Timely Wireless Flows with Arbitrary Traffic Patterns: Capacity Region and Scheduling Algorithms

Lei Deng*, Chih-Chun Wang†, Minghua Chen*, and Shizhen Zhao†
*Department of Information Engineering, The Chinese University of Hong Kong, Hong Kong
†School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN, USA
Email: dl013@ie.cuhk.edu.hk, chihw@purdue.edu, minghua@ie.cuhk.edu.hk, zhao147@purdue.edu

***Abstract*—Most existing wireless networking solutions are best-effort and do not provide any delay guarantee required by important applications such as the control traffic of cyber-physical systems. Recently, Hou and Kumar provided the first framework for analyzing and designing delay-guaranteed network solutions. While inspiring, their idle-time-based analysis appears to apply only to flows with a special *traffic (arrival and expiration) pattern*, and the problem remains largely open for general traffic patterns.**

This paper addresses this challenge by proposing a new 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 formulate the timely capacity problem as an infinite-horizon Markov Decision Process (MDP) and then judiciously combine different simplification methods to convert it to an equivalent finite-size linear program (LP). This allows us to characterize the timely capacity region of flows with general traffic patterns for the first time in the literature. We then design three timely-flow scheduling algorithms for general traffic patterns. The first algorithm achieves the optimal utility but suffers from the curse of dimensionality. The second and third algorithms are inspired by our MDP framework and are of polynomial-time complexity. Simulation results show that both achieve near-optimal performance and outperform other existing alternatives.

I. Introduction

Real-time communication systems over wireless networks 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 result, real-time wireless traffic has expressed a phenomenal growth in recent years [1], and is predicted to increase its volume by another 13-fold in 2014-2019 [2].

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. Similarly, in CPSs, time-critical applications impose latency constraints within which data or control messages must reach their targeting entities [3]. Additionally, real-time communication systems often require guarantees on the *timely throughput*, defined as the throughput of packets that are delivered on time [4], [5].

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

- 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. This requires any communication scheme to address a notoriously hard problem of scheduling across multiple wireless links subject to interference constraints.
- To ensure timely packet delivery, one has to keep track of the delay of individual packets and properly account for delivery urgency in scheduling link transmissions.
- Wireless transmissions are unreliable and subject to shadowing and fading. The channel quality may also vary in time and differ from link to link.

Systematically addressing them calls for a framework that both captures the challenges of delay constrained communication over wireless networks and offers tractable solutions.

Recently, researchers devote much effort to studying real-time wireless communications [4]–[12]. Among them, Hou and Kumar [4], [5], [7] developed an elegant idle-time-based framework to characterize the time capacity region of flows with a special traffic pattern, over single-hop downlink access-point (AP) wireless networks. Further, they proposed a throughput-optimal *largest-deficit-first* (LDF) scheduling policy that can support any feasible rate vector within the timely capacity region. Inspiring as it is, the idle-time-based framework appears to apply only to flows with the special traffic pattern, and new ways of thinking are needed beyond the special traffic pattern. Overall, despite the exciting existing results, the following fundamental questions remain open:

- How to characterize the capacity region of timely flows over wireless networks with *general traffic patterns*?
- How to design new scheduling algorithms that can achieve or approximate the optimal network utility?

In this paper, we take a first step towards answering these questions 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 make the following contributions:

▷ In Sec. II, we model general traffic (arrival and expiration) patterns. Then in Sec. III, we show that network utility maximization of timely flows 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, including [4], [5].

▷ The new MDP formulation is very challenging to solve. In particular, it is of infinite-horizon, infinite state space, and it is even time-heterogeneous. In Sec. IV, by leveraging the underlying structure of the MDP formulation, we apply three simplification methods to convert the challenging MDP problem to an equivalent finite-size linear program (LP). Although our final solution is built on top of the extremely rich literature of MDP, one main contribution of ours is to *judiciously formulate the problem and adapt several existing techniques of MDPs in a coherent way so that we can fully answer the fundamental open question: "What is the timely capacity region for flows with general traffic patterns?"* As a by-product, our LP solution also gives us a provably optimal scheduling policy to achieve the maximal network utility.

▷ However, our optimal MDP-based scheduler suffers from the curse of dimensionality rooted in the MDP approach. Thus in Sec. V, we propose two new computationally-efficient polynomial-time sub-optimal scheduling algorithms. One is based on LP-relaxation and the other is based on the new concept of *lead-time normalization.* Both scheduling algorithms are inspired by our MDP formulation and simulation results in Sec. VI show that they are near-optimal and outperform other conceivable alternatives for most of practical scenarios.

Due to the space limitation, most proofs are presented in our technical report [13].

II. System Model and Problem Formulation

A. The Communication Model

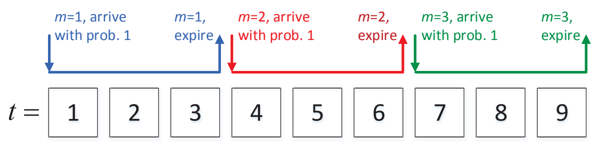
Network Topology and Scheduling Model: We consider a single-hop downlink access-point (AP) scenario where the AP aims to transmit K independent timely traffic to K users, one for each user. The traffic between the AP and user $k \in [1, K]$ is denoted as flow k. Assume slotted transmission. In each slot, only one link can be scheduled and can only send 1 packet. At the beginning of slot t, the action of the AP, denoted by A_t, thus includes two parts: (i) which flow/link to schedule, and (ii) which packet of the selected flow to transmit if there are multiple packets of the selected flow in the current queue. 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 "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*[1] of time t. To model random variation of wireless channels, we assume that along any link k successful delivery happens with some probability p_k, the random delivery events are independently and identically distributed (i.i.d.) over time, and the events for different links are independent.

