Virtualized Control Over Fog: Interplay Between Reliability and Latency

Hazer Inaltekin, *Member, IEEE*, Maria Gorlatova, *Member, IEEE*, and Mung Chiang, *Fellow, IEEE*

***Abstract*—This paper introduces an analytical framework to investigate optimal design choices for the placement of virtual controllers along the cloud-to-things continuum. The main application scenarios include low-latency cyber-physical systems in which real-time control actions are required in response to the changes in states of an Internet of Things (IoT) node. In such cases, deploying controller software on a cloud server is often not tolerable due to delay from the network edge to the cloud. Hence, it is desirable to trade reliability with latency by moving controller logic closer to the network edge. Modeling the IoT node as a dynamical system that evolves linearly in time with quadratic penalty for state deviations, recursive expressions for the optimum control policy and the resulting minimum cost value are obtained by taking virtual fog controller reliability and response time latency into account. Our results indicate that latency is more critical than reliability in provisioning virtualized control services over fog endpoints, as it determines the swiftness of the fog control system as well as the timeliness of state measurements. Based on a drone trajectory tracking model, an extensive simulation study is also performed to illustrate the influence of reliability and latency on the control of autonomous vehicles over fog.**

***Index Terms*—Control, distributed systems, fog computing, Internet of Things (IoT), latency, reliability.**

I. Introduction

FOG computing, sometimes referred to as edge computing, is an emerging computing paradigm in which computing, storage, networking, and control are placed at multiple locations between the endpoint devices and the cloud [1], [2]. Levine [3], a partner at an A-list venture capital firm Andreessen Horowitz, has recently called fog computing *the next multibillion dollar tech market*. The promise of fog computing for enabling the next generation of advances in IoT is underscored by the growing developments of fog computing architectures [4], [5] and ongoing industry-wide standardization efforts [6], [7].

Fog computing offers flexibility in the choice of *virtualized* controller placement options for interactive control applications, as has been proposed in the outlines of the vision of the future of the industry [1], [2], [6]. Furthermore, fog/cloud architecture is also starting to be considered from a practical point of view for futuristic control applications, e.g., moving vehicular controls to different locations is proposed in [8] and [9]. However, while multiple virtual controller placements are starting to become possible in practice [10]–[13], the theoretical foundations for these placement decisions are currently lacking. We take steps toward addressing this gap in this paper.

In particular, this paper focuses on *latency* and *reliability* aspects that arise in a fog computing environment because different virtual controller locations in a fog hierarchy may exhibit different latency and reliability characteristics [2]. For example, fog logic execution points may include local nodes and a wide variety of remote ones, as shown in Fig. 1 (i.e., both Amazon Web Service (AWS) Greengrass [4] and Microsoft Azure IoT Edge [5] allow executing functions both locally and remotely). In these settings, local devices provide low response latency but may not always be reliable. Remote cloud computing nodes, on the other hand, offer considerably longer response times [14] but can be readily designed to guarantee high reliability. Then, what is the optimum design choice for placing controller software to maximize system performance? Critical to resolving this question is the discovery of the interplay between latency and reliability in control applications over fog, which is what the current paper achieves for linear IoT systems with a quadratic cost.

Our analytical framework applies to IoT systems with linear feedback controllers, which are studied in a wide variety of applications [15], [16], that can be virtualized over the fog endpoints. In particular, the trajectory following control for flying drones is a notable example of a control functionality that can be virtualized in different locations in a fog computing system, as shown in Fig. 2, and hence our results can be applied to. Motivated by the advances in quadcopter technology and by the commercial promise of autonomous drone operations, such as Amazon's plan to deliver packages using drones [17], various aspects of drone operations are actively studied [18], [19]. For example, in drone air traffic control,

Manuscript received February 13, 2018; revised September 15, 2018; accepted October 26, 2018. Date of publication November 13, 2018; date of current version January 16, 2019. This work was supported in part by the Comcast Innovation Fund Research Grant, AWS Cloud Credits for Research, Microsoft Azure Research Award, NSF CSR-1812797 grant, NSF Waterman Award under Grant 1759652, NSF NeTS Award under Grant 1759656, and Defense Advanced Research Projects Agency (DARPA) under Contract No. HR001117C0052 and No. HR001117C0048. The opinions, findings and conclusions expressed in this material are those of the author(s) and do not necessarily reflect the views of the Defense Advanced Research Projects Agency. *(Corresponding author: Hazer Inaltekin.)*

H. Inaltekin is with the Department of Electrical and Electronic Engineering, University of Melbourne, Parkville, VIC 3010, Australia (e-mail: hazeri@unimelb.edu.au).

M. Gorlatova is with the Department of Electrical and Computer Engineering, Duke University, Durham, NC 27708 USA (e-mail: maria.gorlatova@duke.edu).

M. Chiang is with the School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47907 USA (e-mail: chiang@purdue.edu).

Digital Object Identifier 10.1109/JIOT.2018.2881202

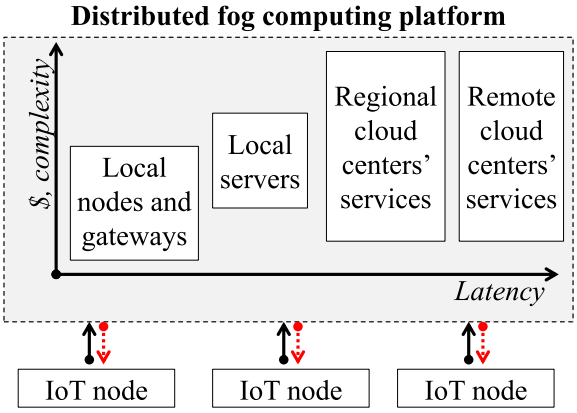

Fig. 1. In a fog computing system, control application can be run at different distributed points and as different services, with different characteristics.

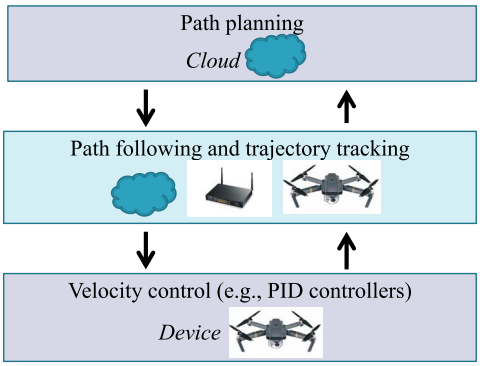

Fig. 2. Layers of control functionality for drones. While path planning belongs on the cloud and velocity control probably belongs on the device itself, trajectory tracking elements could be virtualized on different fog endpoints.

the highest-level global fleet planning decisions require the involvement of the cloud and the low-level high-bandwidth velocity control needs to be done on the drone itself [20], as illustrated by the top and bottom layers in Fig. 2. On the other hand, the important trajectory tracking and path following control operations [21], [22] can be executed on multiple locations in a fog network, as illustrated by the middle layer in Fig. 2.

To the best of our knowledge, this paper is the first systematic study to shed light on the interplay between reliability and latency appearing in virtual control services offered over fog networks. Our main contributions can be summarized as follows.

1) We propose an analytical framework to investigate the effects of latency and reliability on controlling linear IoT processes, disturbed by stochastic environmental factors, by means of a controller software located along the cloud-to-things continuum. Under this framework, the min-cost performance of virtualized control services is obtained.
2) We derive the structure of optimum virtual controllers by considering reliability and latency (both communication and computation) characteristics of the fog endpoint which will execute the controller application. In addition to increased response times between consecutive control actions, we show that an estimator, separated from control, must first be run for distant fog endpoints to estimate live IoT node states from delayed sensor inputs. This collateral effect of latency further decreases the efficacy of software-defined control over imperfectly placed fog endpoints.
3) Based on a drone trajectory tracking model, we conduct extensive simulations to visualize the performance of virtual fog controllers. It is observed that the path following efficiency decreases more quickly with latency than reliability due to its direct and collateral effects (i.e., increased response times and state estimation problem), which suggests to move the controller software as close as possible to the unmanned aerial vehicle (UAV).

The remainder of this paper is organized as follows. In Section II, we compare and contrast our results with related work. In Section III, we elaborate on important properties of fog computing architectures, and present small-scale results related to latency and reliability in fog computing. In Section IV, we introduce the analytical framework to investigate reliability and latency for virtualized control services over fog. In Section V, we derive the structure of the optimum virtual controller without latency, while Section VI contains parallel results for the optimum virtual controller with latency. In Section VII, we present our simulation results for the UAV trajectory tracking problem. In Section VIII, we provide a further discussion of our results and other potential applications. Section IX concludes this paper with future generalizations.

II. Related Work

This paper focuses on fog computing and linear IoT control systems. In this section, we describe the previous work that is most relevant to our technical results. For further discussion on linear modeling of IoT node processes and potential applications, we refer the reader to Section VIII.

Our results in this paper are related to both the emerging body of papers in fog computing [12], [13], [23]–[26] and the more classical literature in control systems [10], [11], [27]–[32]. The papers [12] and [13] focused on the development of fog computing platforms for smart-city and smart-home applications with several control functionalities virtualized either in street cabinets [12] or at the control panel located inside a home [13]. Although these papers provide insightful system implementation showcases to illustrate the utility of fog computing, they do not take any analytical approach, as we do in this paper, to substantiate their design choices.

The papers [23]–[26] studied how to adapt services for fog computing by mainly focusing on computational load offloading and associated computing job scheduling. In particular, Tong *et al.* proposed a hierachical edge/cloud architecture in [23], and showed that the proposed architecture has a higher chance of serving peak loads from virtualized services. Tan *et al.* [24] studied online algorithms for minimizing total weighted response time for edge-cloud networks with upload and download delays. An important feature of their online algorithm is that its performance comes close to the optimal offline algorithm with speed augmentation and without requiring any ex-ante knowledge of job arrival statistics.

Xiao and Krunz [25] investigated a job offloading problem similar to those studied in [23] and [24], but by considering the interest of both fog endpoints and users. Specifically, they optimized response times subject to power efficiency constraints of fog nodes and showed that cooperation among fog nodes has the potential to improve service execution times. Kosta *et al.* [26] developed a novel mobile cloud computing platform to migrate smartphone applications to virtual machines running on the cloud in an attempt to improve mobile computing and energy efficiency at the network edge.

When compared to [23]–[26], we take a simpler but more fundamental approach in this paper. By focusing on virtual control services, we examine the problem of where to place the codebase for a *single* controller application along the cloud-to-things continuum. For each value of reliability and latency parameters, we obtain the min-cost performance of the optimum virtual fog controller. These optimum performance figures can then be inputed to a wider system-level fog optimization problem as in [23]–[25] to determine how to dispatch and schedule a multitude of virtual control services to maximize a collective system utility, which we plan to pursue as a future research direction.

On the side of control systems, the papers [10] and [11] considered virtualized control services over the cloud. Liberatore [27] investigated an integrated play-back mechanism to improve the efficiency of remote control over the network. In [28], they studied the design of a physical control system over a wireless link that can corrupt transmitted data. These papers, however, do not employ any optimization framework to compute the structure of a virtual controller to be run at a fog endpoint. The papers [29]–[32], on the other hand, adopted a more optimization theory-based approach to design control systems over communication channels either allowing opportunistic transmissions [29], [30] or dropping packets randomly [31], [32]. The main point of difference of the current paper from [29]–[32] is that we focus on the virtual controller placement over fog by considering reliability and latency dimensions simultaneously. We show that the optimum virtual fog controller runs an estimator as a delay compensator, which does not appear in these papers. Further, the issue of reliability in our model is shifted from communication links to virtual fog controllers.

III. Fog Computing Architectures for Control-as-a-Service Applications

In this section, we describe the properties of fog computing architectures that are important for placements of virtualized control services, and present the results of small-scale experiments in a fog computing system.

A. Fog Services: Heterogeneity and the Need for Auto-Tuning

Fog computing architectures are expected to include different physical links (e.g., wired, wireless, and satellite), different extends of mobility of different nodes, and a wide range of differences in computing device capabilities [6].

We expect the functionality in fog systems to be provided via *service execution options with different performance parameters* (providing services for control applications can be referred to as creating *control-as-a-service architectures* [8], [9]). Cloud computing service providers are already offering a full range of service options that differ in the speed, cost, and complexity of execution [33], [34]—the diversification that is likely to become more and more prominent in the future. Due to the inherently heterogenous nature of fog systems, we expect them to include a wider range of service execution options than the options provided in traditional cloud computing systems. In particular, we expect services in fog systems to be offered at a range of reliability options, starting from expensive high-availability services with "five nines" uptime guarantees (i.e., 99.999% availability, or the downtime of no more than 5.2 min per year) [35], to cheaper limited or frequently interrupted services provided by low-end nodes, including nodes with long sleep cycles and energy-harvesting-based intermittently powered nodes [36], [37].

Additionally, virtualized control functionality in fog systems will need to be *placed, tuned, and moved around* automatically, without requiring inputs from the users. Existing commercial examples of automatic service placements in cloud computing include serverless computing mechanisms [34], [38]–[40], which use auto-provisioning ("autoscaling") mechanisms to provide robustness to spikes in service request rates at the cost of additional latency [41]. In distributed heterogeneous fog computing settings, we will also require the ability to shift task execution point assignments to save energy, optimize deployment costs, and to free up constrained resources for critical tasks and services. Enabling the provision of such auto-optimizing virtual control services necessitates obtaining quantitative understanding of the tradeoffs between different control system design parameters that are not generally considered simultaneously.

