When Backpressure Meets Predictive Scheduling

Longbo Huang, Shaoquan Zhang, Minghua Chen, *Senior Member, IEEE*, and Xin Liu

***Abstract*—Motivated by the increasing popularity of learning and predicting human user behavior in communication and computing systems, in this paper, we investigate the fundamental benefit of predictive scheduling, i.e., predicting and pre-serving arrivals, in controlled queueing systems. Based on a lookahead-window prediction model, we first establish a novel queue-equivalence between the predictive queueing system with a *fully efficient* scheduling scheme and an equivalent queueing system without prediction. This result allows us to analytically demonstrate that predictive scheduling necessarily improves system delay performance and drives it to zero with increasing prediction power. It also enables us to exactly determine the required prediction power for different systems and study its impact on tail delay. We then propose the `Predictive Backpressure` (`PBP`) algorithm for achieving optimal utility performance in such predictive systems. `PBP` efficiently incorporates prediction into stochastic system control and avoids the great complication due to the exponential state space growth in the prediction window size. We show that `PBP` achieves a utility performance that is within $O(\epsilon)$ of the optimal, for any $\epsilon > 0$, while guaranteeing that the system delay distribution is a *shifted-to-the-left* version of that under the original Backpressure algorithm. Hence, the average delay under `PBP` is strictly better than that under Backpressure, and vanishes with increasing prediction window size. This implies that the resulting utility-delay tradeoff with predictive scheduling can beat the known optimal $[O(\epsilon), O(\log(1/\epsilon))]$ tradeoff for systems without prediction. We also develop the `Predictable-Only PBP` (`POPBP`) algorithm and show that it effectively reduces packet delay in systems where traffic can only be predicted but not pre-served.**

***Index Terms*—Backpressure, human behavior, mobile, optimal control, prediction, queueing.**

I. INTRODUCTION

DUE TO the rapid development of powerful handheld devices, e.g., smartphones or tablet computers, human users now interact much more easily and frequently with the communication and computing infrastructures, e.g., E-commerce websites, cellular networks, and crowdsourcing platforms. Thus, in order to provide high-level quality of service, it is important to understand human behavior features and to utilize such information in guiding system control algorithm design. Therefore, various studies have been conducted to learn and predict human behavior patterns, e.g., online social networking [1], online searching behavior [2], and online browsing [3].

In this paper, we take one step further and ask the following important question: *What is the fundamental system benefit of having such user-behavior information*? Our objective is to obtain a theoretical quantification of this gain. To mathematically carry out our investigation, we consider a multiuser single-server queueing system. At every time, user workload arriving at the system will first be queued at corresponding buffer space. Then, the server allocates resources and decides the scheduling for serving the jobs. These operations allow the server to serve certain amount of workload for each user, but also result in a system cost due to resource utilization. Different from most existing work in multiqueue systems, here we assume that the server can *predict and serve future arrivals before they arrive at the system*. Hence, at every time, the server updates his prediction of future arrivals and adapts his control action. The objective is to serve all user workload with minimum cost, and to ensure small job latency for each user.

This is an important problem and can be used to model many practical systems where traffic prediction and pre-serving can be performed. The first example is scheduling in online video watching. In this case, an online video site, e.g., Youtube, serves users' video demand. Instead of waiting for a user to click on video clips and then start transmitting them, in which case the user may need to wait for the video to load and experience degraded quality of service, the server can guess what the user wants and preload part of the videos to the user's device first. The second example scenario is prefetching in computing systems, e.g., [4] and [5]. Here, data or instructions are preloaded into memory before they are actually requested. Doing so enables faster access or execution of the commands and enhances system performance. Another example is computing management, e.g., in computers. In this case, each user represents a software application, and the server represents a workload management unit. Then, according to the needs of the applications, the managing unit precomputes certain information in case some later applications request them, e.g., branch prediction in computer architecture [6], [7].

There have been many previous works studying multiqueue system scheduling with utility optimization. Reference [8]

Manuscript received June 28, 2014; revised March 23, 2015; accepted June 22, 2015; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor S. Weber. This work was supported in part by the National Basic Research Program of China under Grants 2011CBA00300 and 2011CBA00301; the National Natural Science Foundation of China under Grants 61033001, 61361136003, and 61303195; the Tsinghua Initiative Research Grant; the Microsoft Research Asia Collaborative Research Award; and the China Youth 1000-talent Grant. The work of M. Chen was supported by the National Basic Research Program of China under Project No. 2013CB336700 and the University Grants Committee of the Hong Kong Special Administrative Region, China, under Area of Excellence Grant Project No. AoE/E-02/08 and General Research Fund No. 14201014. The work of X. Liu was supported in part by the NSF under Grants CNS-1457060, CNS-1147930, and CIF-1423542. This paper was presented in part at the 15th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc), Philadelphia, PA, USA, August 2014. *(Corresponding authors: Longbo Huang and Shaoquan Zhang.)*

L. Huang is with the Institute for Interdisciplinary Information Sciences, Tsinghua University, Beijing 100084, China (e-mail: longbohuang@tsinghua.edu.cn).

S. Zhang and M. Chen are with the Department of Information Engineering, The Chinese University of Hong Kong, Shatin, Hong Kong (e-mail: imarch1.zhang@gmail.com; minghua@ie.cuhk.edu.hk).

X. Liu is with the Computer Science Department, University of California, Davis, Davis, CA 95616 USA (e-mail: liu@cs.ucdavis.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNET.2015.2460749

studies the fundamental tradeoff between energy consumption and packet delay for a single-queue system. Reference [9] extends the results to a downlink system and designs algorithms to achieve the optimal tradeoff. Reference [10] designs algorithms for minimizing energy consumption of a stochastic network. Reference [11] designs energy optimal scheme for satellites. Reference [12] looks at the problem of quality-of-service guaranteed energy-efficient transmission using a calculus approach. Reference [13] studies the tradeoff between energy and robustness for downlink systems. References [14] and [15] develop algorithms for achieving the optimal utility-delay tradeoff in multihop networks.

However, we note that all the aforementioned works assume that the system only takes *causal* scheduling actions, i.e., the server will start serving packets only after they enter the system. While this is necessary in many systems, pre-serving future traffic can actually be done in systems that have highly predictable traffic. While predictive scheduling approaches have been investigated, e.g., [7], not much analytical study has been conducted. Closest to our work are [16] and [17], which study the benefit of proactive scheduling, and [18], which studies the impact of future arrival information on queueing delay in $M/M/1$ queues. However, we note that [16] and [17] do not consider the effect of queueing, which very commonly appears in communication and computing systems, whereas [18] does not consider controllable rates and scheduling. Indeed, due to the joint existence of prediction and controlled queueing, the problem considered here is very complicated.

Delay problems in stochastic controlled queueing systems are known to be hard. Moreover, arrival prediction advances in a sliding-window pattern over time, i.e., at every time, the system can predict slightly further into the future. Designing control algorithms for such systems often involves dynamic programming (DP). However, since the state-space size grows exponentially with the prediction window size, the DP approach may not be computationally practical even for small systems. Even without prediction, the complexity of DP can still be very high due to the large queue state space. Moreover, since the system prediction evolves according to a sliding-window pattern, it is also not possible to apply the frame-based Lyapunov technique as in [19] and [20].

To resolve the above difficulties, we first establish a novel equivalence between the queueing system under prediction and a class of *fully efficient* scheduling scheme and a queueing system without prediction but with a different initial condition and an equivalent scheduling policy. This connection is made by carrying out a sample-path queueing argument and enables us to analytically quantify the delay gain due to predictive scheduling for general multiqueue single-server systems. Our result shows that for such systems, the packet delay distribution is *shifted-to-the-left* under predictive scheduling. Hence, the average delay necessarily decreases and *approaches zero* as the prediction window size increases. Based on this result, we further propose a low-complexity `Predictive Backpressure` (`PBP`) scheduling policy for utility maximization in such predictive systems. `PBP` retains all desired features of the original Backpressure algorithm [21], e.g., greedy, does not require statistical information of the system dynamics, and has strong theoretical performance guarantee.

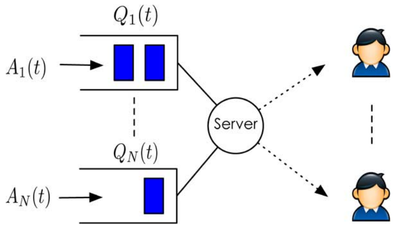


Fig. 1. Multiqueue system where a server is serving workloads for different users/applications.

We prove that the `PBP` algorithm can achieve an average cost that is $O(\epsilon)$ of the minimum cost for any $\epsilon > 0$, while guaranteeing an average delay that is strictly smaller than that under the original Backpressure algorithm. Hence, the resulting utility-delay tradeoff with predictive scheduling can beat the known optimal $[O(\epsilon), O(\log(1/\epsilon))]$ tradeoff for systems without prediction. We also demonstrate analytically and numerically with real data trace that when the first-in–first-out (FIFO) queueing discipline is used, `PBP` achieves an average packet delay reduction that is linear in the prediction window size, and that when the last-in–first-out (LIFO) discipline is used, a prediction window of logarithmic size is enough to guarantee that most packets are pre-served and do not experience any delay at all. These results demonstrate the power of predictive scheduling and provide explicit quantification of the benefits, which also provides useful guidelines for predictive algorithm design.

The rest of the paper is organized as follows. In Section II, we present our system model and problem formulation. We develop the `Predictive Backpressure` (`PBP`) algorithm in Section III. The analysis of delay performance under general predictive scheduling and `PBP` is given in Section IV. We extend the results to predictable-only systems in Section V. Predictive scheduling in multistage processing networks is considered in Section VI. Simulation results are presented in Section VII, followed by the conclusions in Section VIII.

II. SYSTEM MODEL

We consider a general multiqueue single-server system shown in Fig. 1. In this system, a server serves N queues, one for each user that utilizes the service of the server. This multiqueue system has many applications. For instance, it can be used to model downlink transmission in cellular networks, where the server represents the base station and the users are mobile users. Another example is a task management system of a mobile device, where each user represents an application and the server represents the operating system that manages all computing workloads. We assume that the system operates in slotted time, i.e., $t \in \{0, 1, \dots\}$.

A. *Traffic Model*

We use $A_n(t)$ to denote the amount of new workload arriving at the system at time t (called packets below). Here, the workload can represent newly arrived data units that need to be delivered to their destinations, or new computing tasks that the server must fulfill eventually. We use $\boldsymbol{A}(t) = (A_1(t), \dots, A_N(t))$ to denote the vector of arrivals at time t. We assume that $\boldsymbol{A}(t)$ is i.i.d. with $\mathbb{E}\{A_n(t)\} = \lambda_n$.[1] We also assume that for each n, $0 \le A_n(t) \le A_{\max}$.

---

[1] The arrivals can be arbitrarily correlated among different users.

B. Service Rate Model

Every time-slot, the server allocates power for serving the pending packets.[2] However, due to the potential system dynamics, e.g., channel fading coefficient changes, serving different users at different times may result in different resource consumption and generate different service rates. We model this fact by assuming that the server connects to each user n with a time-varying channel, whose state is denoted by $S_n(t)$. We then denote $\boldsymbol{S}(t) = (S_1(t), \dots, S_N(t))$ as the system link state. We assume that $\boldsymbol{S}(t)$ is i.i.d. and takes values in $\{s_1, \dots, s_K\}$.[3] We use π_{s_i} to denote the probability that $\boldsymbol{S}(t) = s_i$.

The server's power allocation over link n at time t is denoted by $P_n(t)$. We denote the aggregate system power allocation vector by $\boldsymbol{P}(t) = (P_1(t), \dots, P_N(t))$. Under a system link state s_i, we assume that the power allocation vector $\boldsymbol{P}(t)$ must be chosen from some feasible power allocation set $\mathcal{P}^{(s_i)}$, which is compact and contains the constraint $0 \leq P_n(t) \leq P_{\max}$. Then, under the given link state $\boldsymbol{S}(t)$ and the power allocation vector $\boldsymbol{P}(t)$, the amount of backlog that can be served for user n is determined by $\mu_n(t) = \mu_n(\boldsymbol{S}(t), \boldsymbol{P}(t))$. We assume that $\mu_n(\boldsymbol{S}(t), \boldsymbol{P}(t))$ is a continuous function of $\boldsymbol{P}(t)$ for all $\boldsymbol{S}(t)$. Also, we assume that there exists $\mu_{\max} < \infty$ such that $\mu_n(\boldsymbol{S}(t), \boldsymbol{P}(t)) \leq \mu_{\max}$ for all n, all time t, and under any $\boldsymbol{S}(t)$ and $\boldsymbol{P}(t)$.