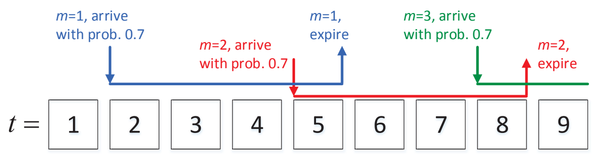
We also assume that at the end of time t, every user will inform the AP through a separate control channel whether it has received the transmitted packet or not (ACK/NACK). The information will then be used for scheduling at time $(t+1)$ and onward.

The above model captures the practical Wi-Fi networks and is widely adopted in the real-time wireless network communication literature, e.g., [4]–[8].

[1]"At the end of slot t" is equivalent to "at (the beginning of) slot $(t+1)$".



(a) $(\mathsf{offset}_1, \mathsf{prd}_1, \mathsf{D}_1, \mathsf{B}_1) = (0, 3, 3, 1)$



m=1, arrive with prob. 0.7
m=2, arrive with prob. 0.7
m=1, expire
m=3, arrive with prob. 0.7
m=2, expire
t =
1
2
3
4
5
6
7
8
9

(b) $(\mathsf{offset}_2, \mathsf{prd}_2, \mathsf{D}_2, \mathsf{B}_2) = (1, 3, 4, 0.7)$

Fig. 1. The illustration of two traffic (arrival & expiration) patterns.

B. Packet Arrival & Hard Delay Constraints

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 "traffic (arrival & expiration) patterns." For any flow k, its traffic pattern can be described by a 4-dim. vector: $(\mathsf{offset}_k, \mathsf{prd}_k, \mathsf{D}_k, \mathsf{B}_k)$. That is, for any integer $m = 1$ to ∞, at slot

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

one flow-k packet will arrive independently *with probability* B_k. If the packet indeed arrived, it will expire at slot

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

Intuitively, offset_k denotes the time offset for the start of the arrival process of flow k. The arrival time of the m-th[2] flow-k packet is thus described by (1). D_k is the deadline for each flow-k packet. Namely, if it arrived, the m-th packet of flow k has to be delivered before its *expiration time*, described in (2). After $t_{\text{exp}}^{[k]}(m)$, the m-th flow-k packet will expire and be removed from the system because it is no longer useful.

An illustration of two traffic patterns 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.

Remark 1: Our general traffic model can capture the following traffic patterns widely used in the scheduling literature:

- **Frame-Synchronized Traffic Pattern**: Let

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

  where T is called the *frame length*. Namely, all K flows arrive and expire simultaneously with period T. This model is used in [4], [8], [10].
- **I.I.D. Traffic Pattern**: Let

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

  Namely, at every slot one flow-k packet arrives with probability B_k and will last for D_k slots. This reduces to the traditional i.i.d. arrival processes. ◇

Remark 2: Although our work is described only for the periodic-i.i.d. traffic patterns, the same principle can be readily

[2] We slightly abuse the notation and still call the packet arriving at $t_{\text{arr}}^{k}(m)$ the m-th packet, even though the previous $m-1$ packets may not arrive.

extended to the much more general cyclostationary Markovian arrivals with observable states. $\Diamond$

C. The Objective

The timely throughput R_k of flow k is defined as

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

which computes the average number of flow-k packets delivered before expiration over the total duration $[1, \mathsf{T}]$.

Obviously, R_k depends on how to schedule the link/packet for all time slots from $t = 1$ to ∞. The network utility maximization (NUM) problem thus becomes

$$(\mathbf{P0}) \qquad \max_{\text{All possible scheduling policies}} \sum_{k=1}^{K} U_k(R_k)$$

where $U_k(\cdot)$ is the utility function for flow k, which is assumed to be increasing, concave, and continuously differentiable. Instead of maximizing the utility as in $(\mathbf{P0})$, a simpler version is to maximize the weighted sum rate

$$(\mathbf{P1}) \qquad \max_{\text{All possible scheduling policies}} \sum_{k=1}^{K} w_k R_k$$

where $w_k > 0$ is the utility coefficient for flow k. Note that $(\mathbf{P0})$ and $(\mathbf{P1})$ are highly related as $(\mathbf{P1})$ can be viewed as an intermediate step of solving $(\mathbf{P0})$ through the Lyapunov-drift analysis[3] (see [14], [15]). Thus, $(\mathbf{P1})$ is the core of the wireless timely throughput problem. For ease of exposition, we will focus mostly on $(\mathbf{P1})$.

III. An Infinite-Dim. Infinite-Horizon MDP

This section explains how to cast the NUM problem $(\mathbf{P1})$ as an infinite-dimension infinite-horizon MDP problem. In Sec. IV, we will further simplify the infinite-size MDP so that the optimal utility and the corresponding optimal scheduling policy can be computed by solving a finite-size LP.

An MDP problem [16] can be described in many different forms, e.g., time-homogeneous MDPs with discounted/average rewards, etc. The MDP used in this work is described by a tuple $(\mathcal{S}, \{\mathcal{A}_s : s \in \mathcal{S}\}, \{P_t\}, r)$ 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 state-to-state transition probabilities in time t:

$$P_t(S_{t+1} = s' | S_t = s, A_t = a), \forall t, \forall s, s' \in \mathcal{S}, \forall a \in \mathcal{A}_s. \quad (3)$$

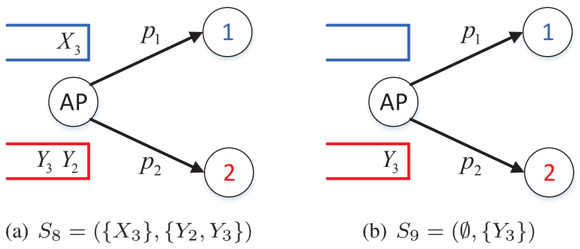
The reward function $r(s, a)$ denotes the per-slot (additive) reward of taking the action $A_t = a$ when the system state is $S_t = s$. We now describe how the NUM problem $(\mathbf{P1})$ can be cast as an MDP by describing the corresponding $(\mathcal{S}, \{\mathcal{A}_s : s \in \mathcal{S}\}, \{P_t\}, r)$.