B. Latency and Reliability Tradeoffs: Small-Scale Experiments

To better understand latency-reliability tradeoffs in fog computing systems, we conducted small-scale experiments with simple linear virtual controllers.

The control service we implemented received, as inputs, a 2-D vector and a time index, looked up a corresponding 2-by-2 matrix, performed matrix multiplication, and returned the results. This service implementation closely follows the controller operations we examine in this paper. We executed the control application at different AWS Lambda [34] cloud computing service points worldwide, at a Microsoft Azure (serverless) Functions computing point [39], and on a local consumer-grade hardware device with an Intel Atom single-core 1.6 GHz CPU. The local control service was implemented using a popular Flask [42] micro-service development framework over the built-in Flask HTTP server and over a Gunicorn WSGI HTTP Unix Server [43].

Our small-scale experiments demonstrated the expected richness of virtualized fog control services in the latency-reliability space. Specifically, our response latency measurements, summarized in Table I, demonstrate that response times

TABLE I
LATENCY OF CONTROL SERVICE EXECUTION IN DIFFERENT NODES

#	Control service placement	Latency [secs]
1	Local node	0.06
2	Microsoft Azure Functions US East	0.08
3	AWS Lambda US East (Virginia)	0.5
4	AWS Lambda US West (Seattle)	0.8
5	AWS Lambda Tokyo	1.3

vary over a wide range.[1] We observed a wide variety of reliability options in fog settings as well. Serverless AWS Lambda and Microsoft Azure Functions computing are provisioned on demand and hence can be seen as always available. Local control service, on the other hand, has only a finite number of control service processes deployed, and thus can handle only a fixed number of responses at the same time. Thus, as expected, we observed in our experiments with both default Flask server settings and with the Gunicorn server that when the instantiated processes are occupied, the responses can be deemed to be dropped in time-critical control applications due to large waiting times behind others. It is clear that these differences in reliability and latency are likely to be even more dramatic in *fully* heterogeneous fog computing architectures outlined above.

These experimental observations as well as the drone trajectory following problem provide the underlying motivation for this paper to undertake a systematic study for delineating the direct and collateral effects of reliability and latency on virtual control services over fog.

IV. VIRTUALIZED CONTROL OVER FOG

In this section, we will introduce the details of our system model, definitions of the main concepts in relation to this model and the virtualized control problem over fog.

A. System Model

We consider an IoT node (such as a UAV, a robotic arm, etc.) whose dynamics evolve *linearly* in discrete-time according to

$$\boldsymbol{x}_{k+1} = \boldsymbol{A}_k \boldsymbol{x}_k + \boldsymbol{B}_k \boldsymbol{u}_k + \boldsymbol{w}_k \tag{1}$$

where $\boldsymbol{x}_k \in \mathbb{R}^n$ is the state vector, $\boldsymbol{u}_k \in \mathbb{R}^s$ is the control signal, $\boldsymbol{w}_k \in \mathbb{R}^n$ is the zero-mean random disturbance (independent over time) with a covariance matrix $\boldsymbol{W}_k$, and $\boldsymbol{A}_k \in \mathbb{R}^{n\times n}$ and $\boldsymbol{B}_k \in \mathbb{R}^{n\times s}$ are the system matrices that modulate the system states and control signals, respectively, for $k = 0, \ldots, N-1$.[2] The IoT node does not possess an on-board controller circuitry, and hence the controller functionality is virtualized on a fog endpoint, as shown in Fig. 3. We note that the model in (1) is general enough to include the cases in which the control information is not precise. In particular, if the control signal received by the IoT node at time k is distorted by the zero-mean additive noise $\boldsymbol{\xi}_k$ according to $\boldsymbol{u}_k + \boldsymbol{\xi}_k$, then the IoT node state at time $k+1$ is given by $\boldsymbol{x}_{k+1} = \boldsymbol{A}_k \boldsymbol{x}_k + \boldsymbol{B}_k \boldsymbol{u}_k + \boldsymbol{B}_k \boldsymbol{\xi}_k + \boldsymbol{w}_k$. Hence, the analysis of the system with randomly distorted control signal $\boldsymbol{u}_k + \boldsymbol{\xi}_k$ is the same with that of the system without any disturbance at control signals but having the random disturbance $\boldsymbol{B}_k \boldsymbol{\xi}_k + \boldsymbol{w}_k$ at the IoT node. The states are monitored by the on-board sensors and the measurements are transmitted over a communication channel to the fog controller. The received data at the output of the channel is modeled by

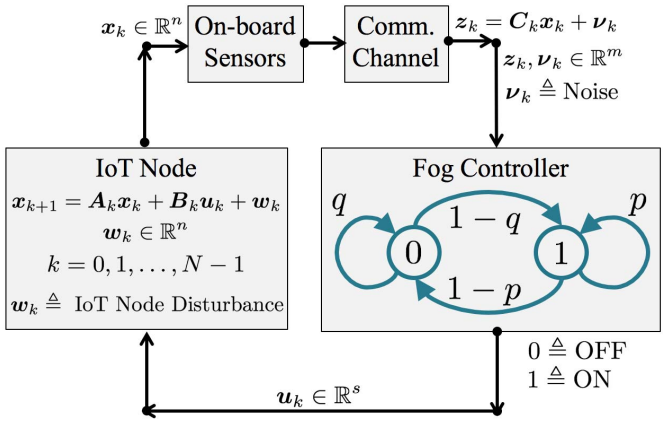

Fig. 3. Model for virtualized control over fog.

$$\boldsymbol{z}_k = \boldsymbol{C}_k \boldsymbol{x}_k + \boldsymbol{\nu}_k \tag{2}$$

where $\boldsymbol{C}_k \in \mathbb{R}^{m\times n}$ is the measurement matrix (alternatively called sensor matrix) and $\boldsymbol{\nu}_k \in \mathbb{R}^m$ is the measurement-plus-channel noise. Several remarks are in order about this model. First, this model is mathematically more general than the case $\boldsymbol{z}_k = \boldsymbol{x}_k$, which we call the fully observed state information case in the sequel. With partially observed state information (i.e., $\boldsymbol{z}_k \neq \boldsymbol{x}_k$), the IoT control system we have can be considered to be in the form of a hidden Markov model (HMM). In general, HMMs have a broad range of applications in communications, signal processing and robotics [45]–[47]. Second, the measurement matrix is not necessarily in the form of a square matrix since some of the states may not be measured by means of available sensor types [15], [16]. For example, in the drone trajectory tracking problem, only location measurements may be available, while the state vector contains both drone location and velocity values. Finally, it is expected that $\boldsymbol{\nu}_k$ mostly contains the measurement noise since the communication noise is robustly handled by the lower layers in modern digital communication systems [48]. However, the communication process is still expected to cause additional distortion in the measured data due to quantization, lossy data compression (for communication over channels with limited capacity), and time-frequency uncertainty [49]–[51]. Therefore, we will continue to call $\boldsymbol{\nu}_k$ measurement-plus-channel noise in the

[1] Our control service code was not optimized for speed; service execution latency can be improved with additional software engineering effort. However, latency variation across different execution points would be a factor for latency-optimized code as well, due to network, service invocation, and underlying hardware platform differences between the different execution points.

[2] This paper does not consider how the time discretization is performed, which depends on the time-scale of change of the IoT process to be controlled as well as on other several design degrees of freedom such as control quality and precision. For nonlinear IoT node processes, it is assumed that Hartman–Grobman theorem holds and the system dynamics can be linearized around an equilibrium point [44].

remainder of this paper.[3] It is assumed that system disturbance and measurement-plus-channel noise vectors are independent among themselves as well as being independent over time. It is also assumed that reception of z_k marks a *request-for-control* and initiates the generation of a control signal at the fog endpoint.

The causal ordering of events to update IoT node states is the state measurement, forward delivery of measured states to the virtual fog controller, generation of control signals and backward delivery of control signals to the IoT node. When all these events happen in the same time-slot, we say that the fog controller is *perfectly matched* to the IoT node dynamics. Otherwise, we say that it is *imperfectly matched*, in which case the control signals lag behind the IoT node state updates. To simplify the exposition, we will explain the rest of the system model for the case of perfect match, and relegate the details of the latter to Section VI.

We model the reliability issues observed in our fog computing experiments through a stylized Markov process having two states with transition probabilities

$$\mathsf{Pr}\big\{\tau_{k+1}=1\big|\tau_k=1\big\}=p \tag{3}$$

and

$$\mathsf{Pr}\big\{\tau_{k+1}=0\big|\tau_k=0\big\}=q \tag{4}$$

where τ_k is the internal state of the fog controller at time $k=0,\ldots,N-1$. Here, the states 1 (i.e., ON state) and 0 (i.e., OFF state) indicate the capacitated occupancy status (due to multiple instantiated computing processes) of the fog endpoint for provisioning the requested control service. We note that (3) and (4) are enough to identify the Markov chain transitions since the state transition probability from ON state to OFF state is $1-p$, and the one from OFF state to ON state is $1-q$.

The control $\boldsymbol{u}_k$ depends on the available information at the fog controller by time k, which will be denoted as

$$H_k=\{z_i:\tau_i=1, 0\le i\le k\}\bigcup\{\boldsymbol{u}_i:0\le i\le k-1\}. \tag{5}$$

We define $\mathcal{H}_k$ to be the collection of all possible information sets available at time k. Note that the fog controller knows the previous control signals when generating $\boldsymbol{u}_k$ at time k, which is embodied in H_k. The class of dynamic control policies of interest to us in this paper is introduced in Definition 1.

Definition 1: A control policy is a sequence of functions $\boldsymbol{\pi}=(\pi_0,\ldots,\pi_{N-1})$ such that the kth component function $\pi_k:\mathcal{H}_k\times\{0,1\}\mapsto\mathbb{R}^s$ determines the control applied at time $k=0,\ldots,N-1$, i.e., $\pi_k(H_k,\tau_k)=\boldsymbol{u}_k$.

We note that $\boldsymbol{\pi}$ is an online rule that observes the realizations of system history and determines the control signals to be applied based on these observations. In this paper, we will only focus on a class of control policies that always output zero as the control signal when they are at the OFF state, i.e., $\pi_k(H_k,\tau_k)=0$ if $\tau_k=0$. Accordingly, we will say that a control policy $\boldsymbol{\pi}$ is *feasible* for the optimum stochastic control problem we want to solve if it belongs to this class of control policies, i.e., $\pi_k(H_k,0)=0$ for all $H_k\in\mathcal{H}_k$ and $k=0,\ldots,N-1$. To put it another way, a feasible control policy does not output any control signal when the fog controller is at the OFF state. We note that the class of feasible control policies contains the ones that can output zero even when the fog controller is not switched off. For example, it contains the one that always outputs zero as the control signal regardless of the internal state of the fog controller. Despite its richness and spanning an infinite dimensional functional space, we also note that our definition of feasibility in this paper does not include an important and practical set of control policies that can produce nonzero control signals at the OFF state such as those holding the last control value. An important subclass of control policies that we consider in this paper is that of memoryless ones, formally defined in Definition 2.

Definition 2: A control policy is said to be *memoryless* if the control applied at time k depends only on z_k and τ_k for $k=0,\ldots,N-1$.

B. Cost Minimization Problem

As is standard in control systems [16], [30]–[32], [53]–[55], we rank the quality of virtualized control over fog by means of a quadratic cost function. In particular, we define the per-stage-cost under control policy $\boldsymbol{\pi}=(\pi_0,\ldots,\pi_{N-1})$ at time k as

$$g_{\pi_k}(\boldsymbol{x}_k,H_k,\tau_k)=\boldsymbol{x}_k^\top\boldsymbol{Q}_k\boldsymbol{x}_k+\boldsymbol{u}_k^\top\boldsymbol{R}_k\boldsymbol{u}_k \tag{6}$$

where $\boldsymbol{u}_k$ is the control generated by $\boldsymbol{\pi}$ after having observed H_k, and $\boldsymbol{Q}_k$ is positive semidefinite and $\boldsymbol{R}_k$ positive definite for all k. Over a finite horizon of $N+1$ time-slots, the aggregate cost depends on three sources of randomness: 1) IoT node disturbance; 2) measurement-plus-channel noise; and 3) fog controller reliability. Hence, we write the total cost incurred over the time horizon of interest and averaged over the existing sources of randomness as

$$J_{\boldsymbol{\pi}}^{(p,q)}(\boldsymbol{x}_0,\tau_0)=\mathsf{E}_{(\boldsymbol{x}_0,\tau_0)}\left[\boldsymbol{x}_N^\top\boldsymbol{Q}_N\boldsymbol{x}_N+\sum_{k=0}^{N-1}g_{\pi_k}(\boldsymbol{x}_k,H_k,\tau_k)\right] \tag{7}$$

where $\boldsymbol{x}_0$ and τ_0 are the initial states and $\mathsf{E}_{(\boldsymbol{x}_0,\tau_0)}$ indicates expectation starting from these initial sates. Our aim is to minimize $J_{\boldsymbol{\pi}}^{(p,q)}(\boldsymbol{x}_0,\tau_0)$ over the set of all feasible control policies Π, i.e.,

$$J^{\star(p,q)}(\boldsymbol{x}_0,\tau_0)=\inf_{\boldsymbol{\pi}\in\Pi}J_{\boldsymbol{\pi}}^{(p,q)}(\boldsymbol{x}_0,\tau_0). \tag{8}$$

We will solve the optimization problem in (8) by utilizing the dynamic programming (DP) approach [16], [56]. To this end, we define cost-to-go functions as

$$J_i^{(p,q)}(\boldsymbol{x}_i,\tau_i)=\mathsf{E}_{(\boldsymbol{x}_i,\tau_i)}\left[\boldsymbol{x}_N^\top\boldsymbol{Q}_N\boldsymbol{x}_N+\sum_{k=i}^{N-1}g_{\pi_k}(\boldsymbol{x}_k,H_k,\tau_k)\right]. \tag{9}$$

We note that $J_i^{(p,q)}(\boldsymbol{x}_i,\tau_i)$ depends on the ith tail policy $\boldsymbol{\pi}_i$ for $\boldsymbol{\pi}$, which is defined as $\boldsymbol{\pi}_i=(\pi_i,\ldots,\pi_{N-1})$. The principle of optimality for DP states that the control policy $\boldsymbol{\pi}^\star$ having

[3] We also note the affinity of this model with the classical multiple-input multiple-output wireless channel model [52]. Hence, our model can be alternatively considered to exemplify uncoded or low-complexity coded wireless transmissions such as simple CRC schemes so that some channel noise still inflicts the received state measurements, which is a practical assumption for the IoT nodes limited by computational resources and battery life.

optimal tail policies $\boldsymbol{\pi}_i^\star = (\pi_i^\star, \ldots, \pi_{N-1}^\star)$ for minimizing the cost-to-go function $J_i^{(p,q)}(\boldsymbol{x}_i, \tau_i)$ for any starting state $\boldsymbol{x}_i \in \mathbb{R}^n$ and $\tau_i \in \{0, 1\}$ for all $i \in \{0, \ldots, N-1\}$ solves the main DP problem in (8). We will utilize this fact for revealing the structure of the optimum control in both cases of perfectly and imperfectly matched fog controllers.