C. Predictive Service Model

Different from most previous works, we assume that the server can *predict and serve* future packet arrivals. Specifically, we first parameterize our prediction model by a vector $\boldsymbol{D} = (D_1, \dots, D_N)$, where $D_n \geq 1$ is the prediction window size of user n. That is, at each time t, the server has access to the arrival information in the *lookahead window* $\{A_n(t), \dots, A_n(t + D_n - 1)\}$ and can allocate rates to serve the future arrivals in the current time-slot.[4] Such a lookahead window model approximates practical scenarios and was also used in [18] and [23]. For notation simplicity, we also use $D_n = 0$ to denote the case when there is no prediction, in which case, we will directly work with $\mu_n(t)$ and $A_n(t)$.

We then use $\{\mu_n^{(d)}(t)\}_{d=0}^{D_n-1}$ to denote the rate allocated in time-slot t to serving the arriving packets in time-slot $t+d$, i.e., the rates are allocated to *pre-serve* the user demand even before they enter the system, and let $\mu_n^{(-1)}(t)$ denote the rate allocated for serving the packets that are already in the system. Note that we always have $\sum_{d=-1}^{D_n-1} \mu_n^{(d)}(t) \leq \mu_n(t)$. Fig. 2 shows the slot structure and the predictive service model.[5]

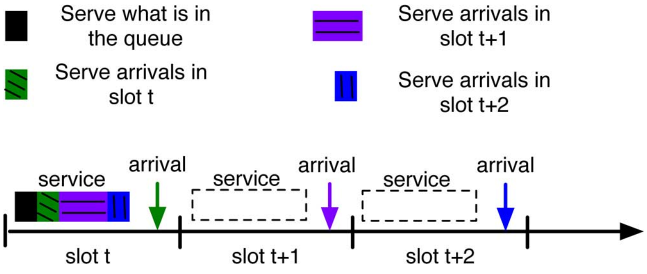

■ Serve what is in the queue
▨ Serve arrivals in slot t+1
▨ Serve arrivals in slot t
▨ Serve arrivals in slot t+2
service
arrival
service
arrival
service
arrival
slot t
slot t+1
slot t+2

Fig. 2. What happens in a single time-slot in the case of $D_n = 3$. The server predicts and serves the arrivals in time-slots t, $t+1$ and $t+2$, respectively.

D. Queueing

Denote by $Q_n(t)$ the number of packets queued at the server for user n (observed at the end of the slot). We assume the following queueing dynamics:

$$Q_n(t+1) = \left[Q_n(t) - \mu_n^{(-1)}(t)\right]^+ + A_n^{(-1)}(t). \tag{1}$$

Here, $A_n^{(-1)}(t)$ denotes the number of packets that *actually* enter the queue after going through a series of predictive service phases, i.e., for all $-1 \leq d \leq D_n - 2$[6]

$$A_n^{(d)}(t) = \left[A_n^{(d+1)}(t) - \mu_n^{(d+1)}(t-d-1)\right]^+ \tag{2}$$

and $A_n^{(D_n-1)}(t) = A_n(t)$. In this paper, we say that the system is *stable* if the following condition holds:

$$Q_{\text{av}} \triangleq \limsup_{t\to\infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \sum_n \mathbb{E}\left\{Q_n(\tau)\right\} < \infty. \tag{3}$$

E. System Objective

In every time-slot, the server spends certain cost due to power expenditure. We denote this cost by $f(\boldsymbol{S}(t), \boldsymbol{P}(t))$. One simple example is $f(S(t), \boldsymbol{P}(t)) = \sum_n P_n(t)$, which denotes the total power consumption. We assume that under any state $\boldsymbol{S}(t)$, there exists a constant $f^{\max}$ such that $f(\boldsymbol{S}(t), \boldsymbol{P}(t)) \leq f^{\max}$. The special case when $f(\boldsymbol{S}(t), \boldsymbol{P}(t))$ is independent of $\boldsymbol{P}(t)$ corresponds to the stability scheduling problem [21].

The system's objective is to find a power allocation and scheduling scheme for minimizing the time average cost, defined as

$$f_{\text{av}} \triangleq \limsup_{t\to\infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\left\{f(\tau)\right\} \tag{4}$$

subject to the constraint that the queues in the system must be stable, i.e., (3) holds. We use $f_{\text{av}}^{\boldsymbol{D}*}$ to denote the minimum average cost under any feasible predictive scheduling algorithm with prediction vector $\boldsymbol{D}$, i.e., those that predict the arrivals for D_n slots and allocate service rates to serving the arrivals within the window $[t, t+D_n-1]$ for each user n. We then use f_{av}^* to denote the minimum average power consumption incurred under any nonpredictive scheduling policy, i.e., $D_n = 0, \forall n$.

F. Discussion of the Model

Note that the lookahead-window model is an idealized model that assumes the system can perfectly predict future arrivals. Because of this, our results can be viewed as upper bounds of the fundamental benefit of predictive scheduling, which provide

---

[2] Our results can be extended to the case where the server consumes multiple types of resources, e.g., power and CPU cycles.

[3] Our results can easily be generalized to the case when both arrivals and channel conditions are Markovian, using the variable-size drift analysis developed in [22].

[4] Since we assume that the arrivals in a time-slot can only be served in the next slot, we also consider $A_n(t)$ to be future arrivals.

[5] Our model does not explicitly take into account the freshness requirement of the contents being served, e.g., freshness of daily news. In this case, predictive scheduling must be done carefully to ensure that users get the most up-to-date contents.

[6] The introduction of $A_n^{(d)}(t)$ here is for simplifying the presentation of the arrival dynamics.

important criteria for evaluating predictive control algorithms. We also investigate the impact of prediction error in Section VII.

We also note that our model is very different from previous controlled queueing system works, which almost all assume that the system operates in a *causal* manner, i.e., only serves packets after they arrive at the system. Our model is motivated by prefetching techniques used in memory management [4], branch prediction in computer architecture [6], as well as recent advancement in data mining for learning user behavior patterns [3].

Our model is most relevant for modeling problems where future workload can be predicted and served before they enter the system. One such application scenario is scheduling in online video watching, where an online video site, e.g., Youtube, serves users' video demand. Instead of waiting for a user to click on video clips and then start to transmit them, in which case the user may need to wait for the video to load and experience degraded quality of service, the server can guess what the user wants and preload part of the videos to the user's device first.[7]

Without such predictive control, the cost minimization problem has been extensively studied, and algorithms have been proposed, e.g., [10]. However, very little is known about the fundamental impact of prediction on system performance, let alone finding optimal control policies for such predictive queueing systems. Moreover, due to the existence of prediction windows and the fact that arrival processes are stochastic, the system naturally evolves according to a Markov chain whose state-space size grows exponentially in the prediction window size. Thus, this problem is challenging to solve.

III. PREDICTIVE BACKPRESSURE

In this section, we present our algorithm, which is designed by incorporating prediction information into the Backpressure technique [21]. Note that since future arrival information is made available in a sliding-window form, prediction couples the current action with future arrivals in every time-slot. This prohibits the use of frame-based Lyapunov technique [22] and makes the problem complicated. Fortunately, as we will see, with the development of a novel *queue-equivalence* result, one can incorporate prediction into system control cleanly and significantly reduce the complexity in both algorithm design and analysis.

A. Prediction Queues

For our algorithm development and analysis, we now introduce the notion of a prediction queue, which records the number of residual arrivals in every slot in time window $[t, t+D_n-1]$. Specifically, we denote $Q_n^{(d)}(t)$ as the number of remaining arrivals currently in future slot $t+d$, i.e., d slots into the future, and denote $Q_n^{(-1)}(t)$ as the number of packets already in the system. We see then the queues evolve according to the following dynamics:

1) If $d = D_n - 1$, then

$$Q_n^{(d)}(t+1) = A_n(t+D_n). \tag{5}$$

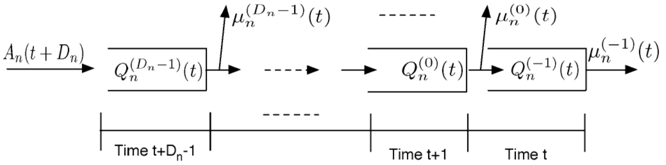


Fig. 3. Prediction queues that describe the system evolution.

2) If $0 \le d \le D_n - 2$, then

$$Q_n^{(d)}(t+1) = \left[Q_n^{(d+1)}(t) - \mu_n^{(d+1)}(t)\right]^+. \tag{6}$$

3) For $Q_n^{(-1)}(t)$, we have

$$Q_n^{(-1)}(t+1) = \left[Q_n^{(-1)}(t) - \mu_n^{(-1)}(t)\right]^+ + \left[Q_n^{(0)}(t) - \mu_n^{(0)}(t)\right]^+ \tag{7}$$

with $Q_n^{(-1)}(0) = 0$.

Fig. 3 shows the definition of the prediction queues. One can see that $\{Q_n^{(d)}(t)\}_{d=0}^{D_n-1}$ are not really queues. They simply record the residual arrivals going through the timeline, whereas $Q_n^{(-1)}(t)$ records the true backlog in the system. Notice that $Q_n^{(-1)}(t)$ is exactly the same as $Q_n(t)$ in (1).[8] Since $Q_n^{(-1)}(t)$ is the only actual queue, the system is stable if and only if $Q_n^{(-1)}(t)$ is stable.

B. Predictive Backpressure

Here, we construct our algorithm based on the above prediction queues and Backpressure. Our main idea is to use the sum of all the queues $Q_n^{\text{sum}}(t) \triangleq \sum_{d=-1}^{D_n-1} Q_n^{(d)}(t)$ for decision making.

To describe the algorithm in detail, we define the notion of queueing discipline for the predictive system, i.e., how to select packets to serve from queues $\{Q_n^{(d)}(t)\}_{d=-1}^{D_n-1}$. Specifically, we order the packets in $Q_n^{(d)}(t)$ with labels $p_{\sum_{d'<d} Q_n^{d'}(t)+1}, \ldots, p_{\sum_{d'\le d} Q_n^{d'}(t)}$. Then, the packets in $\{Q_n^{(d)}(t)\}_{d=-1}^{D_n-1}$ are ordered from p_1 to $p_{Q_n^{\text{sum}}(t)}$. When a particular queueing discipline is applied in the predictive system, we select packets to serve according to the discipline using the order of the packets. For instance, if FIFO is used, then the server will serve the packets $p_1, \ldots, p_{\min[\mu_n(t), Q_n^{\text{sum}}(t)]}$ from the queues every time. We now also define the notion of a *fully efficient* predictive scheduling policy.

Definition 1: A predictive scheduling policy is called *fully efficient* if for every user n, we have: (i) $\sum_d \mu_n^{(d)}(t) = \mu_n(t)$; and (ii) whenever there exists any $-1 \le d \le D_n - 1$ such that $\mu_n^{(d)}(t) > Q_n^{(d)}(t)$, $\mu_n^{d'}(t) \ge Q_n^{d'}(t), \forall d' \neq d$. ◇

In other words, if a policy is fully efficient, it always tries to utilize all service opportunities and does not allocate more service rate to serve any queue unless all other queues are already fully served.[9] Hence, it will not waste any service opportunity unless there are more. With this definition, we now present our algorithm, in which $\epsilon > 0$ is a control parameter used to trade off utility performance and system delay (see Theorem 5).

`Predictive Backpressure (PBP):` In every time-slot, compute $Q_n^{\text{sum}}(t) = \sum_{d=-1}^{D_n-1} Q_n^{(d)}(t)$ for all n. Then, observe the current channel state vector $\boldsymbol{S}(t)$ and perform:

---

[7] In certain scenarios, one may only be able to pre-serve future user demand within a very small time window, e.g., in news reading, even if we can predict that a user will read the daily news starting at a particular time, we can only pre-push news at most a few minutes ahead of time to ensure the freshness of the content. How to adapt our results to handle such freshness requirements is an interesting topic for future research.

[8] This can be seen by considering $t = t - D_n + 1$ in the timeline in Fig. 3 and by equating $Q_n^{(D_n-1)}(t - D_n + 1)$ to $A_n(t)$. Then, it can be checked that (2) and $A_n^{(D_n-1)}(t) = A_n(t)$ coincide with (5) and (6).

[9] It is equivalent to work-conserving in queue scheduling.

- Choose the power allocation vector $\boldsymbol{P}(t)$ to solve the following problem:

$$\min: \quad \frac{1}{\epsilon} f(\boldsymbol{S}(t), \boldsymbol{P}(t)) - \sum_n Q_n^{\text{sum}}(t)\mu_n\left(\boldsymbol{S}(t), \boldsymbol{P}(t)\right) \quad (8)$$

$$\text{s.t.} \quad \boldsymbol{P}(t) \in \mathcal{P}^{(\boldsymbol{S}(t))}. \quad (9)$$

  Then, allocate the service rates $\{\mu_n^{(d)}(t)\}_{d=-1}^{D_n-1}$ to the queues $\{Q_n^{(d)}(t)\}_{d=-1}^{D_n-1}$ in a fully efficient manner according to any prespecified queueing discipline.
- Update the queues according to (5)–(7). ◇

Remark 1: Notice that the `PBP` algorithm has a very clean format. Indeed, `PBP` can be viewed as weighting the predicted future arrivals for different users into current system control. One can in principle also design predictive scheduling algorithms based on DP. However, DP suffers from the well-known curse-of-dimensionality. As an example, suppose each $A_n(t)$ can take 10 values, then there are at least 10^{10} states in DP, even when $n = 2$ and $D_n = 5$. Moreover, with the general rate-power functions $\mu_n(\boldsymbol{S}(t), \boldsymbol{P}(t))$ and cost function $f(\boldsymbol{S}(t), \boldsymbol{P}(t))$, structural properties of the optimal DP-based scheme are hard to characterized (thus it can also be very hard to design approximate DP-based schemes). Our algorithm, on the other hand, offers an efficient way for designing low-complexity predictive control schemes.