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, is the collection of all non-expired flow-k packets in the AP's queue.

[3]The reason is that our problem $(\mathbf{P1})$ is exactly a Max-Weight problem and we can regard w_k as the (virtual) queue length of flow k.



X_3
p_1
1
AP
$Y_3\ Y_2$
p_2
2
p_1
1
AP
Y_3
p_2
2
(a) $S_8 = (\{X_3\}, \{Y_2, Y_3\})$
(b) $S_9 = (\emptyset, \{Y_3\})$

Fig. 2. An example for network states at slots 8 and 9, respectively.

For example, suppose that there are only $K = 2$ flows with the corresponding traffic patterns 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.

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

Definition of the action: An action A_t represents which flow and which packet to serve in time t. For example, if the network state at slot 8 is as Fig. 2(a), then there are 3 possible actions[4]:

- Action 1: schedule link 1 and transmit packet X_3;
- Action 2: schedule link 2 and transmit packet Y_2;
- Action 3: schedule link 2 and transmit packet Y_3.

One can quickly see that there are $\max(\sum_{k=1}^{K} |s^k|, 1)$ possible actions[5] when $S_t = s \triangleq (s^1, \cdots, s^K)$ and the collection of them is denoted by $\mathcal{A}_s$. By enumerating over all $s \in \mathcal{S}$, we can specify $\{\mathcal{A}_s : s \in \mathcal{S}\}$.

Definition of the transition probabilities: We observe that the transition probability P_t from $S_t = s$ to $S_{t+1} = s'$ depends on (i) the action $A_t = a$ in slot t; (ii) the channel success probabilities $\{p_k : k = 1, \cdots, K\}$; and (iii) 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) to (iii), we can explicitly construct the transition probability P_t in (3) for all t, s, s' and a. For example, 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

[4]Note that it suffices to consider *work-conserving* actions/policies [4].

[5]The maximum operator is to handle the case $s = (\emptyset, \cdots, \emptyset)$, for which we still have an "idle action".

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.

Definition of the reward: Based on the weights w_1 to w_K in $(\mathbf{P1})$ and we define $r(s,a) = \sum_{k=1}^{K} w_k r_k(s,a)$, where $r_k(s,a)$ is the individual reward for flow k:

$$r_k(s,a) = p_k \cdot 1_{\{\text{a flow-}k\text{ pkt is scheduled under state } s \text{ \& action } a\}} \quad (4)$$

That is, the indicator function $1_{\{\cdot\}}$ counts whether the chosen action schedules a flow-k packet and p_k is the probability that the scheduled packet is successfully delivered. Eq. (4) 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 in the queue, 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 overall reward is

$$r(S_8, A_8) = w_1 r_1(S_8, A_8) + w_2 r_2(S_8, A_8) = w_1 \cdot p_1 + w_2 \cdot 0.$$

The overall MDP-based optimization: The problem $(\mathbf{P1})$ is equivalent to the following infinite-horizon MDP problem:

$$(\mathbf{P2}) \quad \max_{\text{solutions of the MDP}} \liminf_{\mathsf{T}\to\infty} \frac{\sum_{t=1}^{\mathsf{T}} \mathsf{E}\{\sum_{k=1}^{K} w_k r_k(S_t, A_t)\}}{\mathsf{T}}$$

where the underlaying MDP is characterized by $(\mathcal{S}, \{\mathcal{A}_s : s \in \mathcal{S}\}, \{P_t\}, r)$ described previously.

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 control variable A_t not only affects $r(S_t, A_t)$ but also affects the next network state S_{t+1} through the transition probability (3).

Remark 3: Although the above MDP captures the essence of timely throughput optimization, it is highly non-trivial to find the optimal solution. Specifically, the MDP of interest has an infinite state space $\mathcal{S}$ and is of infinite horizon. A closer look at the MDP also shows that it is *time inhomogeneous*, which is difficult to solve, since the transition probability P_t depends on whether there is any packet arrives or expires at the beginning of time $(t+1)$ and thus varies for different t's (see (1) and (2)). Next we will provide three lossless simplification methods that circumvent these obstacles and solve the infinite-size MDP problem $(\mathbf{P2})$ by a finite-size LP problem. $\Diamond$

IV. SIMPLIFICATION

The first contribution of this work is to observe that the optimal timely throughput problem $(\mathbf{P1})$ is in essence an infinite-dimension, infinite-horizon, time-inhomogeneous MDP problem $(\mathbf{P2})$. In this section, we demonstrate how to reduce $\mathcal{S}$ and $\mathcal{A}_s$ to finite sets. On the other hand, the time-inhomogeneity cannot be easily circumvented. Instead, we show that our MDP is actually *almost cyclostationary*. We then adapt the existing time-homogeneous infinite-horizon MDP literature for our almost-cyclostationary setting, and derive the corresponding finite-size LP problem that can find the optimal solution of $(\mathbf{P2})$.

A. Reduce The State Space

Define the *lead time* (see [17] 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

$$t_{\text{lead}}^{[k]}(m) = t_{\text{exp}}^{[k]}(m) - \tau. \quad (5)$$

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 (5) 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

$$|\mathcal{S}| = |\mathcal{S}^1| \cdot |\mathcal{S}^2| \cdot \cdots \cdot |\mathcal{S}^K| \leq 2^{\mathsf{D}_1 + \mathsf{D}_2 \cdots \mathsf{D}_K} < \infty. \quad (6)$$

The new lead-time-based state space $\mathcal{S}$ is therefore bounded even though our MDP is of infinite horizon.

The reason that (6) 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. II-B, at each time t, the flow-k state $S_t^k = l_1^k l_2^k, \cdots l_T^k$ can only be one of the following

$$\begin{cases} l_i^k = 0, \forall i \in [1, T], & \text{if no flow-}k \text{ packet;} \\ l_{g(t)}^k = 1, l_i^k = 0, \forall i \neq g(t), & \text{if } \exists \text{ a flow-}k \text{ packet.} \end{cases} \quad (7)$$

where $g(t) = T - ((t-1) \mod T)$. 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 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 (6).