V. Control Over Fog With Perfect Match

In this section, we will investigate the performance of virtual controllers with the focus of reliability being on the fog endpoint that is perfectly matched to the IoT node dynamics. Despite following the standard DP steps [16], [56], our analysis in this section will be helpful to set the stage for the case of imperfectly matched fog controllers, which can only access to the delayed and intermittent state information from the IoT node. We will consider the cases of fully observed and partially observed state information separately, starting with the former one below.

A. Fully Observed State Information

This is the case in which the fog controller perfectly observes the measured states, i.e., $\boldsymbol{v}_k = \boldsymbol{0}$ and $\boldsymbol{C}_k$ is identity for all $k = 0, \ldots, N-1$. Hence, the network does not distort the state measurements while relaying them. When the IoT node states are fully observed, it is then enough to focus on the memoryless policies [16], which is why we write the optimum policy below as a function of current states rather than the complete system history. Theorem 1 establishes the recursive relationship for the optimum control policy $\boldsymbol{\pi}^\star$ and the min-cost performance under $\boldsymbol{\pi}^\star$. In the remainder of this paper, for any square matrix $\boldsymbol{A} \in \mathbb{R}^{n\times n}$, $\mathrm{Tr}(\boldsymbol{A})$ will represent the classical trace operation given by $\mathrm{Tr}(\boldsymbol{A}) = \sum_{i=1}^n a_{ii}$.

Theorem 1: Assume $p = 1 - q$ and the measured states can be *fully* observed with *perfect* match. Then, the optimum control policy $\boldsymbol{\pi}^\star = (\pi_0^\star, \ldots, \pi_{N-1}^\star)$ and $J^{\star(p,q)}(\boldsymbol{x}_0, \tau_0)$ are given by

$$J^{\star(p,q)}(\boldsymbol{x}_0, \tau_0) = \boldsymbol{x}_0^\top \big(\boldsymbol{L}_0 - \boldsymbol{\Lambda}_0 \mathbf{1}_{\{\tau_0=1\}}\big)\boldsymbol{x}_0 + \sum_{k=0}^{N-1} \mathrm{Tr}(\boldsymbol{K}_{k+1}\boldsymbol{W}_k) \tag{10}$$

and

$$\pi_k^\star(\boldsymbol{x}_k, \tau_k) = -\boldsymbol{V}_k \boldsymbol{x}_k \mathbf{1}_{\{\tau_k=1\}} \tag{11}$$

where $\boldsymbol{V}_k = (\boldsymbol{R}_k + \boldsymbol{B}_k^\top \boldsymbol{K}_{k+1}\boldsymbol{B}_k)^{-1}\boldsymbol{B}_k^\top \boldsymbol{K}_{k+1}\boldsymbol{A}_k$ and the matrices $\boldsymbol{K}_k$, for $k = 1, \ldots, N-1$, are given recursively as

$$\begin{aligned} \boldsymbol{K}_N &= \boldsymbol{Q}_N \\ \boldsymbol{K}_k &= \boldsymbol{L}_k - p\boldsymbol{\Lambda}_k \\ \boldsymbol{L}_k &= \boldsymbol{Q}_k + \boldsymbol{A}_k^\top \boldsymbol{K}_{k+1}\boldsymbol{A}_k \end{aligned}$$

and

$$\boldsymbol{\Lambda}_k = \boldsymbol{A}_k^\top \boldsymbol{K}_{k+1}\boldsymbol{B}_k\boldsymbol{V}_k.$$

Proof: Using the DP algorithm and bearing in mind the relation between feasible $\boldsymbol{u}_k$ and τ_k, we write

$$J_N^{\star(p,q)}(\boldsymbol{x}_N, \tau_N) = \boldsymbol{x}_N^\top \boldsymbol{Q}_N \boldsymbol{x}_N \tag{12}$$

and

$$J_k^{\star(p,q)}(\boldsymbol{x}_k, \tau_k) = \boldsymbol{x}_k^\top \boldsymbol{Q}_k \boldsymbol{x}_k + \min_{\boldsymbol{u}_k \in \mathbb{R}^s} \Big\{ \boldsymbol{u}_k^\top \boldsymbol{R}_k \boldsymbol{u}_k + \mathsf{E}_{(\boldsymbol{x}_k, \tau_k)} \times \Big[J_{k+1}^{\star(p,q)}(\boldsymbol{x}_{k+1}, \tau_{k+1}) \big| \boldsymbol{u}_k \Big] \Big\} \tag{13}$$

for the optimum cost-to-go expressions. We first consider the stage $N-1$. Using the IoT node process evolution in (1) and minimizing the resulting quadratic form for $\tau_{N-1} = 1$, one can obtain

$$\pi_{N-1}^\star(\boldsymbol{x}_{N-1}, 1) = -\boldsymbol{V}_{N-1}\boldsymbol{x}_{N-1}$$

and

$$J_{N-1}^{\star(p,q)}(\boldsymbol{x}_{N-1}, 1) = \boldsymbol{x}_{N-1}^\top (\boldsymbol{L}_{N-1} - \boldsymbol{\Lambda}_{N-1})\boldsymbol{x}_{N-1} + \mathrm{Tr}(\boldsymbol{K}_N \boldsymbol{W}_{N-1}).$$

On the other hand, no control is applied and the matrix $\boldsymbol{\Lambda}_{N-1}$ above disappears when $\tau_{N-1} = 0$, which leads to the desired result for stage $N-1$ in Theorem 1. For stage $N-2$, the same analysis holds, but one also needs to average over the Markov process transitions from $N-2$ to $N-1$. By considering the symmetry assumption, this leads to the optimum control given in (11) and the minimum cost-to-go expression

$$J_{N-2}^{\star(p,q)}(\boldsymbol{x}_{N-2}, \tau_{N-2}) = \boldsymbol{x}_{N-2}^\top \Big(\boldsymbol{L}_{N-2} - \boldsymbol{\Lambda}_{N-2}\mathbf{1}_{\{\tau_{N-2}=1\}}\Big)\boldsymbol{x}_{N-2} + \mathrm{Tr}(\boldsymbol{K}_{N-1}\boldsymbol{W}_{N-2}) + \mathrm{Tr}(\boldsymbol{K}_N \boldsymbol{W}_{N-1}).$$

Iterating similarly, one can complete the proof. ∎

An appealing feature of the optimum control, given by Theorem 1, for implementing a virtual controller at a fog endpoint is its linear structure, which is easy to implement over light-weight fog computing nodes. The effect of the fog endpoint reliability appears as a multiplicative coefficient in the definition of $\boldsymbol{K}_k$ matrices. In particular, as p increases, the additive cost terms $\mathrm{Tr}(\boldsymbol{K}_{k+1}\boldsymbol{W}_k)$, i.e., matrix traces, decrease and we start to have a smaller cost value in (10).[4] We note that this is also the same virtual control service structure implemented over AWS Lambda, Microsoft Azure, Flask HTTP, and Gunicorn WSGI HTTP Unix servers in our experiments above. For the asymmetric case, a similar optimum control recursion can also be obtained after averaging over exponentially many sample ON–OFF scheduling tail-paths of the fog controller at each time $k = 0, \ldots, N-1$, which is, however, not computationally practical for a fog computing system. As a result, we provide performance upper and lower bounds as well as a low-complexity control policy achieving these bounds in the following theorem.

Theorem 2: Assume $p > 1 - q$. Then

$$J^{\star(p,q)}(\boldsymbol{x}_0, \tau_0) \leq J^{\star(1-q,q)}(\boldsymbol{x}_0, \tau_0)$$

and

$$J^{\star(p,q)}(\boldsymbol{x}_0, \tau_0) \geq J^{\star(p,1-p)}(\boldsymbol{x}_0, \tau_0).$$

Moreover, the optimum control achieving $J^{\star(1-q,q)}(\boldsymbol{x}_0, \tau_0)$ with *full* state information and *perfect* match also attains a performance in between these two bounds.

[4] This follows from observing the fact that the matrices $\boldsymbol{K}_k$, $\boldsymbol{L}_k$, and $\boldsymbol{\Lambda}_k$ are positive semi-definite for all $k = 0, \ldots, N-1$.

Proof: Intuitively, decreasing p results in a less reliable fog computing system, which leads to the upper bound in the theorem. Similarly, decreasing q leads to a more reliable fog computing system, which leads to the lower bound in the theorem. For the sake of exposition, the details are relegated to the Appendix. ■

B. Partially Observed State Information

We now consider the case in which measured IoT node states can only be observed partially due to possible distortion in the measurement process and communication environment. In this case, the optimum control policy derived in Theorem 1 loses its memoryless property and depends on the realizations of observed measurement history. However, it is well-known that the information structure considered in this paper does not introduce any dual-effects, and control and estimation problems can be separated [16], [57], which leads to our next theorem.

Theorem 3: Assume $p = 1-q$ and the measured states can be *partially* observed with *perfect* match. Then, the optimum control policy $\boldsymbol{\pi}^\star = (\pi_0^\star, \ldots, \pi_{N-1}^\star)$ and $J^{\star(p,q)}(\boldsymbol{x}_0, \tau_0)$ are given by

$$J^{\star(p,q)}(\boldsymbol{x}_0, \tau_0) = \boldsymbol{x}_0^\top \big(\boldsymbol{L}_0 - \boldsymbol{\Lambda}_0 \mathbf{1}_{\{\tau_0=1\}}\big)\boldsymbol{x}_0 + \sum_{k=0}^{N-1} \mathrm{Tr}(\boldsymbol{K}_{k+1}\boldsymbol{W}_k) + \mathsf{E}_{(\boldsymbol{x}_0,\tau_0)}\Big[\boldsymbol{\epsilon}_0^\top \boldsymbol{\Lambda}_0 \mathbf{1}_{\{\tau_0=1\}}\boldsymbol{\epsilon}_0\Big] + p\sum_{k=1}^{N-1} \mathsf{E}_{(\boldsymbol{x}_0,\tau_0)}\Big[\boldsymbol{\epsilon}_k^\top \boldsymbol{\Lambda}_k \boldsymbol{\epsilon}_k\Big] \tag{14}$$

and

$$\pi_k^\star(H_k, \tau_k) = -\boldsymbol{V}_k \mathsf{E}\big[\boldsymbol{x}_k \big| H_k\big] \mathbf{1}_{\{\tau_k=1\}} \tag{15}$$

where $\boldsymbol{\epsilon}_k = \boldsymbol{x}_k - \mathsf{E}[\boldsymbol{x}_k|H_k]$ and the matrices $\boldsymbol{V}_k$, $\boldsymbol{K}_k$, $\boldsymbol{\Lambda}_k$, and $\boldsymbol{L}_k$ are as defined in Theorem 1.

Proof: The proof follows from the existence of no dual-effects and the property of conditional expectations $\mathsf{E}[\mathsf{E}[X|\mathcal{F}_2]\big|\mathcal{F}_1] = \mathsf{E}[X|\mathcal{F}_1]$ for any two nested σ-algebras $\mathcal{F}_1 \subseteq \mathcal{F}_2$ from [58]. ■

The first two terms in (14) give the minimum cost attained in the former case of fully observed state information. On the other hand, the last two terms describe the effect of state estimation on the performance of virtual controller over the fog. Further, the structure of optimum control in (15) is similar to the one in Theorem 1, except the nonlinear estimator $\mathsf{E}[\boldsymbol{x}_k|H_k]$ that needs to be run at the fog endpoint, separately from the controller application. This can be an onerous and time-consuming task for light-weight fog nodes. Hence, it can be replaced with its *linear* approximation through Kalman filtering in practical settings with some loss of optimality when the measurement-plus-channel noise is not Gaussian.