IV. PERFORMANCE ANALYSIS

In this section, we first present an important theorem which states that if a predictive scheduling policy is fully efficient, then the queueing system under the scheme evolves in the exact same way as a nonpredictive queueing system with delayed arrivals and a different initial queue state. Using this queue-equivalence result, we obtain an interesting delay distribution shifting theorem. After that, we present our delay analysis for the `PBP` algorithm.

A. Performance of Fully Efficient Scheduling Policies

We start by presenting the theorem regarding the equivalence between predictive and nonpredictive systems.[10]

Theorem 1: Let $\hat{Q}_n(t)$ be the queue size of a single-queue system that: 1) has $\hat{Q}_n(0) = \sum_{t=0}^{D_n-1} A_n(t)$; 2) has arrival $\hat{A}_n(t) = A_n(t + D_n)$; 3) has service $\hat{\mu}_n(t) = \sum_{d=-1}^{D_n-1} \mu_n^{(d)}(t)$; and 4) evolves according to

$$\hat{Q}_n(t+1) = \left[\hat{Q}_n(t) - \hat{\mu}_n(t)\right]^+ + \hat{A}_n(t). \quad (10)$$

Then, if the predictive system uses a fully efficient predictive scheduling policy (with any queueing discipline) with $Q_n^{(-1)}(0) = 0$ for all n, we have for all t that

$$Q_n^{\text{sum}}(t) = \hat{Q}_n(t) \qquad \forall n. \quad ◇ \quad (11)$$

Proof: See Appendix A. ■

Theorem 1 provides an important connection between the predictive system and the system without prediction. Using this result, we derive the following theorem, which relates the delay distribution of the predictive system to the equivalent system without prediction.

Theorem 2: (Delay Distribution Shifting): Denote $\pi_{n,k}^{(D_n)}$ as the steady-state probability that a user n packet experiences a delay of k slots under a fully efficient predictive scheduling policy in the predictive system, and let $\hat{\pi}_{n,k}$ denote the steady-state probability that a user n packet experiences a k-slot delay in $\hat{Q}_n(t)$. Suppose the set of queues $\{Q_n^{(d)}(t)\}_{d=-1}^{D_n-1}$ and $\hat{Q}_n(t)$ use the same queueing discipline. Then, we have for each queue n that[11]

$$\pi_{n,0}^{(D_n)} = \sum_{k=0}^{D_n} \hat{\pi}_{n,k} \quad \text{and} \quad \pi_{n,k}^{(D_n)} = \hat{\pi}_{n,k+D_n}, k \geq 1. \quad (12)$$

That is, the distribution of the original queue can be viewed as *shifted to the left by D_n slots* under predictive scheduling with D_n-slot prediction. ◇

Proof: See Appendix B. ■

Theorem 2 is important for the general framework of predictive scheduling. It allows us to compare scheduling with prediction to the original queueing system without prediction, and enables us to leverage existing results in queueing theory for analyzing predictive systems. To formalize this idea, first notice that if we start with $\hat{Q}_n(0) = 0$ and have $\hat{A}_n(t) = A_n(t)$, then $\hat{Q}_n(t)$ becomes exactly the same as the queueing process in the *original* system *without* prediction. Thus, if the steady-state behavior of $\hat{Q}_n(t)$ does not depend on the initial condition and the shift of the arrival process, e.g., a $G/G/1$ queue [25], then the delay performance of the predictive system can be understood by studying the delay distribution of the original system without prediction.

Corollary 1: Suppose $\{Q_n^{(d)}(t)\}_{d=-1}^{D_n-1}$ and $\hat{Q}_n(t)$ use the same queueing discipline. For any arrival and service processes under which the delay distribution of $\hat{Q}_n(t)$ does not depend on $\hat{Q}_n(0)$ and the shift in the arrival process, we have

$$\pi_{n,0}^{(D_n)} = \sum_{k=0}^{D_n} \pi_{n,k} \quad \text{and} \quad \pi_{n,k}^{(D_n)} = \pi_{n,k+D_n}, k \geq 1. \quad (13)$$

Here, $\pi_{n,k}$ is the steady-state probability that a user n packet experiences a delay of k slots in the system without prediction, i.e., $D_n = 0$. ◇

Corollary 1 applies to general multiqueue single-server systems where the steady-state behavior depends only on the statistical behavior of the arrival and service processes. Note that (13) also quantifies how *tail delay* changes with predictive scheduling. Specifically, we now have in the predictive system that

$$\Pr\{\text{Delay}_n > K\} = \sum_{k>K} \pi_{n,k+D_n}. \quad (14)$$

This is often drastically smaller compared to that under the nonpredictive system (see Figs. 7 and 8 in simulation).

With Theorem 2, we can now quantify how much delay improvement one can obtain via predictive scheduling. This is summarized in the following theorem, in which we use W_{tot} to denote the average delay of the original system without prediction, i.e.,

$$W_{\text{tot}} = \frac{\sum_{n=1}^N \lambda_n \sum_{k\geq 0} k\pi_{n,k}}{\sum_{n=1}^N \lambda_n}. \quad (15)$$

---

[10] We also show in [24] that this result holds in continuous-time queueing systems.

[11] Note that due to the queueing dynamic (1), each packet will stay in the queue for at least one slot in our system. However, we still choose to take the summation starting from delay zero. This is due to the fact that Theorem 2 is intended for general fully efficient scheduling policies and it can be shown to hold for systems with other queueing dynamics where zero delay is possible, provided that the equivalent nonpredictive system is constructed based on the same queueing dynamics.

Theorem 3: Suppose the conditions in Corollary 1 hold. The delay reduction offered by predictive scheduling with prediction window vector $\boldsymbol{D}$, denoted by $R(\boldsymbol{D})$, is given by

$$R(\boldsymbol{D}) = \frac{\sum_{n=1}^{N} \lambda_n (\sum_{1 \le k \le D_n} k\pi_{n,k} + D_n \sum_{k \ge 1} \pi_{n,k+D_n})}{\sum_{n=1}^{N} \lambda_n}. \tag{16}$$

In particular, if $W_{\text{tot}} < \infty$, the average system delay goes to zero as D_n goes to infinity for all queue n, i.e.,

$$\lim_{\boldsymbol{D} \to \infty} R(\boldsymbol{D}) = W_{\text{tot}}. \tag{17}$$

Here, $\boldsymbol{D} \to \infty$ means $D_n \to \infty$ for all n. ◇

Proof: See Appendix C. ■

Theorems 2 and 3 and Corollary 1 show that *systems with predictive scheduling can be analyzed by studying the original system without prediction*. Also note that the above results hold under *any* queueing discipline. The resulting delay distribution, of course, changes under different disciplines. Hence, the results also provide an efficient way for deciding how much prediction power is needed for different systems under different control policies, e.g., if a system has a delay distribution under which most packets experience a delay of no more than D slots, then using a prediction window of D slots suffices to reap most of the benefit of predictive scheduling, and further investment on improving prediction power can readily be saved.

B. Performance of `PBP`

In this section, we analyze the performance of `PBP`. We will assume the *slack condition* (18), i.e., there exist a set of power vectors and probabilities $\{\boldsymbol{P}_m^{(s_i)}, a_m^{(s_i)}\}$, and a constant $\eta > 0$, such that

$$\lambda_n - \sum_{s_i} \pi_{s_i} \sum_{m=0}^{\infty} a_m^{(s_i)} \mu_n(s_i, \boldsymbol{P}_m^{(s_i)}) \le -\eta \qquad \forall n. \tag{18}$$

Formula (18) is commonly assumed in stochastic queueing system works and $\eta \ge 0$ is necessary for system stability [21]. The following theorem states that allowing predictive scheduling does not change the optimal average cost.

Theorem 4: For any vector $\mathbf{0} \preceq \boldsymbol{D} \prec \infty$, we have

$$f_{\text{av}}^{\boldsymbol{D}*} = f_{\text{av}}^{*}. \quad \diamond \tag{19}$$

Proof: See Appendix D. ■

Theorem 4 is interesting and shows that predictive scheduling does not reduce the minimum cost needed for stability. Instead, the theorem, together with Theorem 5, delivers an important message that predictive scheduling *improves the system delay given the same utility performance*.

We now have the following theorem, which shows that `PBP` achieves an average power consumption that is within $O(\epsilon)$ of the minimum and guarantees an average congestion bound.

Theorem 5: The `PBP` algorithm achieves the following:

$$f_{\text{av}}^{\texttt{PBP}} \le f_{\text{av}}^{\boldsymbol{D}*} + B\epsilon, Q_{\text{av}}^{\text{sum}} = Q_{\text{av}}^{\text{BP}} = \frac{B + f^{\max}/\epsilon}{\eta}. \tag{20}$$

Here, $B \triangleq (N/2)(\mu_{\max}^2 + A_{\max}^2)$ is a constant independent of ϵ, $Q_{\text{av}}^{\text{sum}}$ denotes the average expected queue size of $\sum_n Q_n^{\text{sum}}(t)$, and $Q_{\text{av}}^{\text{BP}}$ denotes the average expected queue size of the nonpredictive system under Backpressure. ◇

Proof: See Appendix E. ■

Theorem 5 is similar to the results in previous literature of Backpressure, e.g., [21]. It states that the average size of $\sum_n Q_n^{\text{sum}}(t)$ is the same as $Q_{\text{av}}^{\text{BP}}$ under Backpressure without prediction. Since $Q_n^{\text{sum}}(t)$ is the total size of the actual queue and the prediction queues, we see that the actual queue size is *strictly smaller* than that under Backpressure. Since the average queue size under `PBP` is finite, we can apply Theorem 3 to obtain the following immediate corollary.

Corollary 2: Suppose there exists a steady-state distribution of the queue vector under `PBP`. Then, the average delay under `PBP` goes to zero as $\boldsymbol{D} \to \infty$. ◇

Corollary 2 shows that with predictive scheduling, it is possible to achieve an $O(\epsilon)$ performance with an arbitrarily small average delay. This is fundamentally different from the nonpredictive case, in which the best utility-delay tradeoff is $[O(\epsilon), O(\log(1/\epsilon))]$ [9]. It is also tempting to analyze the exact delay reduction offered by `PBP`. However, due to the complex queueing dynamics under Backpressure, it is challenging to compute the exact distributions $\pi_{n,k}$ even without prediction. Thus, in the following, we consider a general class of cost-minimization problems, and study the delay reduction due to prediction in this case. For stating the results, we define the following optimization problem:

$$\min: \quad f_{\text{av}} = \sum_{s_i} \pi_{s_i} \sum_{m=1}^{N+2} a_m^{(s_i)} f\left(s_i, \boldsymbol{P}_m^{(s_i)}\right) \tag{21}$$

$$\begin{aligned} \text{s.t.} \quad & \sum_{s_i} \pi_{s_i} \sum_{m=1}^{N+2} a_m^{(s_i)} \mu_n\left(s_i, \boldsymbol{P}_m^{(s_i)}\right) \ge \lambda_n \qquad \forall n \\ & \boldsymbol{P}_m^{(s_i)} \in \mathcal{P}^{(s_i)}, \sum_m a_m^{(s_i)} = 1, a_m^{(s_i)} \ge 0 \qquad \forall s_i, m. \end{aligned} \tag{22}$$

Then, we consider a scaled version of its dual problem of problem as follows:

$$\max: \quad g(\boldsymbol{\gamma}), \quad \text{s.t.}\ \boldsymbol{\gamma} \succeq \mathbf{0} \tag{23}$$

where $g(\boldsymbol{\gamma})$ is defined as

$$\begin{aligned} g(\boldsymbol{\gamma}) = \sum_{s_i} \pi_{s_i} \inf_{\boldsymbol{P}_m^{(s_i)} \in \mathcal{P}^{(s_i)}} & \Bigg\{ \frac{1}{\epsilon} f\left(s_i, \boldsymbol{P}_m^{(s_i)}\right) \\ & + \sum_n \gamma_n \left[\lambda_n - \mu_n\left(s_i, \boldsymbol{P}_m^{(s_i)}\right)\right] \Bigg\}. \end{aligned} \tag{24}$$

Note that (24) does not contain the variables $\{a_m^{(s_i)}\}$. This is because the inf operator in (24) ensures that there always exists an optimal set of $\{a_m^{(s_i)*}\}$ values, where for every s_i, there exists one m_i such that $a_{m_i}^{(s_i)*} = 1$ and $a_m^{(s_i)*} = 0$ for $m \neq m_i$. Hence, removing $\{a_m^{(s_i)}\}$ from the dual function does not affect its value at any point and properties. This fact is formally shown in [22]. We now state our theorem regarding the average backlog reduction due to predictive scheduling. In the theorem, we use $\boldsymbol{\gamma}^*$ to denote an optimal solution of (23).