B. Reduce The Action Space

Recall that the action A_t consists of two parts: (i) which flow to schedule, say flow k, and (ii) which flow-k packet to transmit. However, by some simple sample-path-based arguments, one can prove that it is always better to schedule the oldest flow-k packet (the one with the smallest lead time) than to schedule any of the younger packets. Consequently, we can reduce the action space to

$$\mathcal{A}_s = \begin{cases} \{\text{idle}\}, & \text{if } s^k = \vec{0}, \forall k \text{ ;} \\ \{k : s^k \neq \vec{0}\}, & \text{otherwise.} \end{cases} \tag{8}$$

Once flow k is chosen, we automatically schedule the packet with the smallest lead time. For ease of exposition, we sometimes enlarge the individual action space in (8) and just write $\mathcal{A}_s = \{1, 2, \cdots, K\}$[6] or simply $\mathcal{A} = \{1, 2, \cdots, K\}$ by omitting the subscript s.

C. Solving The Infinite-Horizon MDP

With the above simplifications, the new MDP is of finite dimension (but still of infinite horizon). However, it is still time inhomogeneous, which makes it difficult to apply the existing techniques that solve infinite-horizon MDP. 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

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

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 $t' = (l+1) \cdot \mathsf{Prd} + \tau$. $\diamondsuit$

Due to the space limit, we refer the proof to our technical report [13]. The intuition 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 patterns are periodic. A subtle point of the above new observation, i.e., Lemma 1, is that the "period" of P_t depends only on the arrival period prd_k but is independent of the deadlines D_k. For future reference, we define $\tau_{\text{trans}} = L \cdot \mathsf{Prd}$ and call the time interval $[1, \tau_{\text{trans}}]$ the *transient duration*.

Next we will use the fact that P_t is almost cyclostationary to adapt the existing LP methods [16], previously developed for time-homogeneous MDP, and use it to solve ($\mathbf{P2}$). We will particularly emphasize how to interpret the LP solution and use it to design the optimal scheduling solution.

Definition 1: A scheduling policy π is randomized if for every time t, the action A_t is described by a probability

$$\mathsf{Prob}_{A_t|S_t}(a|s) = \mathsf{Prob}(A_t = a|S_t = s).$$

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 $s \in \mathcal{S}, a \in \mathcal{A}, t \geq \tau_{\text{trans}}$. Namely, for all $t \geq \tau_{\text{trans}}$, the conditional probabilities repeat themselves after Prd slots, and (ii) the random process[7] of the MDP state after time τ_{trans}, $\{S_{\tau_{\text{trans}}+t} : \forall t \geq 1\}$, is cyclostationary with period Prd. $\diamondsuit$

We now have the following result.

Theorem 1: The optimal objective value of the MDP ($\mathbf{P2}$) can always be achieved by a randomized almost cyclostationary (RAC) policy. $\diamondsuit$

Proof: Please see Appendix A for the proof sketch. ■

Theorem 1 shows that to achieve the optimal timely throughput we only need to search for the best RAC scheme. This greatly reduces the search space since, if we ignore the transient phase ($t \leq \tau_{\text{trans}}$), 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 search space is now bounded even though we consider an infinite-horizon MDP problem.

The following theorem describes how to search for the optimal $\mathsf{Prob}_{A_t|S_t}(a|s)$ and $\mathsf{Prob}_{S_t}(s)$. Our solution can be viewed as a generalization of [16, Chap. 8] from time-homogeneous MDPs to cyclostationary MDPs.

Theorem 2: The optimal objective of the MDP ($\mathbf{P2}$) can be found by solving the following finite-size LP

$$(\mathbf{P3}) \quad \max_{\vec{R}, \vec{x} \geq 0} \quad \sum_{k=1}^{K} w_k R_k \tag{9a}$$

$$\text{s.t.} \quad \sum_{a \in \mathcal{A}} x_{t+1}(s', a) = \sum_{s \in \mathcal{S}} \sum_{a \in \mathcal{A}} P_t(s'|s, a) x_t(s, a), \quad \forall s' \in \mathcal{S}, t \in [T_1, T_2 - 1] \tag{9b}$$

$$\sum_{a \in \mathcal{A}} x_{T_1}(s', a) = \sum_{s \in \mathcal{S}} \sum_{a \in \mathcal{A}} P_{T_2}(s'|s, a) x_{T_2}(s, a), \quad \forall s' \in \mathcal{S} \tag{9c}$$

$$R_k \leq \sum_{t=T_1}^{T_2} \sum_{s \in \mathcal{S}} \sum_{a \in \mathcal{A}} \frac{r_k(s, a)}{\mathsf{Prd}} x_t(s, a), \forall k \in [1, K] \tag{9d}$$

$$\sum_{s \in \mathcal{S}} \sum_{a \in \mathcal{A}} x_t(s, a) = 1, \forall t \in [T_1, T_2] \tag{9e}$$

where we define $[T_1, T_2] \triangleq [L \cdot \mathsf{Prd} + 1, (L+1)\mathsf{Prd}]$. $\diamondsuit$

The intuition of Theorem 2 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 (9e)). Once

[6]For simplicity, we omit the "idle action" here.