The main procedures both at the IoT node and fog controller sides to implement a fog controller system with perfect match are illustrated in Algorithms 1 and 2. For the sake of exposition, random disturbance/noise generation and their effects on the state updates and measured data are illustrated explicitly but in a somewhat synthetic way. In an actual system implementation, they are expected to arise intrinsically due to underlying system dynamics and environmental conditions. We only represent the distortion due to measurement noise in

Algorithm 1 Fog Controller System Implementation With Perfect Match—IoT Node Side

IoT Node Side: Receives control signals, updates current states and sends state measurements to the fog controller.

Initialization: Store $\boldsymbol{A}_k, \boldsymbol{B}_k, \boldsymbol{C}_k$ for $k = 0, \ldots, N-1$.

procedure RECEIVECONTROL(sockID) ▷ Listens to receive control signals over sockID
procedure RANDOM(F) ▷ Generates random disturbance/noise with distribution F
procedure STATEUPDATE($k, \boldsymbol{x}_k, F_k$, sockID) ▷ Updates the IoT node states
  $\boldsymbol{u}_k = \mathbf{0}$
  while TimeOut == **false do** ▷ Waits one time-unit to receive a control signal
    $\boldsymbol{u}_k \leftarrow$ RECEIVECONTROL(sockID)
  $\boldsymbol{w}_k =$ RANDOM (F_k)
  $\boldsymbol{x}_{k+1} \leftarrow \boldsymbol{A}_k\boldsymbol{x}_k + \boldsymbol{B}_k\boldsymbol{u}_k + \boldsymbol{w}_k$
  return $\boldsymbol{x}_{k+1}$
procedure MEASUREANDSEND($k, \boldsymbol{x}_k, F_k$)
  $\boldsymbol{v}_k =$ RANDOM(F_k)
  $\boldsymbol{z}_k = \boldsymbol{C}_k\boldsymbol{x}_k + \boldsymbol{v}_k$
  SEND2FOG($\boldsymbol{z}_k$)

Algorithm 2 Fog Controller System Implementation With Perfect Match—Fog Controller Side

Fog Controller Side: Receives measurements, generates control signals and sends them to the IoT node.

Initialization: Store $\boldsymbol{A}_k, \boldsymbol{B}_k, \boldsymbol{C}_k, \boldsymbol{Q}_k, \boldsymbol{R}_k$ for $k = 0, \ldots, N-1$.

procedure RECEIVEMEASUREMENT(sockID) ▷ Listens to receive measurements over sockID
procedure STATEESTIMATE($\boldsymbol{z}$, hist) ▷ Estimates the IoT node states based on the currently received measurement $\boldsymbol{z}$ and history hist through Kalman filtering
procedure CONTROLLERMATRICES(p) ▷ Generates controller matrices
  $\boldsymbol{K}_N = \boldsymbol{Q}_N$
  for $k = N-1$:0 **do**
    $\boldsymbol{L}_k = \boldsymbol{Q}_k + \boldsymbol{A}_k^\top \boldsymbol{K}_{k+1}\boldsymbol{A}_k$
    $\boldsymbol{V}_k = \big(\boldsymbol{R}_k + \boldsymbol{B}_k^\top \boldsymbol{K}_{k+1}\boldsymbol{B}_k\big)^{-1}\boldsymbol{B}_k^\top \boldsymbol{K}_{k+1}\boldsymbol{A}_k$
    $\boldsymbol{\Lambda}_k = \boldsymbol{A}_k^\top \boldsymbol{K}_{k+1}\boldsymbol{B}_k\boldsymbol{V}_k$
    $\boldsymbol{K}_k = \boldsymbol{L}_k - p\boldsymbol{\Lambda}_k$
  return $\boldsymbol{V}_k$ for $k = 0, \ldots, N-1$
procedure CONTROLANDSEND(k, τ_k)
  $\boldsymbol{u}_k = 0$
  if $\tau_k == 1$ **then**
    $\boldsymbol{z}_k =$ RECEIVEMEASUREMENT(sockID)
    $\hat{\boldsymbol{x}}_k =$ STATEESTIMATE($\boldsymbol{z}_k$, hist)
    $\boldsymbol{u}_k \leftarrow -\boldsymbol{V}_k\hat{\boldsymbol{x}}_k$
  SEND2IoT($\boldsymbol{u}_k$)

these algorithms. Finally, we note that the measurement matrix $\boldsymbol{C}_k$ and the distribution of the noise $\boldsymbol{v}_k$ are used by the state estimation procedure in Algorithm 2 although we do not show this explicitly.

An important remark about the steady-state behavior of the optimum control policy, which will facilitate the implementation in Algorithm 2, is the following. As formulated in its most general form described by (1) and (6), we have a time-varying IoT control system. Hence, all the states are transient and we cannot talk about the steady-state behavior. In this case, all the system matrices $\boldsymbol{A}_k, \boldsymbol{B}_k, \boldsymbol{Q}_k$, and $\boldsymbol{R}_k$ for $k = 0, \ldots, N-1$ must be stored and the discrete-time Riccati equation must be solved iteratively, starting from $k = N$ and going backwards in time, to obtain controller gain matrices $\boldsymbol{V}_k$. On the other hand, if the system is time-invariant, i.e., $\boldsymbol{A}_k = \boldsymbol{A}, \boldsymbol{B}_k = \boldsymbol{B}, \boldsymbol{Q}_k = \boldsymbol{Q}$, and $\boldsymbol{R}_k = \boldsymbol{R}$ for all $k = 0, \ldots, N-1$, then $\boldsymbol{K}_k$ converges to a steady-state matrix $\boldsymbol{K}$ given by the solution of the algebraic Riccati equation as $k \to -\infty$ [16], [55]. This implies the convergence of gain matrices $\boldsymbol{V}_k$ to $\boldsymbol{V} = (\boldsymbol{R} + \boldsymbol{B}^\top \boldsymbol{K}\boldsymbol{B})^{-1}\boldsymbol{B}^\top \boldsymbol{K}\boldsymbol{A}$ as we go backwards in time starting from large N values. Here, due to the DP recursion, the steady-state behavior occurs for small values of k, whereas the transient-state behavior occurs as k approaches N. Ignoring the transient behavior, the optimum control policy can be approximated by the stationary policy $\boldsymbol{\pi}_{\rm ss}$ using $\boldsymbol{V}$ as its gain matrix for all time, i.e., $\pi_{k,{\rm ss}}(\boldsymbol{x}) = -\boldsymbol{V}\boldsymbol{x}$ for all $k = 0, \ldots, N-1$. Implementation of $\boldsymbol{\pi}_{\rm ss}$ only requires the knowledge of $\boldsymbol{V}$. This approximate implementation saves considerable memory space, especially when N is large and the system has high dimensionality.

The next theorem provides the analogous upper and lower bounds on the fog controller performance with partial state information.

Theorem 4: Assume $p > 1 - q$. Then

$$J^{\star(p,q)}(\boldsymbol{x}_0, \tau_0) \leq J^{\star(1-q,q)}(\boldsymbol{x}_0, \tau_0)$$

and

$$J^{\star(p,q)}(\boldsymbol{x}_0, \tau_0) \geq J^{\star(p,1-p)}(\boldsymbol{x}_0, \tau_0).$$

Moreover, the optimum control achieving $J^{\star(1-q,q)}(\boldsymbol{x}_0, \tau_0)$ with *partial* state information and *perfect* match also attains a performance in between these two bounds.

Proof: The proof follows from similar lines in the Appendix by considering the same starting states and observed history for both systems in the inductive arguments, and hence is omitted to avoid repetition. ∎

Remark: It should be noted that we use the same notation $J^{\star(p,q)}(\boldsymbol{x}_0, \tau_0)$ for the min-cost value achieved by the optimum control policy in both cases of fully and partially observed state information in order to avoid excessive proliferation of different symbols throughout this paper. It is clear from the context to which min-cost value we are referring to. For example, the min-cost values in Theorem 4 are those obtained in the case of partially observed state information, whereas the ones in Theorem 2 are those obtained in the case of fully observed state information. Hence, the corresponding bounds in these two different cases are different from each other.

VI. Control Over Fog With Imperfect Match

Despite being insightful when communication and computation latency can be ignored with respect to the IoT node state evolution dynamics, our analysis in Section V cannot capture the wide variation of delay values measured in our virtual fog controller experiments, i.e., see Table I. Therefore, in this section, we will extend the basic model above to discover the effects of latency on the utility of fog controller placement along the cloud-to-things continuum. The augmented model is aimed to characterize the swiftness of control and timeliness of state measurements due to network delay between the IoT node and the fog endpoint.

Specifically, two types of delay are considered: 1) forward delay M_F and 2) backward delay M_B. M_F is the delay incurred on the path from the IoT node to the fog endpoint at which the controller software runs, whereas M_B is the delay in the reverse direction. To make the exposition simpler, we assume that any delay to compute the optimum control signal is included in M_B in the sequel. The total delay incurred in both ways is then equal to $M = M_F + M_B$.[5] In this setup, any measurement about the IoT node state sent out at time k arrives to the fog controller at time $k + M_F$, and a possible corrective control signal arrives back to the IoT node at time $k + M$. A potential manifestation of this latency is a proportional decrease in the frequency of control actions arriving to the IoT node, which we model below by assuming delay-spreaded requests-for-control as in event-driven interactive control systems. We start our analysis with the case of fully observed state information.

A. Fully Observed State Information

This is the case in which transmitted state measurements can be fully observed by the fog controller with delay M_F. The next theorem establishes the optimum control rule and the min-cost performance under optimum control with fully observed but delayed state information.

Theorem 5: Assume $p = 1 - q$, $N \geq M \geq 1$ and the measured states can be *fully* observed with *imperfect* match. Define $c \triangleq ([N-a]/M)$, where $a = N \mod M$ if N is not an integer multiple of M, and $a = M$ otherwise. Then, the optimum control policy $\boldsymbol{\pi}^\star = (\pi_0^\star, \ldots, \pi_{N-1}^\star)$ and $J^{\star(p,q)}(\boldsymbol{x}_0, \tau_0)$ are given by

$$J^{\star(p,q)}(\boldsymbol{x}_0, \tau_0) = \boldsymbol{x}_0^\top \boldsymbol{L}_0 \boldsymbol{x}_0 + \sum_{k=0}^{N-1} \mathrm{Tr}(\boldsymbol{K}_{k+1}\boldsymbol{W}_k) + p \sum_{k=0}^{cM-1} \mathrm{Tr}(\boldsymbol{P}_{k+1}\boldsymbol{W}_k) \tag{16}$$

and

$$\pi_k^\star(\boldsymbol{\lambda}_k, \tau_{k-M_B}) = \begin{cases} \boldsymbol{0}, & \text{if } k \neq 0 \mod M \\ -\boldsymbol{V}_k \mathsf{E}[\boldsymbol{x}_k | \boldsymbol{\lambda}_k] \mathbf{1}_{\{\tau_{k-M_B}=1\}}, & \text{if } k = 0 \mod M \end{cases} \tag{17}$$

for $k \in \{1, \ldots, N-1\}$, where $\boldsymbol{\lambda}_k = (\boldsymbol{x}_{k-M}, \boldsymbol{u}_{k-M})$ (for $k \geq M$), $\boldsymbol{V}_k = (\boldsymbol{R}_k + \boldsymbol{B}_k^\top \boldsymbol{K}_{k+1}\boldsymbol{B}_k)^{-1}\boldsymbol{B}_k^\top \boldsymbol{K}_{k+1}\boldsymbol{A}_k$ and the matrices $\boldsymbol{K}_k$ and

[5]Here, we assume that these delays can either be reliably estimated or do not vary a lot around their means so that the discrete model for them can be reasonably approximated as being deterministic.

$\boldsymbol{P}_k$ are given recursively as

$$\begin{aligned}\boldsymbol{K}_N &= \boldsymbol{Q}_N \\ \boldsymbol{K}_k &= \begin{cases} \boldsymbol{L}_k, & \text{if } k \neq 0 \mod M \\ \boldsymbol{L}_k - p\boldsymbol{\Lambda}_k, & \text{if } k = 0 \mod M \end{cases} \\ \boldsymbol{L}_k &= \boldsymbol{Q}_k + \boldsymbol{A}_k^\top \boldsymbol{K}_{k+1}\boldsymbol{A}_k\end{aligned}$$

and

$$\boldsymbol{\Lambda}_k = \boldsymbol{A}_k^\top \boldsymbol{K}_{k+1}\boldsymbol{B}_k\boldsymbol{V}_k$$

for $k \in \{0, \ldots, N-1\}$, and

$$\begin{aligned}\boldsymbol{P}_{cM} &= \boldsymbol{\Lambda}_{cM} \\ \boldsymbol{P}_k &= \begin{cases} \boldsymbol{A}_k^\top \boldsymbol{P}_{k+1}\boldsymbol{A}_k, & \text{if } k \neq 0 \mod M \\ \boldsymbol{\Lambda}_k, & \text{if } k = 0 \mod M \end{cases}\end{aligned}$$

for $k \in \{0, \ldots, cM-1\}$.

Proof: We will provide the proof only for when N is not an integer multiple of M. The other case follows from observing that the final time a useful control arrives is $N-M$ and repeating the same steps below.