Theorem 6: Suppose: (i) $\boldsymbol{\gamma}^* = \Theta(1/\epsilon) > 0$ is unique; (ii) the η-slack condition (18) is satisfied with $\eta > 0$; (iii) the dual function $g(\boldsymbol{\gamma})$ satisfies

$$g(\boldsymbol{\gamma}^*) \ge g(\boldsymbol{\gamma}) + L\|\boldsymbol{\gamma}^* - \boldsymbol{\gamma}\| \qquad \forall \boldsymbol{\gamma} \succeq \mathbf{0} \tag{25}$$

for some constant $L > 0$ independent of ϵ; (iv) there exists a steady-state distribution of $\boldsymbol{Q}^{\text{sum}}(t)$ under `PBP`; (v) $D_n = O(1/A_{\max}[\gamma_n^* - G - K(\log(1/\epsilon))^2 - \mu_{\max}]^+)$ for all n; and

(iv) FIFO is used. Then, under `PBP` with a sufficiently large $1/\epsilon$, we have

$$Q_{\text{av}}^{(-1)} \leq Q_{\text{av}}^{\text{BP}} - \sum_n D_n \left(\lambda_n - O\left(\epsilon^{\log\left(\frac{1}{\epsilon}\right)}\right)\right)^+ . \quad \diamond \qquad (26)$$

Proof: See Appendix F. ■

As shown in [15], conditions (i)–(iii) in Theorem 6 are satisfied in many practical network optimization problems, especially when the power allocation sets $\{\mathcal{P}^{(s_i)}\}_{i=1}^M$ are finite. In this case, queue vector $\boldsymbol{Q}(t) = (Q_1(t), \ldots, Q_N(t))$ mostly stays close to the fixed point $\boldsymbol{\gamma}^*$ [15]. Using Little's theorem, Theorem 6 implies that the system delay is reduced roughly linearly in the prediction window size D_n. Note that the linear reduction in D_n is due to the use of the FIFO discipline and the fact that the queue size distribution has a concentration property as in Theorem 12, which states that the queue sizes stay close to the optimal Lagrange multipliers γ_n^*. When D_n is larger than $O(1/A_{\max}[\gamma_{Vn}^* - G - K(\log(1/\epsilon))^2 - \mu_{\max}]^+)$, the delay reduction will become sublinear, but more packets will be pre-served.

Below, we consider the case when LIFO is used in `PBP`. In this case, Theorem 7 shows that a small prediction window is enough to guarantee that *most packets experience zero delay*. In the theorem, we define the average rate of the set of packets that are served before entering $Q_n^{(-1)}(t)$, i.e., with delay zero, to be λ_n^0.

Theorem 7: Suppose conditions (i)–(iv) in Theorem 6 hold, and that $D_n = [\log(1/\epsilon)]^k$ for $k > 2$ and for all n. Then, under `PBP` with LIFO, we have

$$\lambda_n^0 \geq \left[\lambda_n - O\left(\frac{1}{\left[\log\left(\frac{1}{\epsilon}\right)\right]^{k-2}}\right)\right]^+ . \quad \diamond \qquad (27)$$

Proof: See Appendix G. ■

Theorem 7 shows that a prediction window of poly-logarithmic size in $1/\epsilon$ is sufficient for guaranteeing that most packets experience zero delay under `PBP` with LIFO. This is very different from the FIFO case and shows that under different scheduling policies, one requires different prediction power for achieving a similar delay reduction.

V. PREDICTABLE-ONLY ARRIVAL

Here, we discuss how `PBP` can also be applied (with slight modification) to the case when arrivals can only be predicted but not pre-served. The idea is to first pretend that the predictable-only traffic can also be pre-served, and then construct the algorithm and show that pre-serving rarely happens.

`Predictable-Only PBP (POPBP):` In every time-slot, run `PBP`. In addition, for each queue n, do:

- (Marking) Mark all packets in $\{Q_n^{(d)}(t)\}_{d=0}^{D_n-1}$ that are served in the current time-slot as *mistaken packets*.
- (Dropping) Drop all mistaken packets when they enter $Q_n^{(-1)}(t)$. $\diamond$

Here, "run `PBP`" means carrying out all the steps in `PBP` including choosing and implementing actions, and updating queue values. Note that now $\{Q_n^{(d)}(t)\}_{d=0}^{D_n-1}$ do not exactly correspond to the number of remaining future arrivals, as they are not served under `POPBP`. Instead, they are equal to the number of future arrivals excluding the mistaken packets.

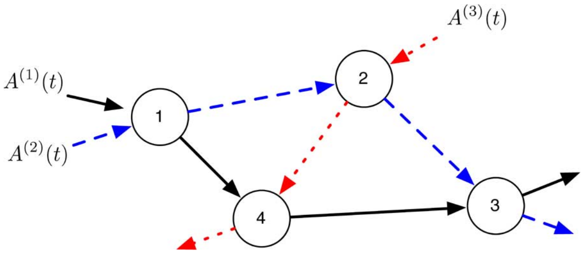


Fig. 4. Multistage processing system. In this system, we have three commodity flows; each represents a certain job. The flows enter and leave the system after being processed. Each node thus maintains certain queues for the jobs.

We summarize the performance of `POPBP` in Theorem 8. Since packets can be dropped under `POPBP`, to take into account the potential additional cost due to dropping, we introduce an additional system cost function $z(\boldsymbol{r}^d) = \sum_n z_n(r_n^d)$, where $\boldsymbol{r}^d = (r_n^d, n = 1, \ldots, N)$ denotes the average packet dropping rate vector at all queues and $z_n(r)$ is the cost of dropping for user n. We assume that for each n: (i) $z_n(0) = 0$; and (ii) $z_n(r) \leq \xi r$ for some finite constant $\xi = \Theta(1) > 0$. For example, $z_n(r)$ can represent the average power consumption for retransmitting the packets that are dropped, or in the utility maximization case, $z_n(r)$ can denote the utility loss due to dropping packets. In both cases, we see that properties (i) and (ii) hold.

Theorem 8: Suppose the conditions in Theorem 6 hold. Then, under `POPBP` with a sufficiently small ϵ, we have

$$f_{\text{av}}^{\texttt{POPBP}} + z^{\texttt{POPBP}}(\boldsymbol{r}^d) \leq f_{\text{av}}^{\boldsymbol{D}*} + \epsilon B + O(\epsilon\xi) \qquad (28)$$

$$Q_{\text{av}}^{(-1)} \leq Q_{\text{av}}^{\text{BP}} - \sum_n D_n \left(\lambda_n - O\left(\epsilon^{\log\left(\frac{1}{\epsilon}\right)}\right)\right)^+ . \qquad (29)$$

Moreover, for each user n, $r_n^d = O(\epsilon^{\log(1/\epsilon)})$. $\diamond$

Proof: See Appendix H. ■

Theorem 8 is interesting and states that even without pre-serving, traffic prediction information can also be used to significantly reduce system latency.

VI. `PBP` FOR MULTISTAGE PROCESSING SYSTEMS

In this section, we extend the results to general multistage processing systems. This system model can be used to model many information or content processing systems, where computing tasks or assembling missions require multiple steps to complete.

Specifically, we consider a general system where $\mathcal{N}$ denotes the set of system nodes and $\mathcal{E}$ denotes the set of links connecting the nodes. Different job flows enter the system and go through a sequence of nodes, which form a single nonrepeating path according to some predetermined processing procedure.[12] The set of job flows is denoted by $\mathcal{C} = \{1, 2, \ldots, C\}$. For instance, in Fig. 4, job flow 2 (the blue dashed job flow) enters from node 1 and goes through the processing of nodes 1–3, then leaves the system. For each job flow c, we denote its exogenous arrivals at node n by $A^{(c)}(t)$, and denote the sequence of nodes it goes through by $\mathcal{P}_c$ (including the source node and the departure node). Then, for each node $n \in \mathcal{P}_c$, we use $up_n^{(c)}$ to denote its upstream node in $\mathcal{P}_c$ and use $down_n^{(c)}$ to denote its downstream node.

---

[12] It is possible to consider the case when certain nodes are repeated a few times, e.g., they handle multiple steps of the job flow processing, by using multiple queues to store intermediate products.

We denote the service condition between two nodes $n, m \in \mathcal{N}$ by $S_{nm}(t)$ and let $\boldsymbol{S}(t) = (S_{nm}(t), [n,m] \in \mathcal{E})$, e.g., whether the resources needed for this processing are scarce right now due to some background processing tasks. We then denote by $P_{nm}(t)$ the resource allocated over link $[n,m]$ for processing. The rate over each link is then determined by $\mu_{nm}(\boldsymbol{S}(t), \boldsymbol{P}(t))$.

In this case, we similarly have the following `Multistage Predictive Backpressure` (`MPBP`). The main idea is again to include future arrivals into the backlog of a node. Specifically, for each node n, if it is the source node of the job flow c, then we define $Q^{(c)}_{\text{eff},n}(t) = \sum_{d=-1}^{D_n-1} Q^{(c,d)}_n(t)$ as the effective backlog of the node, where $Q^{(c,d)}_n(t)$ is defined as in (5)–(7), with $\mu^d_n(t)$ replaced by $\mu^{(c,d)}_{nm}(t)$. Otherwise, if node $n \in \mathcal{P}_c$ is not a source node, it maintains a queue $Q^{(c)}_{\text{eff},n}(t)$ as the current unprocessed work (including actual and predicted or preprocessed traffic) for job flow c, which evolves according to

$$Q^{(c)}_{\text{eff},n}(t+1) = \left[Q^{(c)}_{\text{eff},n}(t) - \mu^{(c)}_{\text{down}^c_n}(t)\right]^+ + \mu^{(c)}_{\text{up}^c_n}(t). \quad (30)$$

Here, $\mu^{(c)}_n(t)$ denotes the rate allocated to serve job flow c at node n at time t. With the definition of $Q^{(c)}_{\text{eff},n}(t)$, the `MPBP` algorithm works as follows.

`Multistage Predictive Backpressure (MPBP):` In every time-slot, observe $Q^{(c)}_{\text{eff},n}(t)$ for all n and c, and the current channel state vector $\boldsymbol{S}(t)$. Perform:

- For every link $[n,m] \in \mathcal{E}$, define
$$W_{nm}(t) = \max_c \left[Q^{(c)}_{\text{eff},n}(t) - Q^{(c)}_{\text{eff},m}(t)\right]^+.$$
Denote c^*_{nm} as the id of the flow that maximizes the above.
- Choose the power allocation vector $\boldsymbol{P}(t)$ to solve the following problem:
$$\min: \quad \frac{1}{\epsilon} f(\boldsymbol{P}(t)) - \sum_{nm} W_{nm}(t)\mu_{nm}(\boldsymbol{S}(t), \boldsymbol{P}(t)) \quad (31)$$
$$\text{s.t.} \quad \boldsymbol{P}(t) \in \mathcal{P}^{\boldsymbol{S}(t)}. \quad (32)$$
  - Allocate the entire rate $\mu_{nm}(t)$ to c^*_{nm}.
  - If node n is the source node of flow c^*_{nm}, allocate the service rates $\{\mu^{(c,d)}_{nm}(t)\}_{d=-1}^{D_n-1}$ to the queues in a fully efficient manner according to any prespecified queueing discipline.
- Update the queues $Q^{(c)}_{\text{eff},n}(t)$ accordingly. ◇

We notice that the `MPBP` algorithm is similar to the original Backpressure algorithm for multihop systems [21]. In this case, the performance of `MPBP` can similarly be analyzed, and the results are summarized in the following theorem.