[7]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.

the joint probability $\mathsf{Prob}_{S_t,A_t}(s,a)$ is specified by $x_t(s,a)$, we can uniquely compute the marginal probability $\mathsf{Prob}_{S_t}(s)$ and the conditional probability $\mathsf{Prob}_{A_t|S_t}(a|s)$ by

$$\mathsf{Prob}_{S_t}(s) = \sum_{a\in\mathcal{A}} \mathsf{Prob}_{S_t,A_t}(s,a) = \sum_{a\in\mathcal{A}} x_t(s,a), \quad (10)$$

$$\mathsf{Prob}_{A_t|S_t}(a|s) = \frac{\mathsf{Prob}_{S_t,A_t}(s,a)}{\mathsf{Prob}_{S_t}(s)} = \frac{x_t(s,a)}{\sum_{a'\in\mathcal{A}} x_t(s,a')}. \quad (11)$$

The RAC policy is thus uniquely decided by $x_t(s,a)$. Also note that the total number of such variables $x_t(s,a)$ is $\mathsf{Prd}\cdot K\cdot 2^{\sum_{k=1}^{K}\mathsf{D}_k}$.

In $(\mathbf{P3})$, the right-hand side of (9d) computes the average reward of flow k under such an RAC policy. Eq. (9b) is the consistency condition for time $[T_1, T_2-1]$. The left-hand side of (9b) is the marginal probability $\mathsf{Prob}_{S_{t+1}}(s')$. The right-hand side of (9b) 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, (9c) is the consistency condition from time T_2 back to T_1 since we require the periodicity condition $\mathsf{Prob}_{S_{T_2+1}}(s') = \mathsf{Prob}_{S_{T_1}}(s')$.

Theorem 2 answers the two key questions proposed in Sec. I. Firstly, it computes the optimal value of $(\mathbf{P1})$ and shows that the K-dimensional rate region $\vec{R}=(R_1,\cdots,R_K)$ can be characterized as a polygon specified by (9b) to (9e). Secondly, once we found the optimal solution of $(\mathbf{P3})$, denoted by $x_t^*(s,a)$, we can plug into (10) and (11) to find the optimal RAC scheduling policy. When we actually run the RAC algorithm in a practical system, we do the following.[8] For each time t, we use the set of packets that remain in the queue of the AP to decide the current network state S_t. Then we randomly choose an action by flipping a coin according to the conditional probability (11) computed from $x_t^*(s,a)$.

Remark 4: The existing elegant framework in [4], [5], [7] is based on the frame-synchronized setting. In contrast, our framework applies to arbitrary 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 [4], 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 LP problem. 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*. $\diamondsuit$

V. Complexity Reduction and Heuristic Scheduling Algorithms

Thus far we have characterized the optimal timely throughput region and designed the corresponding optimal RAC scheduling policy. To our best knowledge, it is the first time in the literature that one can fully answer these two questions for arbitrary traffic patterns. However, our MDP-based solution $(\mathbf{P3})$, though being finite and computable, has quite high complexity, which is $\approx \mathsf{Prd}\cdot K\cdot 2^{\sum_{k=1}^{K}\mathsf{D}_k}$ (see Sec. IV-C). This makes it less practical as an online[9] solution.

On the other hand, this section shows that our MDP-based framework $(\mathbf{P3})$ also allows us to systematically design new polynomial-time sub-optimal schedulers that perform quite well numerically for general settings, which is a feature absent from the existing idle-time-based framework [4], [5], [7].

A. RAC Approximation

Our first efficient but sub-optimal policy consists of two steps. *Step 1:* Derive a relaxed verson of the LP $(\mathbf{P3})$, called $(\mathbf{P4})$, such that the optimal objective value of $(\mathbf{P4})$ is larger than that of $(\mathbf{P3})$ but $(\mathbf{P4})$ can be solved in polynomial time. This thus gives a loose but efficient outer bound. *Step 2:* We replace (11) by a new formula based on the efficiently-computed solution of $(\mathbf{P4})$. Heuristically, if the problems $(\mathbf{P4})$ and $(\mathbf{P3})$ are close, then their optimal solutions are also "close", and the resulting control probabilities in (11) will be similar. Our new RAC solution based on $(\mathbf{P4})$ will thus have near-optimal performance of the solution of $(\mathbf{P3})$. Our numerical experiment shows that it is indeed the case.

Step 1: 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* $A_t^{k_1}=A_t^{k_2}$ by replacing them with a *common scheduling frequency constraint*. 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 the K parallel systems must share a common scheduling frequency, i.e.,

$$\sum_{s^k\in\mathcal{S}^k} z_t^k(s^k,a) = z_t(a), \forall k\in[1,K], t \quad (12)$$

Obviously, the sample-path-based *synchronized action constraint* $A_t^{k_1}=A_t^{k_2}$ implies the distribution-based *common scheduling frequency constraint* (12).

Following very similar derivation process as discussed in Sec. IV, we can characterize the performance of the relaxed

[8]For ease of exposition, we omit the design of the transient state $t\le\tau_{\text{trans}}$. The complete RAC design can be found in our technical report [13].

[9]The main complexity is the computation of the optimal $x_t^*(s,a)$ in $(\mathbf{P3})$. Once $x_t^*(s,a)$ is known, the actual RAC scheduler is quite simple and involves only coin-flipping with conditional probability (11). 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 LP solvers, $(\mathbf{P3})$ can be solved in a few seconds with $K=7$ to 12 flows and moderate D_k values.

system by the following LP problem.

$$(\mathbf{P4}) \quad \max_{\vec{R},\vec{z}\geq 0} \quad \sum_{k=1}^{K} w_k R_k \tag{13a}$$

$$\text{s.t.} \quad \sum_{a\in\mathcal{A}} z_{t+1}^k(\tilde{s}^k,a) = \sum_{s^k\in\mathcal{S}^k}\sum_{a\in\mathcal{A}} P_t^k(\tilde{s}^k|s^k,a) z_t^k(s^k,a),$$
$$\forall k\in[1,K], \tilde{s}^k\in\mathcal{S}^k, t\in[T_1,T_2-1] \tag{13b}$$

$$\sum_{a\in\mathcal{A}} z_{T_1}^k(\tilde{s}^k,a) = \sum_{s^k\in\mathcal{S}^k}\sum_{a\in\mathcal{A}} P_{T_2}^k(\tilde{s}^k|s^k,a) z_{T_2}^k(s^k,a),$$
$$\forall k\in[1,K], \tilde{s}^k\in\mathcal{S}^k \tag{13c}$$

