outstanding achievement in the area of systems, communications, control, or signal processing) and The Chinese University of Hong Kong Young Researcher Award in 2013. He also received several best paper awards, including the IEEE ICME Best Paper Award in 2009, the IEEE TRANSACTIONS ON MULTIMEDIA Prize Paper Award in 2009, and the ACM Multimedia Best Paper Award in 2012. He received the ACM Recognition of Service Award in 2017 for service contribution to the research community. He serves as a TPC Co-Chair of ACM e-Energy 2016 and the General Chair of ACM e-Energy 2017. He is currently an Associate Editor of the IEEE/ACM TRANSACTIONS ON NETWORKING.

Shizhen Zhao received the B.S. degree from Shanghai Jiao Tong University, China, in 2010, and the Ph.D. degree from Purdue University, West Lafayette, IN, USA, in 2015. He is currently with the Google Platform Networking Team, Google Inc. His research interests are in the analysis, control, and optimization in wireless networks, smart grid, and software-defined networks.