First, observe that no control signal arrives from time $cM+1$ to N. Hence, we can write

$$J_i^{\star(p,q)}(\boldsymbol{x}_i, \tau_i) = \boldsymbol{x}_i^\top \boldsymbol{L}_i \boldsymbol{x}_i + \sum_{k=i}^{N-1} \mathrm{Tr}(\boldsymbol{K}_{k+1}\boldsymbol{W}_k) \tag{18}$$

for $i \in \{cM+1, \ldots, N\}$. Second, we observe that a control signal arrives to the IoT node for the final time at cM, generated by the fog controller at $cM - M_B$ based on the full state observation $\boldsymbol{x}_{(c-1)M}$ and the possible control signal $\boldsymbol{u}_{(c-1)M}$ at time $(c-1)M$. To this end, the fog controller needs to solve

$$\min_{\boldsymbol{u}\in\mathbb{R}^s} \left\{ \boldsymbol{u}^\top \boldsymbol{R}_i \boldsymbol{u} + \mathsf{E}\left[\mathsf{E}_{\boldsymbol{x}_i}\left[J_{i+1}^{\star(p,q)}(\boldsymbol{x}_{i+1}, \tau_{i+1}) \middle| \boldsymbol{u} \right] \middle| \boldsymbol{\lambda}_i \right] \right\}$$

for $i = cM$. Using (18) and system linearity, the above minimization leads to

$$\pi_{cM}^\star(\boldsymbol{\lambda}_{cM}, \tau_{cM-M_B}) = -\boldsymbol{V}_{cM} \mathsf{E}\left[\boldsymbol{x}_{cM} \middle| \boldsymbol{\lambda}_{cM}\right] \mathbf{1}_{\{\tau_{cM-M_B}=1\}}$$

and, with a slight abuse of notation,[6]

$$\begin{aligned} J_{cM}^{\star(p,q)}(\boldsymbol{x}_{cM}, \tau_{cM}) = \boldsymbol{x}_{cM}^\top &\left(\boldsymbol{L}_{cM} - \boldsymbol{\Lambda}_{cM}\mathbf{1}_{\{\tau_{cM-M_B}=1\}}\right)\boldsymbol{x}_{cM} \\ &+ \sum_{k=cM}^{N-1} \mathrm{Tr}(\boldsymbol{K}_{k+1}\boldsymbol{W}_k) \\ &+ \boldsymbol{\epsilon}_{cM}^\top \boldsymbol{P}_{cM} \mathbf{1}_{\{\tau_{cM-M_B}=1\}} \boldsymbol{\epsilon}_{cM} \end{aligned}$$

where $\boldsymbol{\epsilon}_{cM} = \boldsymbol{x}_{cM} - \mathsf{E}[\boldsymbol{x}_{cM}|\boldsymbol{\lambda}_{cM}]$. Now observing that no control signal arrives at time $cM-1$ and using the symmetry assumption as well as system linearity, we have

$$\begin{aligned} J_i^{\star(p,q)}(\boldsymbol{x}_i, \tau_i) = \boldsymbol{x}_i^\top \boldsymbol{L}_i \boldsymbol{x}_i &+ \sum_{k=i}^{N-1} \mathrm{Tr}(\boldsymbol{K}_{k+1}\boldsymbol{W}_k) \\ &+ p\mathrm{Tr}(\boldsymbol{P}_{i+1}\boldsymbol{W}_i) + p\boldsymbol{\epsilon}_i^\top \boldsymbol{P}_i \boldsymbol{\epsilon}_i \end{aligned}$$

for $i = cM-1$ and $\boldsymbol{\epsilon}_{cM-1} = \boldsymbol{x}_{cM-1} - \mathsf{E}[\boldsymbol{x}_{cM-1}|\boldsymbol{\lambda}_{cM}]$, which is the *residual* error from estimation at time cM.

[6] $J_{cM}^{\star(p,q)}(\boldsymbol{x}_{cM}, \tau_{cM})$ actually depends on τ_{cM-M_B} but we have chosen to use above notation for the sake notational consistency.

In order to complete the proof, we repeat the same steps until $(c-1)M$. Observing that the residual error term from estimation at cM is equal to $\boldsymbol{w}_{(c-1)M}$ at $(c-1)M$ and taking the form of a new estimation error appearing due to a potential control signal to be applied at $(c-1)M$ into account, we arrive at

$$\begin{aligned} J_i^{\star(p,q)}(\boldsymbol{x}_i, \tau_i) = \boldsymbol{x}_i^\top &\left(\boldsymbol{L}_i - \boldsymbol{\Lambda}_i \mathbf{1}_{\{\tau_{i-M_B}=1\}}\right)\boldsymbol{x}_i + \sum_{k=i}^{N-1} \mathrm{Tr}(\boldsymbol{K}_{k+1}\boldsymbol{W}_k) \\ &+ p \sum_{k=i}^{cM-1} \mathrm{Tr}(\boldsymbol{P}_{k+1}\boldsymbol{W}_k) + \boldsymbol{\epsilon}_i^\top \boldsymbol{P}_i \mathbf{1}_{\{\tau_{i-M_B}=1\}} \boldsymbol{\epsilon}_i \end{aligned}$$

where $\boldsymbol{\epsilon}_i = \boldsymbol{x}_i - \mathsf{E}[\boldsymbol{x}_i|\boldsymbol{\lambda}_i]$ for $i = (c-1)M$. Observing the emerging structure, and iterating similarly first from $(c-1)M$ to M and then from M to 0, one can complete the proof. ■

There are several important structural features of the optimum control policy and the resulting minimum cost appearing in Theorem 5. First, the linearity property is preserved even with imperfect match between the IoT node process and the fog controller placement. This is important for practical low-complexity implementations of the controller application at the fog endpoint. Second, the separation principle still holds in the case of imperfect match. An important ramification of this observation is that an estimator needs to be run separately from the controller software at the fog endpoint, which can be interpreted as a delay compensator. However, when compared to the estimation problem with partial state observation in (15), this estimator is linear and easy to implement due to linear evolution of IoT node states in (1). Third, the frequency of corrective control signals arriving to the IoT node is decreased in proportion to M. This is due to the aggregate delay over the fog network that spreads transmitted measurements and control signals. Finally, the last summation in (16) represents the collateral impact of latency on the min-cost performance due to delayed state measurements, which would disappear for an hypothetical perfect estimator.[7]

1) Virtual Controller Placement: Equations (10) and (16) in Theorems 1 and 5 provide a pair of quantative performance values for fog network designers to decide about where to place virtual controller software by considering inherent service grade and latency characteristics of available fog endpoints when measurement and channel distortion can be ignored. The next theorem provides the analogous upper and lower bounds on the fog controller performance, as is done in Section V.

Theorem 6: Assume $p > 1-q$. Then

$$J^{\star(p,q)}(\boldsymbol{x}_0, \tau_0) \leq J^{\star(1-q,q)}(\boldsymbol{x}_0, \tau_0)$$

and

$$J^{\star(p,q)}(\boldsymbol{x}_0, \tau_0) \geq J^{\star(p,1-p)}(\boldsymbol{x}_0, \tau_0).$$

Moreover, the optimum control achieving $J^{\star(1-q,q)}(\boldsymbol{x}_0, \tau_0)$ with *full* state information and *imperfect* match also attains a performance in between these two bounds.

[7] $J^{\star(p,q)}(\boldsymbol{x}_0, \tau_0)$ in (16) does not depend on τ_0 since the system forgets about the starting fog node state after one unit delay due to symmetry assumption.

Proof: The proof follows from similar lines in the Appendix by observing that the inductive hypothesis holds for $k = cM, \ldots, N$, and considering the same starting states and observed measurements in the inductive arguments for any $k < cM$. It is omitted to avoid repetition. ∎

Remark: Similar to our remark after Theorem 4, we note that the bounds in Theorem 6 refer to the min-cost values obtained by the optimum control policy for the imperfectly matched fog controller with fully observed state information. Hence, they are different than those appearing in Theorems 2 and 4.

B. Partially Observed State Information

Finally, we analyze the case with partially observed states and imperfect match between the IoT node and fog controller. Updating the definition of system history in (5) in an obvious way to include only delayed measurements and repeating the same steps above, we obtain the following theorems.

Theorem 7: Assume $p = 1 - q$, $N \geq M \geq 1$ and the measured states can be *partially* observed with *imperfect* match. Define $c \triangleq ([N-a]/M)$, where $a = N \mod M$ if N is not an integer multiple of M, and $a = M$ otherwise. Then, the optimum control policy $\boldsymbol{\pi}^\star = (\pi_0^\star, \ldots, \pi_{N-1}^\star)$ and $J^{\star(p,q)}(\boldsymbol{x}_0, \tau_0)$ are given by

$$J^{\star(p,q)}(\boldsymbol{x}_0, \tau_0) = \boldsymbol{x}_0^\top \boldsymbol{L}_0 \boldsymbol{x}_0 + \sum_{k=0}^{N-1} \mathrm{Tr}(\boldsymbol{K}_{k+1}\boldsymbol{W}_k) + p \sum_{k=0}^{cM-1} \mathrm{Tr}(\boldsymbol{P}_{k+1}\boldsymbol{W}_k) + p \sum_{k=1}^{c-1} \mathsf{E}_{(\boldsymbol{x}_0,\tau_0)}\left[\boldsymbol{\epsilon}_{kM}^\top \boldsymbol{P}_{kM} \boldsymbol{\epsilon}_{kM}\right] \quad (19)$$

and

$$\pi_k^\star\big(H_k, \tau_{k-M_B}\big) = \begin{cases} \boldsymbol{0}, & \text{if } k \neq 0 \mod M \\ -\boldsymbol{V}_k \mathsf{E}\big[\boldsymbol{x}_k \big| H_k\big] \mathsf{1}_{\{\tau_{k-M_B}=1\}}, & \text{if } k = 0 \mod M \end{cases} \quad (20)$$

for $k \in \{1, \ldots, N-1\}$, where the matrices $\boldsymbol{V}_k$, $\boldsymbol{K}_k$, $\boldsymbol{\Lambda}_k$, $\boldsymbol{L}_k$, and $\boldsymbol{P}_k$ are as defined in Theorem 5, and $\boldsymbol{\epsilon}_{kM} = \boldsymbol{x}_{kM} - \mathsf{E}[\boldsymbol{x}_{kM}\big|H_{(k+1)M}]$ for $k \in \{1, \ldots, c-1\}$.

Proof: The proof follows from the same steps in the proof of Theorem 5, existence of no dual-effects and the property of conditional expectations $\mathsf{E}[\mathsf{E}[X\big|\mathcal{F}_2]\big|\mathcal{F}_1] = \mathsf{E}[X\big|\mathcal{F}_1]$ for any two nested σ-algebras $\mathcal{F}_1 \subseteq \mathcal{F}_2$ from [58]. ∎

The last summation in (19) represents the error terms arising from the uncertainty that cannot be resolved via partial state observations. As in the case of perfect match with partial state observations, the estimation in (20) is nonlinear, which can be replaced with its linear version through practical Kalman filtering implementation at the expense of some loss of optimality when $\boldsymbol{v}_k$ is not Gaussian. The main procedures to implement a fog controller system with imperfect match are illustrated in Algorithm 3, which only contains the ones for the fog controller side. The procedures at the IoT node side are similar to those in Algorithm 1, except receiving control signals less frequently due to delay, and hence omitted for brevity.

Algorithm 3 Fog Controller System Implementation With Imperfect Match—Fog Controller Side

Fog Controller Side: Receives measurements, generates control signals and sends them to the IoT node.

Initialization: Store $\boldsymbol{A}_k, \boldsymbol{B}_k, \boldsymbol{C}_k, \boldsymbol{Q}_k, \boldsymbol{R}_k$ for $k = 0, \ldots, N-1$.

procedure RECEIVEMEASUREMENT(sockID) ▷ Listens to receive measurements over sockID
procedure ESTIMATEFORWARDDELAY ▷ Estimates the forward delay M_F from the IoT node to the fog controller
procedure ESTIMATEBACKWARDDELAY ▷ Estimates the backward delay M_B from the fog controller to the IoT node
procedure STATEESTIMATE(z, M_B, hist) ▷ Estimates the IoT node states based on the currently received measurement z, backward delay M_B and history hist through Kalman filtering
procedure CONTROLLERMATRICES(p) ▷ Generates controller matrices
    M = EstimateForwardDelay + EstimateBackwardDelay
    $\boldsymbol{K}_N = \boldsymbol{Q}_N$
    for $k = N-1$:0 **do**
        $\boldsymbol{L}_k = \boldsymbol{Q}_k + \boldsymbol{A}_k^\top \boldsymbol{K}_{k+1} \boldsymbol{A}_k$
        $\boldsymbol{V}_k = \big(\boldsymbol{R}_k + \boldsymbol{B}_k^\top \boldsymbol{K}_{k+1} \boldsymbol{B}_k\big)^{-1} \boldsymbol{B}_k^\top \boldsymbol{K}_{k+1} \boldsymbol{A}_k$
        $\boldsymbol{\Lambda}_k = \boldsymbol{A}_k^\top \boldsymbol{K}_{k+1} \boldsymbol{B}_k \boldsymbol{V}_k$
        if k % M == 0 **then**
            $\boldsymbol{K}_k = \boldsymbol{L}_k - p\boldsymbol{\Lambda}_k$
        else
            $\boldsymbol{K}_k = \boldsymbol{L}_k$
    return $\boldsymbol{V}_k$ for $k = 0, \ldots, N-1$
procedure CONTROLANDSEND(k, τ_{k-M_B})
    $\boldsymbol{u}_k = 0$
    M_B = EstimateBackwardDelay
    $M = M_B$+ EstimateForwardDelay
    if τ_{k-M_B} == 1 & k % M == 0 **then**
        $\boldsymbol{z}_{k-M_B}$ = RECEIVEMEASUREMENT(sockID)
        $\hat{\boldsymbol{x}}_k$ = STATEESTIMATE$\big(\boldsymbol{z}_{k-M_B}, M_B$, hist$\big)$
        $\boldsymbol{u}_k \leftarrow -\boldsymbol{V}_k \hat{\boldsymbol{x}}_k$
    SEND2IoT($\boldsymbol{u}_k$)

1) Virtual Controller Placement: Equations (14) and (19) in Theorems 3 and 7 provide a pair of quantative performance values for fog network designers to decide about where to place virtual controller software by considering inherent service grade and latency characteristics of available fog endpoints when measurement and channel distortion cannot be ignored.