$$R_k \leq \sum_{t=T_1}^{T_2}\sum_{s^k\in\mathcal{S}^k}\sum_{a\in\mathcal{A}} \frac{r_k(s^k,a)}{\mathsf{Prd}} z_t^k(s^k,a),$$
$$\forall k\in[1,K] \tag{13d}$$

$$\sum_{s^k\in\mathcal{S}^k} z_t^k(s_k,a) = z_t(a), \forall k\in[1,K], t\in[T_1,T_2] \tag{13e}$$

$$\sum_{s^k\in\mathcal{S}^k}\sum_{a\in\mathcal{A}} z_t^k(s^k,a) = 1, \forall k\in[1,K], t\in[T_1,T_2] \tag{13f}$$

where $P_t^k(\tilde{s}^k|s^k,a)$ is the transition probability from state s^k to state $\tilde{s}^k$ for flow k, $r_k(s^k,a)$ is the flow-k per-slot reward under state s^k and action a (defined similarly as (4)), and T_1 and T_2 are defined as the same in $(\mathbf{P3})$. One can see that the form of $(\mathbf{P4})$ is very close to that of $(\mathbf{P3})$ except that $(\mathbf{P4})$ deals with each 1-to-1 system separately and links them through the common scheduling frequency constraint (13e).

As discussed previously, $(\mathbf{P4})$ is a relaxed version of $(\mathbf{P3})$. In return for the relaxation, we can solve $(\mathbf{P4})$ more efficiently since the state of each flow k is considered separately (rather than considered as a joint network state). The new complexity thus becomes,

$$(2^{\mathsf{D}_1} + 2^{\mathsf{D}_2} + \cdots + 2^{\mathsf{D}_K} + 1)\cdot K\cdot \mathsf{Prd}.$$

This allows us to solve it for significantly large K, Prd and very reasonable practical D_k values. If we further use the lossless simplification method in (7), then the complexity can be further reduced to

$$(2^{\lceil\frac{\mathsf{D}_1}{\mathsf{prd}_1}\rceil} + 2^{\lceil\frac{\mathsf{D}_2}{\mathsf{prd}_2}\rceil} + \cdots + 2^{\lceil\frac{\mathsf{D}_K}{\mathsf{prd}_K}\rceil} + 1)\cdot K\cdot \mathsf{Prd},$$

which is quite manageable for almost all practical system parameters. Note that it is possible to iteratively use this relaxation method (with looser and looser performance) to further bring down the complexity to purely polynomial $O((\sum_{k=1}^K \mathsf{D}_k)\cdot K\cdot \mathsf{Prd})$. Due to the space limit, we omit the details of the further relaxation.

Step 2: We will now use the optimal solution of $(\mathbf{P4})$, denoted by $\bar{z}_t^k(s^k,a)$, to design the control probability of an RAC policy, i.e., we will replace (11) 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[10] the following conditional probability for each action $a\in\mathcal{A}$ and each flow $k\in[1,K]$,

$$\mathsf{Prob}_{A_t|S_t^k}(a|s^k) = \frac{\bar{z}_t^k(s^k,a)}{\sum_{a'\in\mathcal{A}} \bar{z}_t^k(s^k,a')}, \tag{14}$$

[10]Here we are assuming $t\in[T_1,T_2]$. If not, we need to take the modulo over Prd so that we can plug into the variable $\bar{z}_t^k(s^k,a)$.

and then select action a with probability

$$\frac{\prod_{k=1}^K \mathsf{Prob}_{A_t|S_t^k}(a|s^k)}{\sum_{a'\in\mathcal{A}}\prod_{k=1}^K \mathsf{Prob}_{A_t|S_t^k}(a'|s^k)}. \tag{15}$$

The intuition of (15) is as follows. Eq. (14) is the probability that the k-th parallel system will choose a specific action a. Since all parallel system choose their actions independently, the numerator of (15) is the probability that all flows choose the same action a. When all flows choose the same action a, we let the AP take the same 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 (15), it is as if we directly let the AP randomly choose an action a with probability (15).

Our RAC-Approx policy, using $(\mathbf{P4})$, (14), and (15), is very efficient since all the computation can be performed on a per-flow base, as opposed to the network-wide computation in $(\mathbf{P3})$ and (11).

B. Deficit-based Scheduling Algorithm

Our MDP formulation shows that the lead-time-based state representation is critical to finding the optimal solution. In the following, we will show that by incorporating the concept of lead time, we can further improve the performance of the existing deficit-based policies.

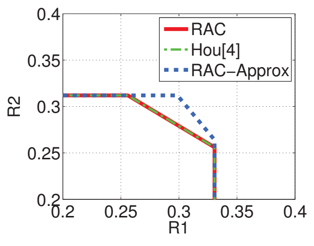
In general, a deficit is the difference between the desired number of transmissions and the actual number of transmissions. Heuristically, the AP should schedule the flow with largest deficit, which is the so-called *Largest-Deficit-First* (LDF) scheduler. [4] proved that LDF is feasiblity/throughput 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, unlike the proposed RAC policy, which is always optimal, the LDF policy is not optimal for general traffic patterns.

To handle heterogeneous deadlines, [9] 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 Fig. 5 and Sec. VI-B for more detailed discussion of the sub-optimality of EPDF.

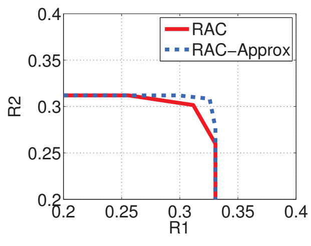
Inspired by our lead-time-based MDP study, we propose the following *Lead-time-normalized-Largest-Deficit-First* (L-LDF) scheduler: At each slot t, the AP computes the *lead-time-normalized deficit* $\bar{d}_k(t)$ for each flow k:

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

where $d_k(t)$ is the deficit of flow k at slot t as defined in [9] (with $M=1$ where M is a parameter in [9]), and smallest-lead-time(k,t) is the smallest lead time among all flow-k packets at slot t. Then, among all flows that currently have packets to send, we select the flow with the largest $\bar{d}_k(t)$.