Theorem 9: Suppose the conditions in Corollary 1 hold. Then, `MPBP` achieves

$$f^{\texttt{PBP}}_{\text{av}} \le f^{\boldsymbol{D}*}_{\text{av}} + O(\epsilon), Q^{\text{sum}}_{\text{av}} = Q^{\text{BP}}_{\text{av}} = O\left(\frac{1}{\epsilon}\right). \quad (33)$$

Moreover, we denote the delay reduction offered by predictive scheduling with prediction window vector $\boldsymbol{D}$ by $R_{\text{muliti}}(\boldsymbol{D})$. Then, if $W_{\text{tot}} < \infty$, the average system delay goes to zero as D_c goes to infinity for all flow c, i.e.,

$$\lim_{\boldsymbol{D}\to\infty} \sum_n R_{\text{muliti}}(\boldsymbol{D}) = W_{\text{tot}}. \quad \diamond \quad (34)$$

Proof: The same as the proof of Theorem 3. Hence, we omit the proof for brevity. ■

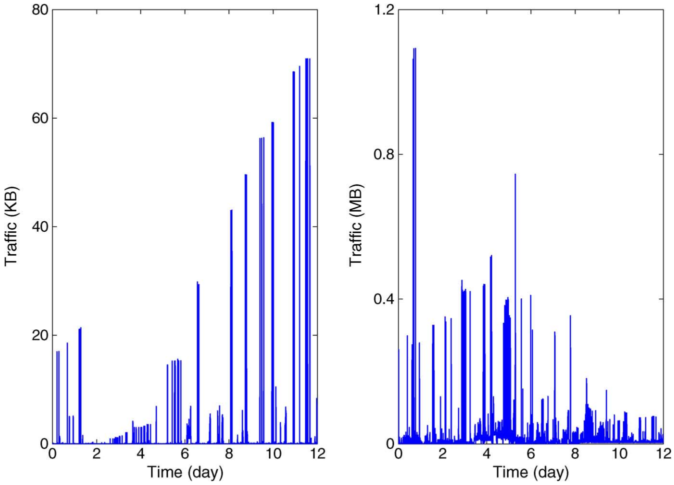


Fig. 5. *(left)* Mobile traffic delivered by the base station to a single user. *(right)* Aggregate mobile traffic delivered by the base station to 10 different users.

VII. SIMULATION

We present simulation results of the `PBP` algorithm in this section in a 10-user single server system.[13]

A. Parameters and Settings

Real Data Trace: We collected data from 10 different mobile users in a 12-day-long period (over 5-min intervals). The data for each user represent the aggregate amount of mobile traffic (over all applications) that is delivered by the base station to the user in that slot. Fig. 5 shows the traffic of a single user and the aggregate traffic of all users in 12 days. In the simulations, we take the traffic data as the workload arriving at the system, which needs to be delivered by the server to different users.

Channel, Power, and Prediction: For each user n, we assume that the channel condition $S_n(t)$ takes values $\{1, 2\}$ with equal probabilities, and $P_n(t) \in \mathcal{P}^{(S_n)} \triangleq \{0, 5, 10\}$. We assume that at any time, only one channel receives nonzero power allocation. The service rate is given by $\mu_n(t) = \lfloor 100 \log(1 + S_n(t)P_n(t)) \rfloor$ kB/s. The cost function $f(S(t), \boldsymbol{P}(t))$ is set to be $\sum_n P_n(t)$, which denotes the total power consumption. We set $D_n = \rho * 5$ for all n. Then, we simulate the cases $\rho \in \{1, 3, 5, 10\}$ to see the effect of the predictive scheduling. We simulate the algorithm with $V \triangleq 1/\epsilon \in \{1, 5, 10, 20, 50, 100\}$.

B. Performance of PBP

Fig. 6 shows the performance of `PBP` with FIFO queueing policy. We see from the left plot that the average power consumption decreases as $1/\epsilon$ increases. The right plot shows the average backlog under `PBP`. It is not hard to see that the average system backlog scales as $O(1/\epsilon)$. One also sees that as the prediction window sizes increase, the network delay decreases linearly in $\boldsymbol{D}$.

Fig. 7 shows the delay distribution under `PBP` for the setting with $1/\epsilon = 100$ and $\rho = 1$. We see that the distributions of the latency for queue n are shifted to the left by D_n, as shown in Theorem 2. It can also be verified that Corollary 1 also holds.

---

[13] Here, we do not compare our scheme to DP-based schemes. As discussed before, designing an implementable DP-based scheme for our general model is a very challenging task and is out of the scope of the current paper.

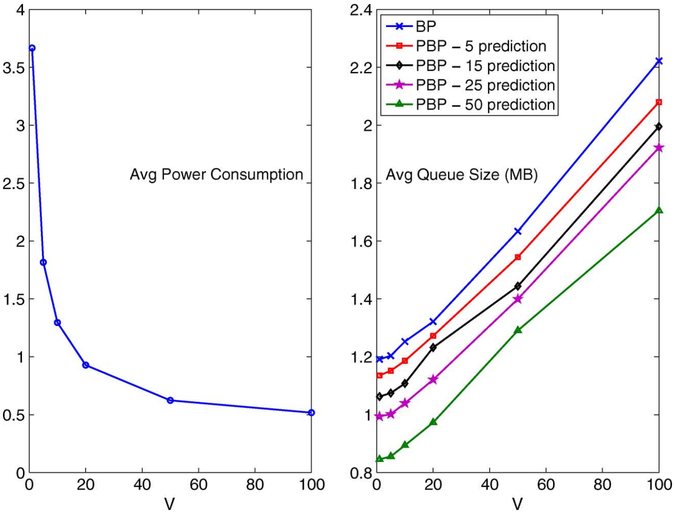


Fig. 6. Performance of `PBP`. *(left)* Average power consumption under `PBP`. *(right)* Average queue size under `PBP` with different prediction window sizes.

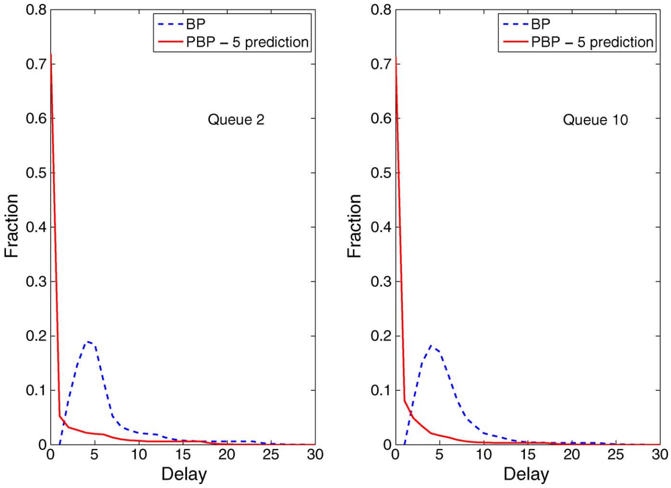


Fig. 7. Packet delay distribution under `PBP` with FIFO scheduling with $1/\epsilon = 100$ [we can also compute the average delay (which is larger than 5) from the distribution] and $D_n = 5$ for all n. We see that predictive scheduling effectively shifts the distribution to the left by 5 slots for both queues.

Fig. 8 then shows the delay distribution under `PBP` and the original Backpressure, under the LIFO discipline. It can be verified that the distribution for the predictive system is also a left-shifted version of the one under Backpressure. We see that large fractions of packets experience zero delay in both queues, i.e., they are served before they arrive at $Q_n^{(-1)}(t)$. This is so because under LIFO Backpressure (no prediction), most packets roughly experience $(\log(1/\epsilon))^2$ delay. Thus, with a moderate-size prediction window size, the server can serve most packets before they enter the system. Since we use a log-scale for the x-axis, we do not plot the fraction for packets that have zero delay. Instead, we show the numbers in the plot with $\pi_0^n(D_n)$. We see that 69% of the packets for queue 2 are served before they even enter the system, whereas 73% of the packets are served for queue 10. These results demonstrate the impact of predictive scheduling on delay reduction.

To also verify Theorem 8, we conducted simulation for `POPBP`. To ensure the conditions of Theorem 8, we choose $1/\epsilon \in \{50, 100, 200, 300, 400, 500\}$ and simulated $D_n = 5$ and 15. We also set $z_n(r_n^d) = 5r_n^d$. For comparison, we have

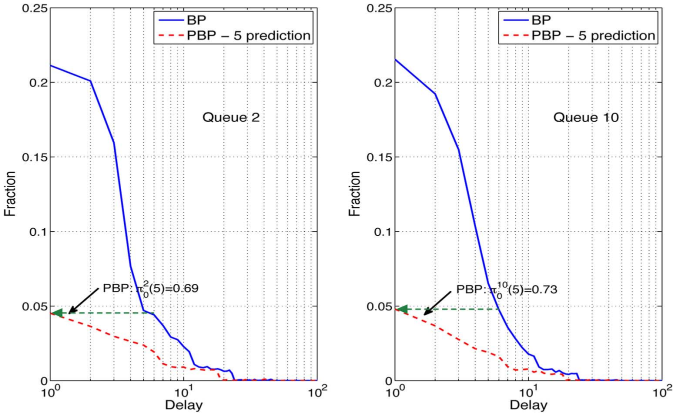


Fig. 8. Packet Delay distribution under `PBP` with LIFO scheduling ($1/\epsilon = 100$ and $D_n = 5$ for all n). The arrows show the shift-to-the-left phenomenon. A large fraction of the packets now experience zero delay under `PBP`, and this fraction is shown with the number $\pi_0^n(D_n)$. This is because with a moderate prediction window size, most packets are served before they arrive at $Q_n^{(-1)}(t)$.

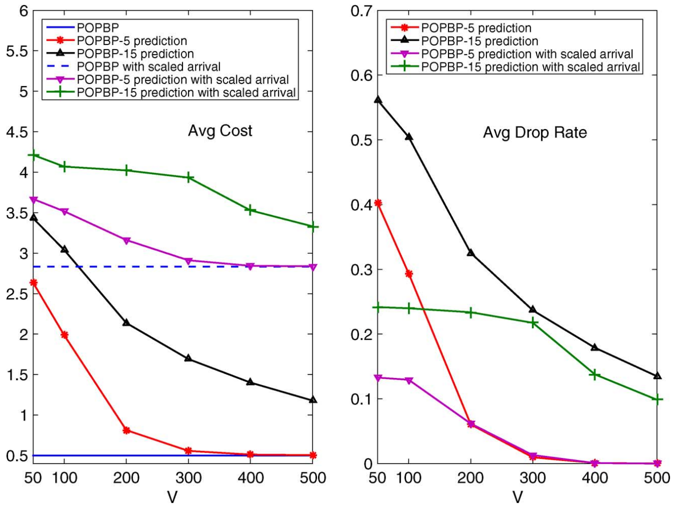


Fig. 9. Average power consumption and dropping rate under `POPBP`. The two horizontal lines in the left plot represent the optimal power consumption under the scaled and unscaled cases.

also simulated another case when the arrival rate is scaled by 5 times. From the right plot of Fig. 9, one sees that the dropping rates under `POPBP` in both the scaled and unscaled case decrease rapidly and quickly converge to 0 when $D_n = 5$ (the $D_n = 15$ case needs larger $1/\epsilon$ values). On the other hand, the left plot shows that the average system cost converges to $f_{\text{av}}^{\boldsymbol{D}*}$, validating Theorem 8.

C. Impact of Imperfect Prediction

To investigate the impact of imperfect prediction, we consider two types of prediction errors. The first type is failing to predict actual arrivals, i.e., miss detection. When it happens, the arrivals will be out of the system's vision and thus will not appear in prediction queues. Therefore, they cannot be served predictively. The other type is false alarm, which happens when the system mistakenly predicts the existence of nonexisting arrivals. Such false arrivals will appear in prediction queues, but will not enter the system. However, the system may incorrectly allocate resources to serve them, resulting in wasted service opportunities.

We model miss detections and false alarms as follows. Each time unit, let R' denote the true amount of arrival. We assume

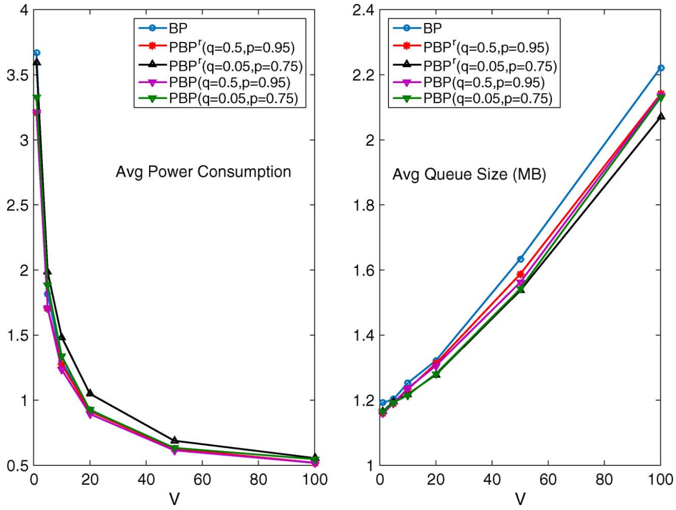


Fig. 10. Performance of `PBP` under imperfect prediction when $D_n = 5$ for all n. Left: Average power consumption under different prediction errors. Right: Average queue size under different prediction errors.