Theorem 8: Assume $p > 1 - q$. Then

$$J^{\star(p,q)}(\boldsymbol{x}_0, \tau_0) \leq J^{\star(1-q,q)}(\boldsymbol{x}_0, \tau_0)$$

and

$$J^{\star(p,q)}(\boldsymbol{x}_0, \tau_0) \geq J^{\star(p,1-p)}(\boldsymbol{x}_0, \tau_0).$$

Moreover, the optimum control achieving $J^{\star(1-q,q)}(\boldsymbol{x}_0, \tau_0)$ with *partial* state information and *imperfect* match also attains a performance in between these two bounds.

Proof: The proof follows from similar lines in the Appendix by observing that the inductive hypothesis holds for $k = cM, \ldots, N$, and considering the same starting states and observed system history in the inductive arguments for any $k < cM$. It is omitted to avoid repetition. ∎

Remark: Similar to our remarks after Theorems 4 and 6, we note that the bounds in Theorem 8 are different than those appearing in Theorems 2, 4, and 6.

VII. UAV Trajectory Control Over Fog

In this section, we illustrate the utility of our analytical results derived above in the context of trajectory tracking problems for UAVs. While this paper applies to all linear systems with quadratic cost, we note that our model maps well to practical drone trajectory tracking control problems. For example, modern high-end DJI quadcopters [59] can be programmed with a sequence of way-points, and can receive velocity adjustment signals. Our virtual controller model can also be used to dictate different levels of trajectory tracking precision in different parts of the trajectory, which is important for practical drone settings where more precision is required close to stationary obstacles, other drones, and no-fly zones than in unrestricted air spaces.

A. State-Space Representation for UAV Control

We consider a planar motion at some certain altitude determined through a sequence of way-points $\{\bar{\boldsymbol{x}}_k\}_{k=0}^{N} \subset \mathbb{R}^2$. $\bar{\boldsymbol{x}}_k$ represents the desired position of the UAV at time $k\Delta t$. Here, Δt is our basic discrete-time unit to communicate location/velocity information (obtained through GPS sensors) and control signals between the fog server and UAV.[8]

The task of the fog controller is to provide velocity adjustment signals represented by $\boldsymbol{u}_k$ (measured in meters per second) for $k = 0, \ldots, N-1$ to determine the velocity of the UAV from time $k\Delta t$ to $(k+1)\Delta t$. Succinctly, the state update equation can be given as

$$\begin{pmatrix} \boldsymbol{x}_{k+1} \\ \boldsymbol{v}_{k+1} \end{pmatrix} = \begin{pmatrix} \boldsymbol{x}_k + \Delta t(\boldsymbol{v}_k + \boldsymbol{u}_k) \\ \boldsymbol{v}_k + \boldsymbol{u}_k \end{pmatrix} + \begin{pmatrix} \boldsymbol{w}_k^x \\ \boldsymbol{w}_k^v \end{pmatrix} \tag{21}$$

where $\boldsymbol{w}_k^x$ and $\boldsymbol{w}_k^v$ are random (possibly correlated) disturbances affecting the location and velocity of the UAV due to environmental conditions, such as wind and rain, respectively. After some manipulations, (21) can be written as

$$\begin{pmatrix} \boldsymbol{e}_{k+1} \\ \boldsymbol{v}_{k+1} \end{pmatrix} = \boldsymbol{A}_k \begin{pmatrix} \boldsymbol{e}_k \\ \boldsymbol{v}_k \end{pmatrix} + \boldsymbol{B}_k \boldsymbol{u}_k + \begin{pmatrix} \boldsymbol{w}_k^x + \bar{\boldsymbol{x}}_k - \bar{\boldsymbol{x}}_{k+1} \\ \boldsymbol{w}_k^v \end{pmatrix}$$

where $\boldsymbol{e}_k = \boldsymbol{x}_k - \bar{\boldsymbol{x}}_k$, $\boldsymbol{A}_k = \begin{pmatrix} \boldsymbol{I}_2 & \Delta t \boldsymbol{I}_2 \\ \boldsymbol{0} & \boldsymbol{I}_2 \end{pmatrix}$ and $\boldsymbol{B}_k = \begin{pmatrix} \Delta t \boldsymbol{I}_2 \\ \boldsymbol{I}_2 \end{pmatrix}$ for $k = 0, \ldots, N-1$.[9] This is similar to the linear IoT node model in Section IV, with drift terms $\bar{\boldsymbol{x}}_k - \bar{\boldsymbol{x}}_{k+1}$ providing desired trajectory information. Hence, the optimum control trying to minimize drifted state measurements will direct the UAV along the desired path. To this end, we minimize the following cost $J = \sum_{k=0}^{N} \mathsf{E}[|\boldsymbol{e}_k|^2 + \alpha(|\boldsymbol{v}_k|^2 + |\boldsymbol{u}_k|^2)]$ in an attempt to balance trajectory deviations and energy expenditure, which leads to $\boldsymbol{Q}_k = \begin{pmatrix} \boldsymbol{I}_2 & \boldsymbol{0} \\ \boldsymbol{0} & \alpha \boldsymbol{I}_2 \end{pmatrix}$, $\boldsymbol{R}_k = \alpha \boldsymbol{I}_2$ and $\alpha \geq 0$ being a design parameter. In this cost expression, we note that the terms containing $|\boldsymbol{e}_k|^2$ measure the total deviation from the desired trajectory, whereas the remaining terms act as a proxy for the total energy spent to follow it.

B. Trajectory Tracking Performance

We consider a target located at a given position in space. The objective of the UAV is to approach the target (represented by a red cross in the figures below), make a circle around it and then turn back to its starting position. The disturbances in location and velocity are jointly Gaussian with mean zero and the covariance matrix $\boldsymbol{\Sigma} = \begin{pmatrix} \sigma_x^2 \boldsymbol{I}_2 & \rho \boldsymbol{I}_2 \\ \rho \boldsymbol{I}_2 & \sigma_v^2 \boldsymbol{I}_2 \end{pmatrix}$.

In Fig. 4, we show the realizations of the vehicle trajectory for different values of the fog controller reliability parameter p. In this figure, we set the system parameters α, σ_x, and σ_y to 0.1 in order to delineate the trajectory tracking performance of the fog controller with respect to its reliability and delay. We chose ρ to have a correlation coefficient of 0.5 in all cases. In the upper figures, we observe that our optimum fog controller without delay is able to stabilize the UAV around the predefined desired trajectory. As expected, the performance of the fog controller in tracking the desired trajectory improves when p increases. In particular, the UAV almost perfectly tracks the desired trajectory for values of p above 0.5, which can used, for example, as the minimum level of virtual controller reliability for control services to be provisioned over fog in this particular case.

In the lower figures in Fig. 4, on the other hand, we illustrate the performance of the fog controller with delay $M = 3\Delta t$. Even with a very reliable fog controller, we observe substantial negative effects of delay. In particular, the fog controller loses its ability to guide the UAV around the desired trajectory starting from $p = 0.5$ and downward even with small disturbance. These observations perfectly illustrate the potential of our analytical expressions to guide the design and engineering efforts to offer control-as-a-service over fog by considering reliability and delay constraints.

Second, we investigate the effect of environmental disturbances on the trajectory tracking performance of a UAV fog controller in Fig. 5. We set p to 0.75 and α to 0.1, and chose ρ to have a correlation coefficient of 0.5 in all cases.

[8]The model can be extended to include time-varying discrete-time units. However, we do not pursue this direction in this paper for the sake of notational simplicity.

[9]More explicitly, the randomly varying velocity process of the UAV can be written as $\boldsymbol{v}(t) = \boldsymbol{v}_k + \boldsymbol{u}_k + \boldsymbol{w}(t)$ for $k\Delta t < t \leq (k+1)\Delta t$, where $\boldsymbol{w}(t)$ is the stochastic disturbance process inflicting the UAV motion. Then, $\boldsymbol{w}_k^x$ is the integral of $\boldsymbol{w}(t)$ from $k\Delta t$ to $(k+1)\Delta t$, and $\boldsymbol{w}_k^v$ is the value of $\boldsymbol{w}(t)$ at time $(k+1)\Delta t$.

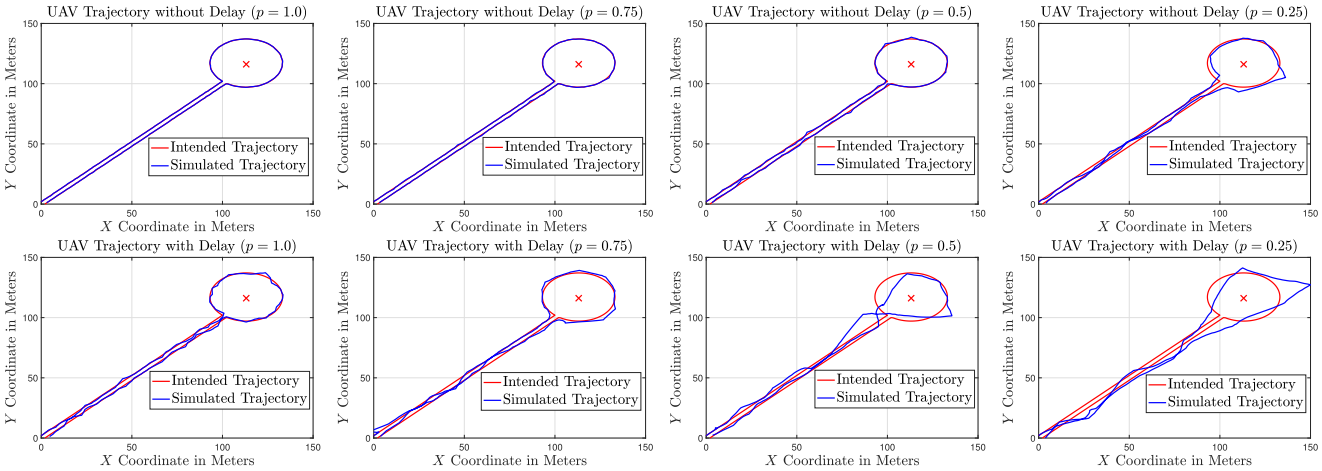

Fig. 4. Effect of reliability on the trajectory tracking performance of a UAV fog controller. $\alpha = 0.1$, $\sigma_x = 0.1$, $\sigma_v = 0.1$, and $\rho = (\sigma_x \sigma_v / 2)$ for all figures. The upper figures are for the case without delay, whereas the lower figures are for the case with delay $M = 3\Delta t$.

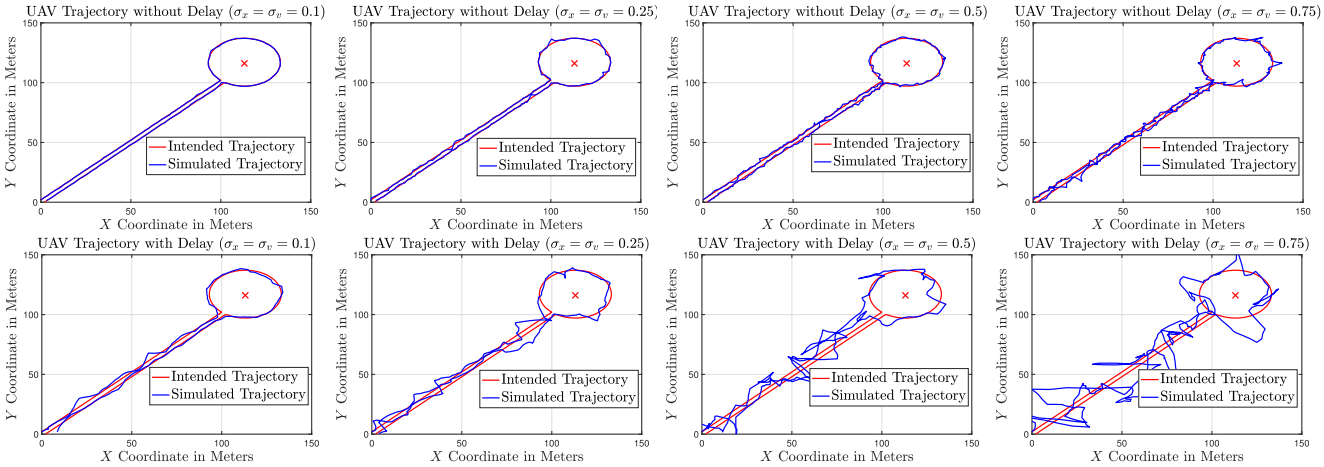

Fig. 5. Effect of environmental disturbances on the trajectory tracking performance of a UAV fog controller. $p = 0.75$, $\alpha = 0.1$, and $\rho = (\sigma_x \sigma_v / 2)$ for all figures. The upper figures are for the case without delay, whereas the lower figures are for the case with delay $M = 3\Delta t$.