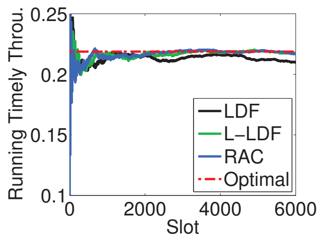


(a) Two synchronized flows

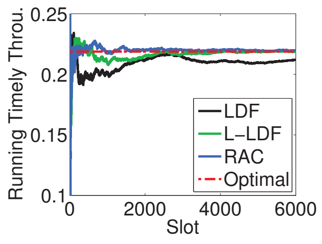


(b) Two non-synchronized flows

Fig. 3. Capacity region of two flows.



(a) Flow 1



(b) Flow 2

Fig. 4. An example showing that LDF is strictly sub-optimal.

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.

VI. Simulation

This section illustrates how to compute the capacity region and compares the performance of the proposed schedulers.

A. Capacity Region

Since the existing idle-time-based analysis [4] can only characterize the capacity region for the frame-synchronized traffic pattern, we also apply our MDP-based computation ($\mathbf{P3}$) 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 [4] and our results successfully characterize the same capacity region.

Next we offset flow-2 by 2 slots. Namely, the two flows are not frame-synchronized any more:

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

The idle-time-based analysis does not hold anymore. However, our MDP-based formulation ($\mathbf{P3}$) 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 relaxed LP ($\mathbf{P4}$), which computes an outer bound of the capacity region. As can be seen, ($\mathbf{P4}$) empirically characterizes a very tight outer bound of the actual capacity region.

B. Proposed Scheduling Algorithms

This work proposes 3 different schedulers: the provably optimal RAC scheduler, the computationally-efficient RAC-Approx scheduler, and the deficit-based L-LDF scheduler. We will compare them to existing LDF [4] and EPDF [9] schedulers. Note that LDF, EPDF and L-LDF are feasibility-check algorithms which require a given timely throughput vector to update the deficits of all flows while the optimal RAC and RAC-Approx do not require such input but need to know the utility function $U_k(r_k)$ for each flow k.

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 ($\mathbf{P3}$), the optimal throughput is $(R_1^*, R_2^*) = (0.2187, 0.2187)$. The optimal RAC converges (R_1^*, R_2^*) as proven in Theorem 1, see Fig. 4. Using (R_1^*, R_2^*) as the rate-vector input for LDF and L-LDF, Fig. 4 shows that LDF is strictly sub-optimal while L-LDF is numerically optimal. L-LDF successfully balances different urgency levels when the flows are no longer frame-synchronized.

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

$$(\mathsf{offset}_1, \mathsf{prd}_1, \mathsf{D}_1, \mathsf{B}_1, p_1) = (0, 4, 4, 1, 0.5), U_1(R_1) = R_1,$$
$$(\mathsf{offset}_2, \mathsf{prd}_2, \mathsf{D}_2, \mathsf{B}_2, p_2) = (0, 4, 3, 1, 0.5), U_2(R_2) = 10^{-5} R_2.$$

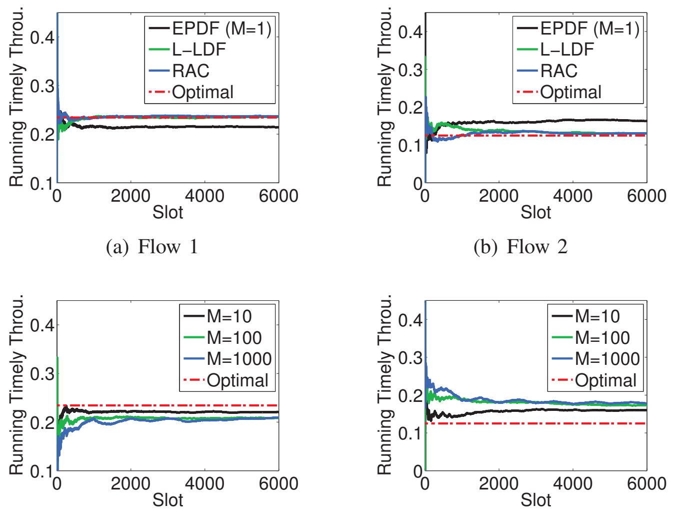
Choosing $w_2 = 10^{-5}$ means that we give absolute priority to flow 1. The optimal rate is $(R_1^*, R_2^*) = (0.2344, 0.1250)$. Fig. 5 shows that the proposed L-LDF and optimal RAC again converge to (R_1^*, R_2^*). However, EPDF is strictly sub-optimal even when using a wide range of different M values, where M is a tuning parameter of EPDF [9]. 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.

Performance of RAC-Approx & L-LDF: For a more diverse traffic pattern, consider a general 3-flow case:

$$(\mathsf{offset}_1, \mathsf{prd}_1, \mathsf{D}_1, \mathsf{B}_1, p_1) = (0, 4, 4, 1, 0.5), U_1(R_1) = \log R_1,$$
$$(\mathsf{offset}_2, \mathsf{prd}_2, \mathsf{D}_2, \mathsf{B}_2, p_2) = (2, 4, 4, 1, 0.5), U_2(R_2) = \log R_2,$$
$$(\mathsf{offset}_3, \mathsf{prd}_3, \mathsf{D}_3, \mathsf{B}_3, p_3) = (0, 1, 3, 0.9, 0.7), U_3(R_3) = \log R_3.$$

Note that here flow 3 is simply the traditional i.i.d. arrival since $\mathsf{prd}_3 = 1$. By solving ($\mathbf{P3}$) where we use the sum-log utility as the objective function[11], the optimal rate vector is uniquely decided as $(R_1^*, R_2^*, R_3^*) = (0.1667, 0.1667, 0.2333)$. Fig. 6 shows that both L-LDF and RAC-Approx converge to the optimal solution, as well as the provably optimal RAC

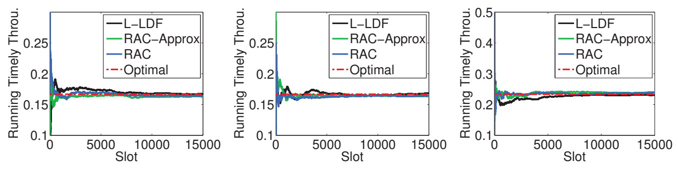
[11] Our LP ($\mathbf{P3}$) can be provably extended to a convex program with general valid utility objective function, which can solve ($\mathbf{P0}$) equivalently.