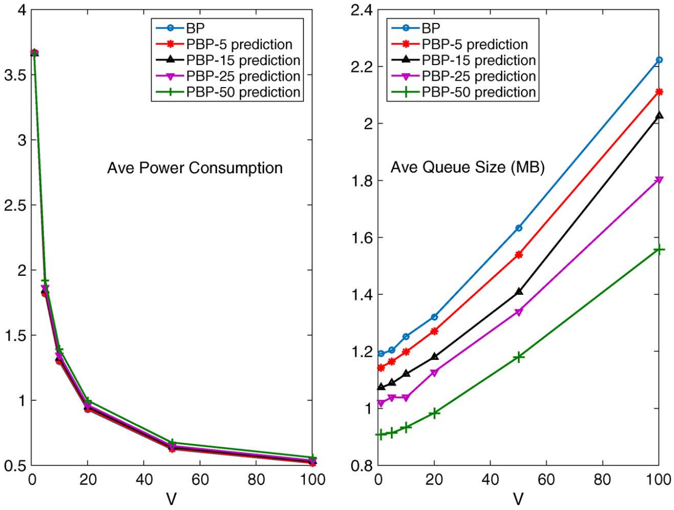


Fig. 11. Performance of `PBP` under the two-state Markov prediction model. The average power consumption remains close to optimal, while the average total queue size is further reduced.

that q fraction of R' are miss detections, i.e., not detected. Denote $R = (1-q)R'$ as the predicted true arrivals. We assume that there are $R \cdot (1-p/p)$ false alarms that come together with them ($p \geq 0.5$). Therefore, the fraction of predicted actual arrivals is given by $(1-q)R/(R \cdot (1-p/p) + R) = (1-q)p$, and these predicted arrivals can be served beforehand. Larger q means more miss detections, and smaller p means more false alarms in the system. For perfect prediction, $q = 0$ and $p = 1$. We simulate three different settings: $(q = 0.5, p = 0.95)$, $(q = 0.05, p = 0,75)$, and $(q = 0.2, p = 0.85)$. The first setting corresponds to the case when the prediction mechanism works very conservatively, which leads to very few false alarms but many miss detections. The second setting corresponds to the case where the prediction mechanism works very aggressively and results in few miss detections but many false alarms. The third setting is in between.

As discussed above, $Q_n^{(d)}(t)$ $(0 \leq d \leq D_n - 1)$ may now contain false alarms besides real arrivals, and the fraction of false alarms is $1-p$ on average. Thus, the effective queue size of $Q_n^{(d)}(t)$ (the number of real arrivals) is $p \cdot Q_n^{(d)}(t)$. Therefore, in the `PBP` algorithm, we use $Q_n^{(-1)}(t) + p\sum_{d=0}^{D_n-1} \cdot Q_n^{(d)}(t)$ as the weight in (8) instead of $\sum_{d=-1}^{D_n-1} Q_n^{(d)}(t)$.

Fig. 10 shows the performance of `PBP` with FIFO queueing policy under imperfect prediction when $D_n = 5$ for all n. We see that `PBP` still improves the delay performance while at the same time keeping a good power performance. This is because that the chance that `PBP` serves future arrivals is decreased when $1/\epsilon$ increases. Therefore, the impact of prediction errors on the utility performance of `PBP` is reduced, showing that `PBP` is robust against prediction error. Compared to `PBP` with perfect prediction, prediction errors increase the average backlog of the system and thus the average delay. This is intuitive since miss detections cannot be served beforehand and false alarms waste service opportunities.

To further evaluate the performance of our algorithms under more realistic prediction model, we construct a two-state Markov model for the arrival (for every user) based on our data trace. Then, we obtain the miss detection and false alarm probabilities with the Markov model. Specifically, in the Markov model, we have two states "L" and "H," indicating low arrival state (arrival less than 15 kB) and high arrival state (such two-state ON/OFF-type Markov models are common in network traffic modeling, e.g., [26]). Then, we estimate the transition probabilities P_{LL}, P_{LH}, P_{HL}, P_{HH} and assume that the operator uses the d-step transition probabilities P_{LL}^d and P_{HH}^d to predict future arrivals when it is in the L state and the H state, respectively. In this case, P_{LH}^d and P_{HL}^d represent the miss detection probability and the false alarm probability, respectively. Under this prediction error model, Fig. 11 shows the performance of the PBP algorithm.

From Fig. 11, we see that `PBP` achieves a similar near-optimal average power consumption, while it further reduces the queue sizes (and delay) compared to the simpler prediction model above. This shows that `PBP` is indeed robust against prediction errors and can perform better with more accurate prediction models.

VIII. CONCLUSION

In this paper, we investigate the fundamental benefit of predictive scheduling in controlled queueing systems. Based on a lookahead prediction window model, we establish a novel queue-equivalence result, which enables exact analysis of queueing systems under predictive scheduling using traditional queueing network control techniques. We then propose the `Predictive Backpressure` (`PBP`) algorithm, and show that `PBP` achieves a cost performance that is arbitrarily close to the optimal, while guaranteeing that the average system delay vanishes as the prediction window size increases. Our results provide useful guidelines for designing control algorithms in systems where system prediction and pre-serving is available, and provide a mathematical framework for provisioning the required prediction power, as well as analyzing the tail delay reduction improvement.

APPENDIX A
PROOF OF THEOREM 1

Here, we prove Theorem 1.

Proof: (Theorem 1) We prove the result by induction with the aid of Fig. 12 showing the evolution of $\hat{Q}_n(t)$.

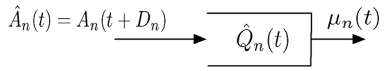


Fig. 12. Original queue without prediction and with a delayed arrival process as well as a different initial queue state.

First, we see that the the result holds for $t = 0$: On one hand, $\hat{Q}_n(0) = \sum_{t=0}^{D_n-1} A_n(t)$. On the other hand, in the system under predictive scheduling, since $Q_n^{(-1)}(0) = 0$ and $Q_n^{(d)}(0) = A_n(d)$ for $d \in \{0, \ldots, D_n - 1\}$, we have $Q_n^{\text{sum}}(0) = \sum_{t=0}^{D_n-1} A_n(t)$.

Suppose the result holds for all $t = 0, \ldots, k$, we show that it holds for $t = k+1$. Using the queueing dynamic (10), we know that in time-slot k, $\tilde{\mu}_n(k) = \min[\mu_n(k), \hat{Q}_n(k)]$ packets will be served from $\hat{Q}_n(k)$. Now consider the queues $\{Q_n^{(d)}(k)\}_{d=-1}^{D_n-1}$. Since the scheduling policy is fully efficient, we must have that the number of packets served from these queues is also $\tilde{\mu}_n(k) = \min[\mu_n(k), \hat{Q}_n(k)]$. To see this, note that if $\tilde{\mu}_n(k) = \mu_n(k)$, there are more packets in the queues than the number of packets that can be served. In this case, we must have $\mu_n^{(d)}(k) \leq Q_n^{(d)}(k)$ for all d. Also, because the policy is fully efficient, we have $\sum_d \mu_n^{(d)}(k) = \mu_n(k)$. Hence, exactly $\mu_n(k)$ packets will be served from $\{Q_n^{(d)}(k)\}_{d=-1}^{D_n-1}$, resulting in $\hat{Q}_n(k+1) = Q_n^{\text{sum}}(k+1)$. On the other hand, suppose $\tilde{\mu}_n(k) = \hat{Q}_n(k)$. Then, there are enough service opportunities to clear all the awaiting packets. In this case, since the scheduling policy is fully efficient, exactly $\hat{Q}_n(k)$ packets will be served. Thus, in both cases, we have $\hat{Q}_n(k+1) = Q_n^{\text{sum}}(k+1) = A_n(k+D_n)$. ■

APPENDIX B
PROOF OF THEOREM 2

Here, we prove Theorem 2.

Proof: (Theorem 2) From Theorem 1, we see that $Q_n^{\text{sum}}(t) = \hat{Q}_n(t)$ for all time. Hence, if the two queueing systems use the same queueing discipline in choosing what packets to serve, then every packet will experience the exact same delay in both $\hat{Q}_n(t)$ and $\{Q_n^{(d)}(t)\}_{d=-1}^{D_n-1}$.

However, in $Q_n^{\text{sum}}(t)$, a packet will enter the actual system only after spending one unit of time in each of the queues in $\{Q_n^{(d)}(k)\}_{d=0}^{D_n-1}$, which takes exactly D_n slots in total. Thus, any packet experiencing a k-slot delay will experience $[k - D_n]^+$ delay in $Q_n^{(-1)}(t)$. ■

APPENDIX C
PROOF OF THEOREM 3

We prove Theorem 3 here.

Proof: (Theorem 3) Using Corollary 1, we see that in the predictive system, the average system backlog size is given by

$$N_{\text{tot}}^{\text{P}} = \sum_{n=1}^{N} \lambda_n \sum_{k \geq 1} k\pi_{n,k+D_n}. \tag{35}$$

On the other hand, the average system backlog without prediction is given by

$$N_{\text{tot}} = \sum_{n=1}^{N} \lambda_n \sum_{k \geq 1} k\pi_{n,k}. \tag{36}$$

Using (36) and (35), we conclude that

$$N_{\text{tot}} - N_{\text{tot}}^{\text{P}} = \sum_{n=1}^{N} \lambda_n \left(\sum_{1 \leq k \leq D_n} k\pi_{n,k} + D_n \sum_{k \geq 1} \pi_{n,k+D_n} \right). \tag{37}$$

Using Little's theorem and dividing both sides by $\sum_n \lambda_n$, we see that (16) follows.

Now we prove (17). By taking a limit as $\boldsymbol{D} \to \infty$, we first obtain

$$\lim_{\boldsymbol{D} \to \infty} \sum_n \lambda_n \sum_{1 \leq k \leq D_n} k\pi_{n,k} = N_{\text{tot}}. \tag{38}$$

Then, using the fact that $W_{\text{tot}} < \infty$, we have

$$\lim_{D_n \to \infty} D_n \sum_{k \geq 1} \pi_{n,k+D_n} = 0 \qquad \forall n. \tag{39}$$

Using the above in (16), we see that (17) follows. ■

APPENDIX D
PROOF OF THEOREM 4

In this section, we prove Theorem 4 using a similar argument as in [10]. For our analysis, we will use the following theorem, which characterizes f_{av}^* in the nonpredictive case (see [10] for its proof).

Theorem 10: The minimum average cost f_{av}^* is the solution to the optimization problem (21). ◇

Theorem 10 can be viewed as saying that the minimum average cost subject to system stability can be achieved by a stationary and randomized policy, which picks a set of power allocations $\{\boldsymbol{P}_m^{(s_i)}, \forall s_i, m\}$ with probabilities $\{a_m^{(s_i)}, \forall s_i, m\}$.

Proof: (Theorem 4) We first see that since any policy without prediction is also a feasible policy for the predictive system, $f_{\text{av}}^{\boldsymbol{D}*} \leq f_{\text{av}}^*$ by definition.

We now prove that $f_{\text{av}}^{\boldsymbol{D}*} \geq f_{\text{av}}^*$. Consider any predictive scheduling scheme Π_P that ensures system stability. Consider the set of slots $t \in \{0, \ldots, M\}$. Let $\mathcal{T}_{s_i}(M)$ denote the set of slots with $\boldsymbol{S}(t) = s_i$ and let $T_{s_i}(M)$ denote its cardinality. We also define the conditional empirical average of transmission rate and power cost under Π_P as follows:

$$\begin{aligned} &\left(\overline{\mu}_1^{(s_i)}(M), \ldots, \overline{\mu}_N^{(s_i)}(M), \overline{f}^{(s_i)}(M)\right) \\ &\triangleq \sum_{t \in \mathcal{T}_{s_i}(M)} \frac{\left(\mu_1^{\Pi_P}(s_i, \boldsymbol{P}(t)), \ldots, \mu_N^{\Pi_P}(s_i, \boldsymbol{P}(t)), f^{\Pi_P}(s_i, \boldsymbol{P}(t))\right)}{T_{s_i}(M)}. \end{aligned} \tag{40}$$

The above is a mapping from the N-dimensional power vector space into the $(N+1)$-dimensional space, and that the right-hand-side is a convex combination of the points in the $(N+1)$-dimensional space. Hence, using Caratheodory's theorem as in [10], one see that for every M, there exists probabilities $\{a_m^{(s_i)}(M)\}_{m=1}^{N+2}$ and power allocation vectors $\{\boldsymbol{P}_m^{(s_i)}(M)\}_{m=1}^{N+2}$, such that

$$\overline{\mu}_n^{(s_i)}(M) = \sum_{m=1}^{N+2} a_m^{(s_i)}(M) \mu_n \left(s_i, \boldsymbol{P}_m^{(s_i)}(M)\right) \qquad \forall n$$

$$\overline{f}^{(s_i)}(M) = \sum_{m=1}^{N+2} a_m^{(s_i)}(M) f \left(s_i, \boldsymbol{P}_m^{(s_i)}(M)\right).$$

Now define

$$\left(\overline{\mu}_1(M),\ldots,\overline{\mu}_N(M),\overline{f}(M)\right) \triangleq \sum_{s_i}\frac{T_{s_i}(M)}{M}\left(\overline{\mu}_1^{(s_i)}(M),\ldots,\overline{\mu}_N^{(s_i)}(M),\overline{f}^{(s_i)}(M)\right).$$

Using the ergodicity of the channel state process, the continuity of $f(s_i,\boldsymbol{P}(t))$ and $\mu_n(s_i,\boldsymbol{P}(t))$, and the compactness of $\mathcal{P}^{(s_i)}$, one can find a sequence of times $\{M_i\}_{i=1}^{\infty}$ and a set of limiting probabilities $\{a_m^{(s_i)}\}_{m=1}^{N+2}$ and power vectors $\{\boldsymbol{P}_m^{(s_i)}\}_{m=1}^{N+2}$ such that

$$f_{\text{av}}^{\Pi_P} = \sum_{s_i}\pi_{s_i}\sum_{m=1}^{N+2}a_m^{(s_i)}f\left(s_i,\boldsymbol{P}_m^{(s_i)}\right) \tag{41}$$

$$\overline{\mu}_n^{\Pi_P} = \sum_{s_i}\pi_{s_i}\sum_{m=1}^{N+2}a_m^{(s_i)}\mu_n\left(s_i,\boldsymbol{P}_m^{(s_i)}\right) \qquad \forall n. \tag{42}$$

Here, $f_{\text{av}}^{\Pi_P}$ denotes the average cost under scheme Π_P and $\overline{\mu}_n^{\Pi_P}$ denotes the average *total* allocated transmission rate to queue n under Π_P. This shows that the average cost and the average allocated rate to any queue under a predictive scheme can be achieved by some randomized schemes.