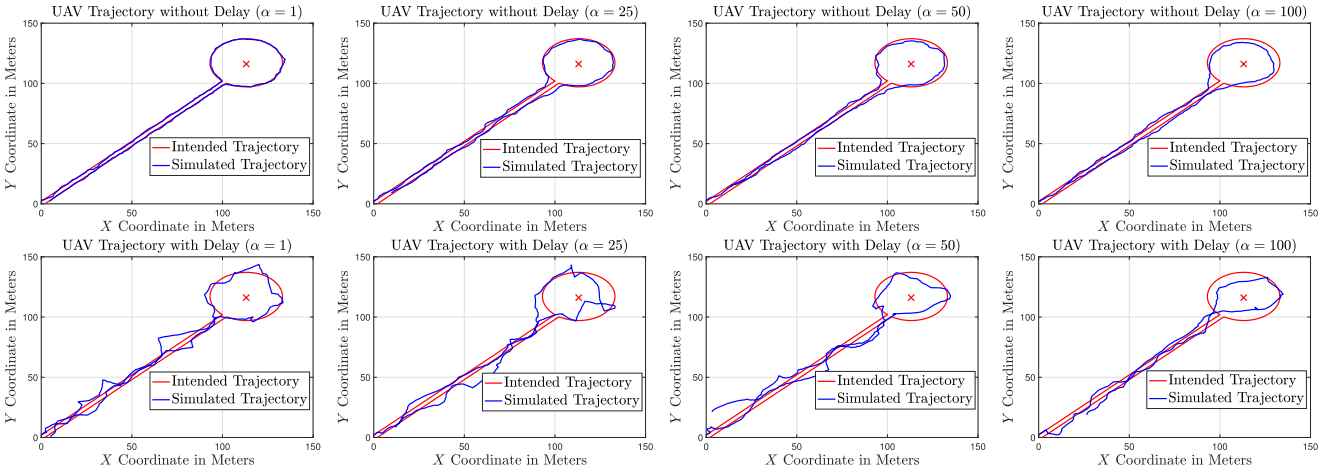

Fig. 6. Effect of the energy weight parameter α on the trajectory tracking performance of a UAV fog controller. $p = 0.75$, $\sigma_x = 0.25$, $\sigma_v = 0.25$, and $\rho = (\sigma_x \sigma_v / 2)$ for all figures. The upper figures are for the case without delay, whereas the lower figures are for the case with delay $M = 3\Delta t$.

In the upper figures, we again observe that our optimum fog controller is able to stabilize the UAV around the predefined desired trajectory, albeit having more jitters around the way-points with harsher environmental conditions. In particular, we see that there is a constant fight between efforts from the fog controller to track the desired path and random

disturbances due to environmental conditions such as wind to deviate from the desired path. In all cases, the UAV does never go uncontrolled, which is a positive indication for the perfectly matched fog controller from the perspective of preventing potential collisions with other vehicles or obstacles existing in close geographical distances. In the lower figures in Fig. 5, on the other hand, we observe that the jitters due to environmental disturbances are more pronounced for the fog controller with delay. The UAV motion almost resembles a random walk around the desired trajectory, which is certainly not desirable for many mission critical applications. This observation is mainly because of the accumulation of random disturbances until a delay-spread corrective action against trajectory deviations is taken by the fog controller. These results signify the importance of dynamic provisioning of quality of control service over fog through a lever to adjust fog node reliability and delay, especially in cases when the jitter around a desired trajectory is detrimental for the mission executed by the UAV such as surveillance monitoring of a geographical region, a remote first-aid operation and a remote swimmer rescue operation by means of drones.

In Fig. 6, we study the effect of the system-level parameter α on the trajectory tracking performance of the UAV fog controller. In this figure, we set p, σ_x, and σ_v to 0.75, 0.25, and 0.25, respectively. We chose ρ to have a correlation coefficient of 0.5 in all cases. The parameter α helps us to adjust the weight associated with the total energy spent during the journey of the UAV around the desired trajectory. By increasing the value of α, the importance ranking of the energy spent rises with respect to the relative importance of how well the UAV tracks its trajectory. This change of perspective results in deteriorations in the trajectory tracking performance of the UAV controlled over fog, as illustrated in Fig. 6. In particular, the already poor trajectory tracking performance of the UAV fog controller with delay becomes much worse as such the UAV deviates significantly from the desired trajectory with increasing values of α (i.e., smaller and distorted circles around the target and incomplete paths) when there is delay to communicate measurements and control signals between the virtual fog controller and the velocity control unit of the UAV. An important engineering implication of these results is the more evident importance of the delay when the energy is at stake to control a UAV over the fog.

VIII. Discussion

In this section, we provide a further discussion of our results and some other potential applications.

A. Linear IoT Control Systems

There is a recent surge of interest in control applications of the IoT technology and cyber-physical systems as well as related security and privacy issues [60]–[65]. Similarly, we have also considered an IoT control system in this paper but assumed that the controller software is virtualized over a fog endpoint existing in a fog computing environment. To discover design insights for control-as-a-service and identify fundamental tradeoffs between latency and reliability in such a fog computing environment, we have studied the behavior of optimum stochastic control policies in the form of a modified linear quadratic regulator (LQR).

LQR is an important tool in a more general class of optimal control techniques [16], [55]. We have illustrated an application of our results for the case of a drone trajectory tracking problem in Section VII. In addition to the drone trajectory tracking problem, the other well-known applications of LQR include aircraft motion control (linearized around equilibrium flight conditions), missile trajectory tracking problem, attitude control of a spacecraft and robot control [55], [66]. These examples, although not directly related to the IoT field, are given for expository purposes to indicate the abundance and importance of physical systems with linear state update dynamics in control theory.

Considering similarity of fundamental physical laws governing real-life dynamical processes, they also indicate the relevance of linear system update dynamics as our modeling assumption for IoT node processes in this paper as well as in other studies [65], [67]–[69]. In [65], a security analysis of linear IoT control systems in terms of their stability, controllability and observability under outsider attacks is performed. In [67], a linear PD-controller is proposed to control the tail behavior of a biologically inspired flying robot. In [68], an optimal H_∞ controller is proposed for a multiscale linear IoT system with quadratic cost. Its performance is compared with that of LQR in a hierarchical building temperature control problem. In [69], a wireless power transfer system is modeled as a state-space linear time-invariant system monitored and controlled by an IoT network. Kalman filtering is used for state estimation and LQR is used to control the wireless power transfer system. Indeed, the model in [69] perfectly fits into our framework, where the controller is virtualized at a fog server closer to the network edge. More generally, in these use-cases and in many others where the system dynamics can be modeled as linear or can be linearized around an equilibrium point, our results will apply directly.

B. Design Implications and Parameter Identification

In cases where the IoT node dynamics are linear (some examples of which are given above), the analytical expressions we have derived can help to determine the location of the fog server for executing the controller logic as follows. Theorems 1 and 5 provide the cost values with and without perfect match when the IoT node states are *fully* observed at the fog server. Comparing these cost values as a function of reliability and latency parameters, we can pinpoint the location of the fog execution point achieving the minimum cost along the cloud-to-things continuum to place the controller logic. Similarly, we can use Theorems 3 and 7 to determine the location of the fog execution point when the IoT node states are *partially* observed at the fog server.

These are practical and quantitative design rules to provision control services over fog, eliminating the costly and lengthy trial-and-error process and system-level simulations in IoT control system design. However, the parameters for the linear state-space IoT node model in (1) and the appropriate

cost function in (6) do still need to be determined for each use-case separately based on the system dynamics and control objectives. This is done for the drone trajectory tracking problem explicitly. Some other examples in the previous work are also cited but the details are not included.

Similarly, latency and reliability parameters must also be determined for each use-case more comprehensively than in our small-scale experiments by considering the level of congestion in fog computing systems. Some target latency and reliability values are available for classical and emerging application scenarios in the literature, with latency values ranging from seconds (e.g., smart-grid) to milliseconds (e.g., robotics, virtual/augmented reality, and industrial control) and reliability values ranging from 10^{-2} (e.g., VoIP) to 10^{-8} (e.g., industrial control) [70]–[72]. These application specific requirements provide a good starting point for pruning the fog execution points unable to satisfy them. However, they do not provide any additional insight about typical latency and reliability values expected to occur in heterogeneous and dynamically changing fog computing systems. The statistical characterization of latency and reliability in fog computing through extensive data collection and experimentation under varying operating conditions is an important research direction. This point requires a more comprehensive experimental study and collective effort from researchers in the field. A recent work to characterize task completion latencies in fog computing can be found in [73].

C. Iterative LQR and Guided Policy Search

Beyond the conventional and emerging applications cited above, the class of linear controllers we have studied in this paper also plays a major, and perhaps more exciting, role in control of nonlinear IoT systems [74]–[77]. A modified version of the LQR problem is iteratively solved to obtain a locally optimal controller for controlling a two-link arm and a group of muscle actuators having nonlinear state update dynamics in [74]. The linearization is based on the Jacobian of the state update function around the system trajectory obtained in the previous iteration. The resulting iterative LQR-based control policy performs better than nonlinear controllers derived based on the Pontryagin's minimum principle, both in terms of its computational power and cost value.

Series of papers [75]–[77] considered guided policy search methods for robots to learn visuomotor control policies. The input is a raw image data, the control policy is a set of weights of the deep convolutional neural network and the output is a set of motor torque commands. The optimum control policy and state update dynamics are highly nonlinear. However, when the control policy is trained to learn the trajectories generated by the LQR iteratively, it is shown to converge to a locally optimum control policy with sufficient level of confidence in accomplishing the tasks such as shape sorting, screwing a cap onto a bottle and placing a coat hanger. These examples indicate the potential of our results, albeit with further effort, to guide the placement of virtual fog controllers even in cases where the IoT node dynamics are nonlinear. Finally, we have only considered discrete-time IoT node processes in this paper, which is possible to obtain by sampling a continuous-time process. The direct analysis of the continuous-time case can be carried out by using either the Hamilton–Jacobi–Bellman equation or the calculus of variations [55].

IX. Conclusion

Fog computing presents two levers of reliability and latency to regulate the performance of virtual control services to enable/manage smarter IoT endpoints over a network. In this paper, we have introduced a framework to investigate the potential of fog computing for this end. Specifically, we have derived optimum control policies and the resulting min-cost performance for controlling stochastic IoT node processes by considering service reliability and communication/computation latency over a fog network. Our results reveal the way in which reliability and latency influence the quality of virtual control services over fog. In particular, it has been observed that latency is more critical than reliability in a fog computing environment since it determines both the frequency of corrective control actions and the timeliness of state measurements. These results have been illustrated for a drone trajectory tracking control problem.

This paper offers an initial step to discover the utility of fog computing for virtual control applications, with several important future research directions remaining. First, this paper does not consider how to provision control services for multitenant control applications running at the same fog endpoint. In such cases, a performance criterion must be jointly optimized over multiple clients by considering their service blockage probabilities and latencies. Second, an imperfect match between the virtual fog controller and the IoT node process introduces only an initial setup delay without impacting the frequency of corrective control to a large extent in some applications. In such cases, our analysis presented in Section VI still applies, but with a virtual estimator and controller running at all times after the initial setup delay. Over a large time-horizon, the negative effects of initial setup delay can be counteracted and the reliability may emerge more detrimental than latency in these cases. Finally, extensions of our results in this paper to nonlinear IoT node processes are also of interest for control applications in which the linearity assumption in (1) does not hold or a reasonable linear approximation for the IoT node process cannot be obtained.