(a) Flow 1 (b) Flow 2

(c) Flow 1 under EPDF (d) Flow 2 under EPDF

Fig. 5. An example showing that EPDF is strictly sub-optimal.



(a) Flow 1 (b) Flow 2 (c) Flow 3

Fig. 6. Comparison of different scheduling policies for three flows.

scheduler. This again demonstrates the effectiveness of the proposed RAC-Approx and L-LDF schedulers.

VII. Conclusion & Future Work

In this paper, we show that network utility maximization of timely flows with general traffic patterns is fundamentally an MDP problem. This formulation allows us to systematically explore a design space beyond those in previous studies. We propose three problem-structure-inspired simplification methods to convert the challenging infinite-horizon time-heterogenous MDP problem to an equivalent finite-size LP problem. This allows us to fully characterize the timely capacity region of flows with general traffic patterns, which was critically missing in the literature. We then design three new scheduling algorithms for general timely traffic patterns. The first algorithm achieves the optimal utility but suffers from the curse of dimensionality. Inspired by our MDP formulation, the second and third algorithms are of polynomial-time complexity, and simulation results show that they are near-optimal and outperform existing alternatives. An interesting and important future direction is to extend the proposed MDP framework to general multi-hop wireless networks.

Acknowledgment

This work was done while the first author was a visiting scholar at Purdue University. This work was supported in part by NSF grants ECCS-1407603 and CCF-1422997, National Basic Research Program of China (Project No. 2013CB336700), and the University Grants Committee of the Hong Kong Special Administrative Region, China (Area of Excellence Grant Project No. AoE/E-02/08 and General Research Fund No. 14209115).

References

[1] "2H 2014 global Internet phenomena report," Sandvine, 2014.

[2] "Cisco visual networking index: global mobile data traffic forecast update, 2014-2019," Cisco, 2015.

[3] 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 Workshop on Automotive Networking and Applications (AutoNet)*, 2006.

[4] I. Hou, V. Borkar, and P. Kumar, "A theory of QoS for wireless," in *Proc. IEEE INFOCOM*, 2009.

[5] I. Hou and P. Kumar, "Utility-optimal scheduling in time-varying wireless networks with delay constraints," in *Proc. ACM MobiHoc*, 2010.

[6] S. Lashgari and A. S. Avestimehr, "Timely throughput of heterogeneous wireless networks: Fundamental limits and algorithms," *IEEE Trans. Information Theory*, vol. 59, no. 12, pp. 8414–8433, 2013.

[7] I. Hou and P. Kumar, "Scheduling heterogeneous real-time traffic over fading wireless channels," in *Proc. IEEE INFOCOM*, 2010.

[8] ——, "Utility maximization for delay constrained QoS in wireless," in *Proc. IEEE INFOCOM*, 2010.

[9] I. Hou and R. Singh, "Scheduling of access points for multiple live video streams," in *Proc. ACM MobiHoc*, 2013.

[10] 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*, 2010.

[11] J. J. Jaramillo, R. Srikant, and L. Ying, "Scheduling for optimal rate allocation in ad hoc networks with heterogeneous delay constraints," *IEEE J. Selected Areas in Communications*, vol. 29, no. 5, pp. 979–987, 2011.

[12] 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*, 2013.

[13] L. Deng, C.-C. Wang, M. Chen, and S. Zhao, "Timely wireless flows with arbitrary traffic patterns: Capacity region and scheduling algorithms," technical report. [Online]. Available: http://www.ie.cuhk.edu.hk/%7Emhchen/papers/Timely.Wireless.Flow.INFOCOM.16.TR.pdf

[14] M. J. Neely, "Stochastic network optimization with application to communication and queueing systems," *Synthesis Lectures on Communication Networks*, vol. 3, no. 1, pp. 1–211, 2010.

[15] A. L. Stolyar, "Maximizing queueing network utility subject to stability: Greedy primal-dual algorithm," *Queueing Systems*, vol. 50, no. 4, pp. 401–457, 2005.

[16] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., 1994.

[17] Ł. Kruk, J. Lehoczky, and S. Shreve, "Accuracy of state space collapse for earliest-deadline-first queues," *The Annals of Applied Probability*, vol. 16, no. 2, pp. 516–561, 2006.

Appendix

A. *Intuitive Proof of Theorem 1*

The main idea is that the randomness of an RAC scheduler allows us to "average in (probability) space" the state transition and the reward for a given time t. On the other hand, for any given scheme, which may be deterministic or randomized, we can average the state transition and the reward over $t = l \cdot \mathsf{Prd} + \tau$ for all $l = 0, 1, 2, \cdots$. We call such an operation "average in time" since it performs averages over multiple time slots that are of the same position in the overall period Prd. Since the transition probability P_t is almost periodic, we can show that for any given scheme and any RAC scheduler, if the average-in-time of the former gives the same scheduling "frequency" as the average-in-space of the latter, then both schemes achieve the same timely throughput. *As a result, any given scheme (deterministic or not) can be converted to an equally good RAC scheduler.* The proof is thus completed. For the full proof, please refer to our technical report [13].