Let $\beta_n^{(d)}(t)$ be the number of packets that enter $Q_n^{(d)}(t)$ at time t and let $\mu_n^{(d)}(t)$ denote the service rate allocated to serve the packets in $Q_n^{(d)}(t)$ at time t.[14] Further let $\eta_n^{(d)}(t)$ be the number of packets served from $Q_n^{(d)}(t)$ at time t. Then, denote $\beta_n^{(d)}$, $\mu_n^{(d)}$, and $\eta_n^{(d)}$ as their average values, i.e., $\beta_n^{(d)} = \lim_{T\to\infty}(1/T)\sum_{t=0}^{T-1}\mathbb{E}\{\beta_n^{(d)}(t)\}$, $\mu_n^{(d)} = \lim_{T\to\infty}(1/T)\sum_{t=0}^{T-1}\mathbb{E}\{\mu_n^{(d)}(t)\}$, and $\eta_n^{(d)} = \lim_{T\to\infty}(1/T)\sum_{t=0}^{T-1}\mathbb{E}\{\eta_n^{(d)}(t)\}$.[15]

Using the queueing dynamics of $\{Q_n^{(d)}(t)\}_{d=-1}^{D_n-1}$, we have

$$\beta_n^{(d)} - \beta_n^{(d-1)} = \eta_n^{(d)} \qquad \forall d = D_n-1,\ldots,0. \tag{43}$$

Because $\eta_n^{(d)}(t) \le \mu_n^{(d)}(t)$ and $\sum_d \mu_n^{(d)}(t) = \mu_n(t)$ for all time, we have

$$\eta_n^{(d)} \le \mu_n^{(d)}, \sum_d \mu_n^{(d)} \le \overline{\mu}_n^{\Pi_P}. \tag{44}$$

Since the system is stable, i.e., $Q_n^{(-1)}(t)$ is stable, we must have

$$\beta_n^{(-1)} \le \eta_n^{(-1)} \le \mu_n^{(-1)}. \tag{45}$$

Summing (45) and (43) over $d=-1,\ldots,D_n-1$, using (44), using $\beta_n^{D_n-1} = \lambda_n$, and using the fact that Π_P stabilizes the system, we conclude that

$$\lambda_n \le \sum_{d=-1}^{D_n-1}\eta_n^{(d)} \le \sum_{d=-1}^{D_n-1}\mu_n^{(d)} \le \overline{\mu}_n^{\Pi_P}.$$

This shows that for any stabilizing predictive policy, one can find an equivalent stationary and randomized scheduling policy, which results in the same cost that can be expressed as (21), and generates the same service rates that must satisfy the constraint (22). Since f_{av}^* is defined to be the minimum cost over the entire class of such stationary and randomized schemes, we conclude that $f_{\text{av}}^{\boldsymbol{D}*} \ge f_{\text{av}}^*$. ■

APPENDIX E
PROOF OF THEOREM 5

Here, we prove Theorem 5. Our proof is based on the following theorem about the Backpressure algorithm [10].

Theorem 11: The Backpressure algorithm with any finite $\boldsymbol{Q}(0)$ achieves the following:

$$f_{\text{av}}^{BP} \le f_{\text{av}}^* + \epsilon B \quad Q_{\text{av}}^{BP} \le \frac{B + f^{\max}/\epsilon}{\eta}. \tag{46}$$

Here, $B \triangleq (N/2)(\mu_{\max}^2 + A_{\max}^2)$ is a constant independent of ϵ. f_{av}^{BP} and Q_{av}^{BP} denote the average expected cost and the average expected system queue size under Backpressure, respectively.◇

Proof: (Theorem 5) To prove the results, we consider the auxiliary system in Theorem 1, i.e., no prediction, $\hat{Q}_n(0) = \sum_{t=0}^{D_n-1}A_n(t)$, $\hat{A}_n(t) = A_n(t+D_n)$, $\hat{\mu}_n(t) = \sum_{d=-1}^{D_n-1}\mu_n^{(d)}(t)$, and $\hat{Q}_n(t)$ evolves according to (10).

Then, we construct the Backpressure algorithm for this auxiliary system. We define a Lyapunov function $L(t) = (1/2)\sum_n \hat{Q}_n(t)^2$ and define a one-slot Lyapunov drift as $\Delta(t) = \mathbb{E}\{L(t+1) - L(t)|\hat{\boldsymbol{Q}}(t)\}$. Using (10), we get

$$\Delta(t) + \frac{1}{\epsilon}\mathbb{E}\left\{f(t)|\hat{\boldsymbol{Q}}(t)\right\} \le B + V\mathbb{E}\left\{f(t)|\hat{\boldsymbol{Q}}(t)\right\} - \sum_n \hat{Q}_n(t)\mathbb{E}\left\{\hat{\mu}_n(t) - \hat{A}_n(t)|\hat{\boldsymbol{Q}}(t)\right\}. \tag{47}$$

By choosing the actions to minimize the right-hand side of (47), we see that Backpressure works as follows: At every time t, solve the following problem and perform the chosen action:

$$\min: \quad \frac{1}{\epsilon}f(\boldsymbol{S}(t),\boldsymbol{P}(t)) - \sum_n \hat{Q}_n(t)\mu_n\left(\boldsymbol{S}(t),\boldsymbol{P}(t)\right) \tag{48}$$

$$\text{s.t.} \quad \boldsymbol{P}(t) \in \mathcal{P}^{(\boldsymbol{S}(t))}. \tag{49}$$

Comparing this to (8), and using the fact that $\hat{Q}_n(t) = Q_n^{\text{sum}}(t)$, we conclude that applying `PBP` to the predictive system is equivalent to applying Backpressure to this auxiliary system. Therefore, Backpressure in the auxiliary system will choose the exact same control actions as `PBP` in the actual system. Since both systems have the same arrival and channel state processes, the two systems will evolve identically. Thus, the average power cost and the average queue size will be the same in both systems. Hence, Theorem 5 follows from Theorem 11. ■

APPENDIX F
PROOF OF THEOREM 6

We prove Theorem 6. For our proof, we use the following theorem (which is [15, Theorem 1]), in which $\boldsymbol{\gamma}^*$ denotes an optimal solution of (23). According to [15], $\boldsymbol{\gamma}^*$ is either $\Theta(V)$ or 0.

Theorem 12: Suppose: (i) $\boldsymbol{\gamma}^*$ is unique; (ii) the η-slack condition (18) is satisfied with $\eta > 0$; (iii) the function $g(\boldsymbol{\gamma})$ satisfies

$$g(\boldsymbol{\gamma}^*) \ge g(\boldsymbol{\gamma}) + L\|\boldsymbol{\gamma}^* - \boldsymbol{\gamma}\|, \quad \forall \boldsymbol{\gamma} \succeq \boldsymbol{0} \tag{50}$$

for some constant $L > 0$ independent of V. Then, under Backpressure, there exist constants $G, K, c = \Theta(1)$, i.e., all independent of V, such that for any $m \in \mathbb{R}_+$

$$\mathcal{P}^{(r)}(G, Km) \le ce^{-m} \tag{51}$$

where $\mathcal{P}^{(r)}(G, Km)$ is defined

$$\mathcal{P}^{(r)}(G, Km) \triangleq \limsup_{t\to\infty}\frac{1}{t}\sum_{\tau=0}^{t-1}\Pr\left\{\exists n, |Q_n(\tau) - \gamma_n^*| > G + Km\right\}. \quad \diamond \tag{52}$$

---

[14] The introduction of $\beta_n^{(d)}(t)$ is to help separate packets from queues in the presentation. We indeed have $\beta_n^{(d)}(t) = Q_n^{(d)}(t)$ for all $0 \le d \le D_n - 1$.

[15] Here, we assume these limits exist. Note that since $A_n(t)$, $\eta_n^{(d)}(t)$ and $\mu_n^{(d)}(t)$ are all bounded, these limits are equal to the sample path limits with probability 1 [27].

Proof: See [15]. ■

We are now ready to present the proof of Theorem 6.

Proof: (Theorem 6) We prove the results using Little's theorem. The main idea is to show that the average system queue length is roughly reduced by $\sum_n \lambda_n D_n$. To prove this, we show that the average total service rate allocated to the prediction queues is $O(\epsilon^{\log(1/\epsilon)})$. Then, the average rate of the packets that go through $\{Q_n^{(d)}(t)\}_{d=0}^{D_n-1}$ will roughly be λ_n, and so the average queue size is reduced by roughly $\sum_n \lambda_n D_n$.

First, using (11) and (51), we see that in steady state

$$\Pr\left\{|Q_n^{\text{sum}}(t) - \gamma_n^*| > G + Km\right\} \le ce^{-m}.$$

Using the fact that $Q_n^{\text{sum}}(t) = \sum_{d=-1}^{D_n-1} Q_n^{(d)}(t)$, we have

$$\Pr\left\{Q_n^{(-1)}(t) < \gamma_n^* - G - Km - \sum_{d=0}^{D_n-1} Q_n^{(d)}(t)\right\} \le ce^{-m}.$$

Now let $m = (\log(1/\epsilon))^2$. Since $\gamma_n^* = \Theta(1/\epsilon)$, we see that when $1/\epsilon$ is sufficiently large, we have

$$\begin{aligned} &\gamma_n^* - G - Km - \sum_d Q_n^{(d)}(t) \\ &= \Theta(V) - G - K\left(\log\left(\frac{1}{\epsilon}\right)\right)^2 - \sum_d Q_n^{(d)}(t) \\ &\overset{(a)}{\ge} \Theta\left(\frac{1}{\epsilon}\right) - G - K\left(\log\left(\frac{1}{\epsilon}\right)\right)^2 - D_n A_{\max} \\ &\overset{(b)}{\ge} \mu_{\max}. \end{aligned} \tag{53}$$

Here, (a) follows from the fact that $Q_n^{(d)}(t) \le A_{\max}$ for all $0 \le d \le D_n - 1$, and in (b) we use the fact that $1/\epsilon$ is sufficiently large and $D_n = O(1/A_{\max}[\gamma_n^* - G - K(\log(1/\epsilon))^2 - \mu_{\max}]^+)$ for all n. This shows that the probability for $Q_n^{(-1)}(t)$ to go below $\mu_{\max}$ is at most $ce^{-(\log(1/\epsilon))^2} = c\epsilon^{\log(1/\epsilon)}$.

Using the fact that under the FIFO queueing discipline, a prediction queue $Q_n^{(d)}(t)$ will be served only when $Q_n^{(-1)}(t) < \mu_{\max}$, we conclude that the average service rate allocated to the prediction queues is no more than $c\mu_{\max}\epsilon^{\log(1/\epsilon)}$. Hence, the average traffic rate of the packets that traverse all prediction queues and eventually enter $Q_n^{(-1)}(t)$ is at least $[\lambda_n - c\mu_{\max}\epsilon^{\log(1/\epsilon)}]^+$. Since every packet stays one slot in every prediction queue, using Little's theorem, we conclude that the average size of the prediction queues, denoted by $\sum_{d=0}^{D_n-1} \overline{Q}_n^{(d)}$ satisfies $\sum_{d=0}^{D_n-1} \overline{Q}_n^{(d)} \ge (\lambda_n - c\mu_{\max}\epsilon^{\log(1/\epsilon)})^+ D_n$. Hence, (26) follows. ■

APPENDIX G

PROOF OF THEOREM 7

We prove Theorem 7 in this appendix.

Proof: First of all, when conditions (i)–(iv) hold, we see from [28, Theorem 4] that, under Backpressure with LIFO (without prediction), for each queue n, there exist a set of packets $\mathbb{P}_n$ that have an average rate $\tilde{\lambda}_n$ given by

$$\tilde{\lambda}_n \ge \left[\lambda_n - O\left(\epsilon^{\log\left(\frac{1}{\epsilon}\right)}\right)\right]^+ \tag{54}$$

and packets in $\mathbb{P}_n$ experience an average delay $\text{Delay}_{\mathbb{P}_n}$ which satisfies

$$\text{Delay}_{\mathbb{P}_n} \le \text{DB}_n \triangleq \frac{A + B\left[\log\left(\frac{1}{\epsilon}\right)\right]^2}{\tilde{\lambda}_n} \tag{55}$$

where A and B are $\Theta(1)$ constants. Note here that if $\tilde{\lambda}_n = 0$, then $\lambda_n = O(\epsilon^{\log(1/\epsilon)})$ and the theorem follows. Hence, here we consider $\tilde{\lambda}_n > 0$.

Consider the system under `PBP` with LIFO. From the queue equivalence result in Theorem 1, we can also find a set of packets $\tilde{\mathbb{P}}_n$ in the predictive system, which also have an average rate of $\tilde{\lambda}_n$ and experience an average delay in $\{Q_n^{(d)}(t)\}_{d=-1}^{D_n-1}$ given by $\text{Delay}_{\tilde{\mathbb{P}}_n} \le \text{DB}_n$. Denote the set of packets that eventually enter $Q_n^{(-1)}(t)$ in the predictive system by $\tilde{\mathbb{P}}_n^{(-1)}$ and denote their rate by $\beta_n^{(-1)}$. We see then

$$\lambda_n = \beta_n^{(-1)} + \lambda_n^0. \tag{56}$$

Consider the packets in $\tilde{\mathbb{P}}_n \cap \tilde{\mathbb{P}}_n^{(-1)}$ and define their average rate by $\tilde{\beta}_n^{(-1)}$. Note that these are the packets that enter $Q_n^{(-1)}(t)$, but are taken into consideration when computing the rate and packet delay of $\tilde{\mathbb{P}}_n$ (some packets may not enter $\tilde{\mathbb{P}}_n^{(-1)}$ but are also included in computing the delay). Using (55), we see that

$$\text{DB}_n \ge \text{Delay}_{\tilde{\mathbb{P}}_n} \ge \frac{\tilde{\beta}_n^{(-1)} D_n}{\tilde{\lambda}_n}. \tag{57}$$

Here, (57) holds because the average number of packets from $\tilde{\mathbb{P}}_n \cap \tilde{\mathbb{P}}_n^{(-1)}$ is at least $\tilde{\beta}_n^{(-1)} D_n$, while $\tilde{\mathbb{P}}_n \cap \tilde{\mathbb{P}}_n^{(-1)}$ is a subset of $\tilde{\mathbb{P}}_n$. (57) implies that

$$\tilde{\beta}_n^{(-1)} \le \frac{A + B\left[\log\left(\frac{1}{\epsilon}\right)\right]^2}{D_n} = O\left(\frac{1}{\left[\log\left(\frac{1}{\epsilon}\right)\right]^{k-2}}\right).$$

Using (54), we conclude that

$$\beta_n^{(-1)} \le \tilde{\beta}_n^{(-1)} + O\left(\epsilon^{\log\left(\frac{1}{\epsilon}\right)}\right) = O\left(\frac{1}{\left[\log\left(\frac{1}{\epsilon}\right)\right]^{k-2}}\right). \tag{58}$$

Combining (56) and (58), we see that the result follows. ■

APPENDIX H

PROOF OF THEOREM 8

Here, we prove Theorem 8.

Proof: (Theorem 8) First we see that `POPBP` achieves the exact same utility performance as `PBP`. This is because both algorithms choose actions in the exact same way. Thus, (28) follows from Theorem 5. Similarly, (29) follows because the values of $Q_n^{(-1)}(t)$ are exactly the same under both algorithms.

To see the dropping rate, one sees that dropping happens only when $Q_n^{(-1)}(t)$ becomes empty. However, by choosing $D_n = O(1/A_{\max}[\gamma_n^* - G - K(\log(1/\epsilon))^2 - \mu_{\max}]^+)$, the probability that $Q_n^{(-1)}(t)$ becomes empty is bounded by $ce^{-(\log(1/\epsilon))^2} = c\epsilon^{\log(1/\epsilon)}$ as in the proof of Theorem 6. Since in every time-slot, at most $\mu_{\max}$ packets can be marked as mistaken, we see that the drop rate r_n^d is at most $O(c\epsilon^{\log(1/\epsilon)})$. Combing this result with the fact that $z_n(r_n^d) \le \xi r_n^d$, we see that the total additional system cost is $O(\xi\epsilon)$. ■

REFERENCES

[1] M. Maia, J. Almeida, and V. Almeida, "Identifying user behavior in online social networks," in *Proc. 1st Workshop Social Netw. Syst.*, 2008, pp. 1–6.

[2] I. Weber and A. Jaimes, "Who uses Web search for what? And how?," in *Proc. WSDM*, 2011, pp. 21–30.

[3] R. Kumar and A. Tomkins, "A characterization of online browsing behavior," in *Proc. 19th Int. Conf. World Wide Web*, 2010, pp. 561–570.

[4] V. N. Padmanabhan and J. C. Mogul, "Using predictive prefetching to improve world wide web latency," *Comput. Commun. Rev.*, vol. 26, no. 3, pp. 22–36, Jul. 1996.

[5] J. Lee, H. Kim, and R. Vuduc, "When prefetching works, when it doesn't, and why," *Trans. Archit. Code Optimiz.*, vol. 9, no. 1, Mar. 2012, Art. no. 2.

[6] T. Ball and J. R. Larus, "Branch prediction for free," in *Proc. Conf. Program. Lang. Design Implement., ACM SIGPLAN Notices*, 1993, vol. 28, pp. 300–313.

[7] M. U. Farooq Khubaib and L. K. John, "Store-load-branch (SLB) predictor: A compiler assisted branch prediction for data dependent branches," in *Proc. 19th IEEE HPCA*, Feb. 2013, pp. 59–70.

[8] R. Berry and R. Gallager, "Communication over fading channels with delay constraints," *IEEE Trans. Inf. Theory*, vol. 48, no. 5, pp. 1135–1149, May 2002.

[9] M. J. Neely, "Optimal energy and delay tradeoffs for multi-user wireless downlinks," *IEEE Trans. Inf. Theory*, vol. 53, no. 9, pp. 3095–3113, Sep. 2007.

[10] M. J. Neely, "Energy optimal control for time-varying wireless networks," *IEEE Trans. Inf. Theory*, vol. 52, no. 7, pp. 2915–2934, Jul. 2006.

[11] A. C. Fu, E. Modiano, and J. N. Tsitsiklis, "Optimal energy allocation and admission control for communications satellites," *IEEE/ACM Trans. Netw.*, vol. 11, no. 3, pp. 488–500, Jun. 2003.

[12] M. A. Zafer and E. Modiano, "A calculus approach to energy-efficient data transmission with quality-of-service constraints," *IEEE/ACM Trans. Netw.*, vol. 17, no. 3, pp. 898–911, Jun. 2009.

[13] C. W. Tan, D. P. Palomar, and M. Chiang, "Energy-robustness tradeoff in cellular network power control," *IEEE/ACM Trans. Netw.*, vol. 17, no. 3, pp. 912–925, Jun. 2009.

[14] M. J. Neely, "Super-fast delay tradeoffs for utility optimal fair scheduling in wireless networks," *IEEE J. Sel. Areas Commun.*, vol. 24, no. 8, pp. 1489–1501, Aug. 2006.

[15] L. Huang and M. J. Neely, "Delay reduction via Lagrange multipliers in stochastic network optimization," *IEEE Trans. Autom. Control*, vol. 56, no. 4, pp. 842–857, Apr. 2011.

[16] J. Tadrous, A. Eryilmaz, and H. El Gamal, "Proactive resource allocation: Harnessing the diversity and multicast gains," *IEEE Trans. Inf. Theory*, vol. 59, no. 8, pp. 4833–4854, Aug. 2013.

[17] J. Tadrous, A. Eryilmaz, and H. El Gamal, "Pricing for demand shaping and proactive download in smart data networks," in *Proc. 2nd IEEE INFOCOM SDP*, 2013, pp. 321–326.

[18] J. Spencer, M. Sudan, and K. Xu, "Queueing with future information," ArXiv Tech. Rep. arxiv:1211.0618, 2012.

[19] Y. Yao, L. Huang, A. Sharma, L. Golubchik, and M. J. Neely, "Data centers power reduction: A two time scale approach for delay tolerant workloads," USC CS Tech. Rep. 11-9020, 2011.

[20] I. Hou and P. R. Kumar, "Broadcasting delay-constrained traffic over unreliable wireless links with network coding," in *Proc. MobiHoc*, 2011, Art. no. 4.

[21] L. Georgiadis, M. J. Neely, and L. Tassiulas, "Resource allocation and cross-layer control in wireless networks," *Found. Trends Netw.*, vol. 1, no. 1, pp. 1–144, 2006.

[22] L. Huang and M. J. Neely, "Max-weight achieves the exact $[O(1/V), O(V)]$ utility-delay tradeoff under Markov dynamics," arXiv:1008.0200v1, 2010.

[23] A. Wierman M. Lin, Z. Liu, and L. Andrew, "Online algorithms for geographical load balancing," in *Proc. IGCC*, 2012, pp. 1–10.

[24] S. Zhang, L. Huang, M. Chen, and X. Liu, "Effect of proactive serving on user delay reduction in service systems," in *Proc. ACM SIGMETRICS*, Jun. 2014, (Poster Paper).

[25] R. G. Gallager, *Discrete Stochastic Processes*. Norwell, MA, USA: Kluwer, 1996.

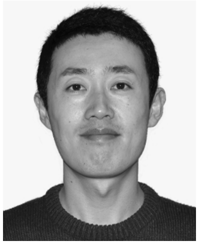
[26] A. Rao *et al.*, "Network characteristic of video streaming traffic," in *Proc. ACM CoNEXT*, 2011, Art. no. 25.

[27] G. B. Folland, *Real Analysis: Modern Techniques and Their Applications*, 2nd ed. New York, NY, USA: Wiley, 1999.

[28] L. Huang, S. Moeller, M. J. Neely, and B. Krishnamachari, "Lifo-backpressure achieves near optimal utility-delay tradeoff," *IEEE/ACM Trans. Netw.*, vol. 21, no. 3, pp. 831–844, Jun. 2013.

Longbo Huang received the B.E. degree from Sun Yat-sen University, Guangzhou, China , in 2003, the M.S. degree from Columbia University, New York, NY, USA, in 2004, and the Ph.D. degree from the University of Southern California, Los Angeles, CA, USA, in 2011, all in electrical engineering.

He then worked as a Postdoctoral Researcher with the Electrical Engineering and Computer Science Department, University of California, Berkeley, CA, USA, from 2011 to 2012. Since 2012, he has been an Assistant Professor with the Institute for Interdisciplinary Information Sciences (IIIS), Tsinghua University, Beijing, China. He was selected into China's Youth 1000-talent program in 2013. His current research interests are in the areas of learning and optimization for networked systems, data center networking, smart grid, and mobile networks.



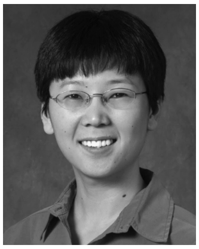
Shaoquan Zhang received the B.Eng. degree from the University of Science and Technology of China, Hefei, China, in 2008, and the Ph.D. degree in information engineering from The Chinese University of Hong Kong, Hong Kong, in 2014.

He is currently with Hong Kong Applied Science and Technology Research Institute (ASTRI), Hong Kong. His research interests include system analysis, algorithm design, and distributed and stochastic network optimization.

Minghua Chen (S'04–M'06–SM'13) received the B.Eng. and M.S. degrees in electronic engineering from Tsinghua University, Beijing, China, in 1999 and 2001, respectively, and the Ph.D. degree in electrical engineering and computer sciences from the University of California (UC), Berkeley, CA, USA, in 2006.

He spent one year visiting Microsoft Research, Redmond, WA, USA, as a Postdoctoral Researcher. He joined the Department of Information Engineering, The Chinese University of Hong Kong, 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 recent research interests include energy systems (e.g., microgrids and energy-efficient data centers), distributed optimization, multimedia networking, wireless networking, network coding, and delay-constrained communication.

Dr. Chen is currently an Associate Editor of the IEEE/ACM Transactions on Networking. 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.



Xin Liu received the Ph.D. degree in electrical engineering from Purdue University, West Lafayette, IN, USA, in 2002.

She is currently a Professor with the Computer Science Department, University of California (UC), Davis, CA, USA. From 2012 to 2014, she was with the Wireless Networking Group, Microsoft Research Asia, Beijing, China. Her current research focuses on data-driven approach in networking.

Prof. Liu became a Chancellor's Fellow in 2011. She received the Best Paper of the Year Award of *Computer Networks* in 2003, the NSF CAREER Award in 2005, and the Outstanding Engineering Junior Faculty Award from the College of Engineering, UC Davis, in 2005.