Appendix
Proof of Theorem 2

Consider two different systems, the first one with Markov transition probabilities $1-q$ and q, and the second one with p and q. Consider the optimum control $\boldsymbol{\pi}^\star$ achieving $J^{\star(1-q,q)}(\boldsymbol{x}_0, \tau_0)$ for the first system. We apply the same control to system 2, although being suboptimum, to obtain the upper bound. Let $J_k^{\star(1-q,q)}(\boldsymbol{x}_k, \tau_k)$ and $J_k^{(p,q)}(\boldsymbol{x}_k, \tau_k)$ be the cost-to-go values for systems 1 and 2 under $\boldsymbol{\pi}^\star$, starting at $\boldsymbol{x}_k$ and τ_k. It is easy to see that

$$J_k^{(p,q)}(\boldsymbol{x}_k, 1) \leq J_k^{\star(1-q,q)}(\boldsymbol{x}_k, 1) \tag{22}$$

and

$$J_k^{(p,q)}(\boldsymbol{x}_k, 0) \leq J_k^{\star(1-q,q)}(\boldsymbol{x}_k, 0) \tag{23}$$

for $k = N-1, N$. Assume the same holds for $k+1 \leq N-1$ as an inductive argument. Then, we have

$$\begin{aligned} J_k^{(p,q)}(\boldsymbol{x}_k, 1) = \boldsymbol{x}_k^\top \boldsymbol{Q}_k \boldsymbol{x}_k + \left(\boldsymbol{u}_k^\star\right)^\top \boldsymbol{R}_k \boldsymbol{u}_k^\star \\ + p\mathsf{E}_{(\boldsymbol{x}_k,1)}\left[J_{k+1}^{(p,q)}(\boldsymbol{x}_{k+1}, 1)\middle|\boldsymbol{u}_k^\star\right] \\ + (1-p)\mathsf{E}_{(\boldsymbol{x}_k,1)}\left[J_{k+1}^{(p,q)}(\boldsymbol{x}_{k+1}, 0)\middle|\boldsymbol{u}_k^\star\right] \end{aligned}$$

where $\boldsymbol{u}_k^\star$ is the optimum control applied to system 1 at time k after observing $\boldsymbol{x}_k$. Using the inductive argument and observing that $J_k^{\star(1-q,q)}(\boldsymbol{x}_k, 1) \leq J_k^{\star(1-q,q)}(\boldsymbol{x}_k, 0)$, we have

$$\begin{aligned} J_k^{(p,q)}(\boldsymbol{x}_k, 1) &\leq \boldsymbol{x}_k^\top \boldsymbol{Q}_k \boldsymbol{x}_k + \left(\boldsymbol{u}_k^\star\right)^\top \boldsymbol{R}_k \boldsymbol{u}_k^\star \\ &\quad + q\mathsf{E}_{(\boldsymbol{x}_k,1)}\left[J_{k+1}^{\star(1-q,q)}(\boldsymbol{x}_{k+1}, 0)\middle|\boldsymbol{u}_k^\star\right] \\ &\quad + (1-q)\mathsf{E}_{(\boldsymbol{x}_k,1)}\left[J_{k+1}^{\star(1-q,q)}(\boldsymbol{x}_{k+1}, 1)\middle|\boldsymbol{u}_k^\star\right] \\ &= J_k^{\star(1-q,q)}(\boldsymbol{x}_k, 1). \end{aligned}$$

Repeating the same steps for $J_k^{(p,q)}(\boldsymbol{x}_k, 0)$ shows that $J_k^{\star(p,q)}(\boldsymbol{x}_k, \tau_k) \leq J_k^{\star(1-q,q)}(\boldsymbol{x}_k, \tau_k)$ since $J_k^{\star(p,q)}(\boldsymbol{x}_k, \tau_k) \leq J_k^{(p,q)}(\boldsymbol{x}_k, \tau_k)$, which proves the upper bound. Similar arguments apply for the lower bound, too.

REFERENCES

[1] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the Internet of Things," in *Proc. ACM MCC*, Aug. 2012, pp. 13–16.

[2] M. Chiang and T. Zhang, "Fog and IoT: An overview of research opportunities," *IEEE Internet Things J.*, vol. 3, no. 6, pp. 854–864, Dec. 2016.

[3] P. Levine. (Dec. 2016). *Return to the Edge and the End of Cloud Computing*. [Online]. Available: a16z.com/2016/12/16/the-end-of-cloud-computing/

[4] *AWS Greengrass*. Accessed: Jan. 2018. [Online]. Available: aws.amazon.com/greengrass

[5] *Azure IoT Edge*. Accessed: Jan. 2018. [Online]. Available: github.com/Azure/iot-edge

[6] OpenFog Consortium. (Feb. 2017). *OpenFog Reference Architecture*. [Online]. Available: www.openfogconsortium.org/ra/

[7] ETSI Industry Specification Group. *Mobile Edge Computing (MEC): Framework and Reference Architecture*. Accessed: Jan. 2018. [Online]. Available: www.etsi.org/technologies-clusters/technologies/multi-access-edge-computing

[8] H. Esen *et al.*, "Control as a service (CaaS): Cloud-based software architecture for automotive control applications," in *Proc. ACM SWEC*, Apr. 2015, pp. 13–18.

[9] Z. Qi, P. Dong, K. Ma, and N. Sargeant, "A design of in-car multi-layer communication network with Bluetooth and CAN bus," in *Proc. IEEE AMC*, Apr. 2016, pp. 323–326.

[10] A. E. Abdelaal, T. Hegazy, and M. Hefeeda, "Event-based control as a cloud service," in *Proc. Amer. Control Conf.*, Seattle, WA, USA, May 2017, pp. 1017–1023.

[11] A. Vick, J. Guhl, and J. Krüger, "Model predictive control as a service—Concept and architecture for use in cloud-based robot control," in *Proc. IEEE MMAR*, Miedzyzdroje, Poland, Aug. 2016, pp. 607–612.

[12] M. Yannuzzi *et al.*, "A new era for cities with fog computing," *IEEE Internet Comput.*, vol. 21, no. 2, pp. 54–67, Mar./Apr. 2017.

[13] M. A. A. Faruque and K. Vatanparvar, "Energy management-as-a-service over fog computing platform," *IEEE Internet Things J.*, vol. 3, no. 2, pp. 161–169, Apr. 2016.

[14] Y. Gao *et al.*, "Are cloudlets necessary?" School Comput. Sci., Carnegie Mellon Univ., Pittsburgh, PA, USA, Rep. CMU-CS-15-139, Oct. 2015.

[15] K. J. Aström and R. M. Murray, *Feedback Systems: An Introduction for Scientists and Engineers*. Princeton, NJ, USA: Princeton Univ. Press, 2010.

[16] D. Bertsekas, *Dynamic Programming and Optimal Control*, vol. 1. Belmont, MA, USA: Athena Sci., 1995.

[17] *Amazon Prime Air*. Accessed: Jan. 2018. [Online]. Available: www.amazon.com/Amazon-Prime-Air/b?node=8037720011

[18] A. Khan, E. Yanmaz, and B. Rinner, "Information exchange and decision making in micro aerial vehicle networks for cooperative search," *IEEE Trans. Control Netw. Syst.*, vol. 2, no. 4, pp. 335–347, Dec. 2015.

[19] X. Wang, A. Chowdhery, and M. Chiang, "SkyEyes: Adaptive video streaming from UAVs," in *Proc. ACM HotWireless*, Sep. 2016, pp. 2–6.

[20] E. Bregu, N. Casamassima, D. Cantoni, L. Mottola, and K. Whitehouse, "Reactive control of autonomous drones," in *Proc. ACM MobiSys*, Jun. 2016, pp. 207–219.

[21] P. B. Sujit, S. Saripalli, and J. B. Sousa, "An evaluation of UAV path following algorithms," in *Proc. IEEE ECC*, Jul. 2013, pp. 3332–3337.

[22] S. Kukreti, M. Kumar, and K. Cohen, "Genetically tuned LQR based path following for UAVs under wind disturbance," in *Proc. IEEE ICUAS*, Arlington, VA, USA, Jun. 2016, pp. 267–274.

[23] L. Tong, Y. Li, and W. Gao, "A hierarchical edge cloud architecture for mobile computing," in *Proc. IEEE INFOCOM*, San Francisco, CA, USA, May 2016, pp. 1–9.

[24] H. Tan, Z. Han, X.-Y. Li, and F. C. M. Lau, "Online job dispatching and scheduling in edge-clouds," in *Proc. IEEE INFOCOM*, Atlanta, GA, USA, May 2017, pp. 1–9.

[25] Y. Xiao and M. Krunz, "QoE and power efficiency tradeoff for fog computing networks with fog node cooperation," in *Proc. IEEE INFOCOM*, Atlanta, GA, USA, May 2017, pp. 1–9.

[26] S. Kosta, A. Aucinas, P. Hui, R. Mortier, and X. Zhang, "ThinkAir: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading," in *Proc. IEEE INFOCOM*, Orlando, FL, USA, Mar. 2012, pp. 945–953.

[27] V. Liberatore, "Integrated play-back, sensing, and networked control," in *Proc. IEEE INFOCOM*, Apr. 2006, pp. 1–12.

[28] N. J. Ploplys, P. A. Kawka, and A. G. Alleyne, "Closed-loop control over wireless networks," *IEEE Control Syst. Mag.*, vol. 24, no. 3, pp. 58–71, Jun. 2004.

[29] K. Gatsis, A. Ribeiro, and G. J. Pappas, "Control with random access wireless sensors," in *Proc. IEEE CDC*, Dec. 2015, pp. 318–323.

[30] K. Gatsis, M. Pajic, A. Ribeiro, and G. J. Pappas, "Opportunistic control over shared wireless channels," *IEEE Trans. Autom. Control*, vol. 60, no. 12, pp. 3140–3155, Dec. 2015.

[31] O. C. Imer, S. Yüksel, and T. Başar, "Optimal control of LTI systems over unreliable communication links," *Automatica*, vol. 42, no. 9, pp. 1429–1439, Sep. 2006.

[32] V. Gupta, B. Hassibi, and R. M. Murray, "Optimal LQG control across packet-dropping links," *Syst. Control Lett.*, vol. 56, no. 6, pp. 439–446, Jun. 2007.

[33] *AWS EC2 Instance Types*. Accessed: Jan. 2018. [Online]. Available: aws.amazon.com/ec2/instance-types/

[34] *AWS Lambda*. Accessed: Jan. 2018. [Online]. Available: aws.amazon.com/lambda/

[35] E. A. Brewer, "Lessons from giant-scale services," *IEEE Internet Comput.*, vol. 5, no. 4, pp. 46–55, Jul./Aug. 2001.

[36] R. Margolies *et al.*, "Energy-harvesting active networked tags (EnHANTs): Prototyping and experimentation," *ACM Trans. Sensor Netw.*, vol. 11, no. 4, Nov. 2015, Art. no. 62.

[37] B. Lucia, V. Balaji, A. Colin, K. Maeng, and E. Ruppel, "Intermittent computing: Challenges and opportunities," in *Proc. SNAPL*, 2017, pp. 1–8.

[38] *Google Cloud Functions*. Accessed: Jan. 2018. [Online]. Available: cloud.google.com/functions/

[39] *Azure Functions*. Accessed: Jan. 2018. [Online]. Available: azure.microsoft.com/en-us/services/functions/

[40] *Bluemix OpenWhisk*. Accessed: Jan. 2018. [Online]. Available: www.ibm.com/cloud-computing/bluemix/openwhisk

[41] M. Fowler. *Serverless Architectures*. Accessed: Jan. 2018. [Online]. Available: martinfowler.com/articles/serverless.html

[42] A. Ronacher. *Flask: Web Development, One Drop at a Time*. Accessed: Jan. 2018. [Online]. Available: flask.pocoo.org/

[43] *Green Unicorn: Python WSGI HTTP Server for UNIX*. Accessed: Jan. 2018. [Online]. Available: https://gunicorn.org/

[44] S. H. Strogatz, *Nonlinear Dynamics and Chaos With Applications to Physics, Biology, Chemistry, and Engineering*. Cambridge, MA, USA: Perseus Books, 1995.

[45] T. Balch and R. C. Arkin, "Behavior-based formation control for multirobot teams," *IEEE Trans. Robot. Autom.*, vol. 14, no. 6, pp. 926–939, Dec. 1998.
[46] M. S. Crouse, R. D. Nowak, and R. G. Baraniuk, "Wavelet-based statistical signal processing using hidden Markov models," *IEEE Trans. Signal Process.*, vol. 46, no. 4, pp. 886–902, Apr. 1998.
[47] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 498–519, Feb. 2001.
[48] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. New York, NY, USA: Wiley, 1991.
[49] T. Berger, *Rate Distortion Theory: A Mathematical Basis for Data Compression*. New York, NY, USA: Prentice-Hall, 1971.
[50] D. Slepian, "On bandwidth," *Proc. IEEE*, vol. 64, no. 3, pp. 292–300, Mar. 1976.
[51] R. G. Gallager, *Principles of Digital Communication*. New York, NY, USA: Cambridge Univ. Press, 2008.
[52] D. N. C. Tse and P. Viswanath, *Fundamentals of Wireless Communications*. New York, NY, USA: Cambridge Univ. Press, 2005.
[53] S. Yuksel and T. Basar, *Stochastic Networked Control Systems: Stabilization and Optimization under Information Constraints*. New York, NY, USA: Birkhäuser, 2013.
[54] S. Tatikonda, A. Sahai, and S. Mitter, "Stochastic linear control over a communication channel," *IEEE Trans. Autom. Control*, vol. 49, no. 9, pp. 1549–1561, Sep. 2004.
[55] D. E. Kirk, *Optimal Control Theory: An Introduction*. Mineola, NY, USA: Dover, 1998.
[56] R. E. Bellman and S. E. Dreyfus, *Applied Dynamic Programming*. Princeton, NJ, USA: Princeton Univ. Press, 1962.
[57] Y. Bar-Shalom and E. Tse, "Dual effect, certainty equivalence, and separation in stochastic control," *IEEE Trans. Autom. Control*, vol. AC-19, no. 5, pp. 494–500, Oct. 1974.
[58] R. Durrett, *Probability: Theory and Examples*, 2nd ed. Belmont, CA, USA: Duxbury Press, 1996.
[59] Dà-Jiāng Innovations. *DJI Matrice 100 Quadcopter*. Accessed: Jan. 2018. [Online]. Available: www.dji.com/matrice100/info
[60] J. A. Stankovic, "Research directions for the Internet of Things," *IEEE Internet Things J.*, vol. 1, no. 1, pp. 3–9, Feb. 2014.
[61] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, "Internet of Things for smart cities," *IEEE Internet Things J.*, vol. 1, no. 1, pp. 22–32, Feb. 2014.
[62] D. Li, Z. Aung, J. Williams, and A. Sanchez, "P3: Privacy preservation protocol for automatic appliance control application in smart grid," *IEEE Internet Things J.*, vol. 1, no. 5, pp. 414–429, Oct. 2014.
[63] K. Sato, Y. Kawamoto, H. Nishiyama, M. Kato, and Y. Shimizu, "A modeling technique utilizing feedback control theory for performance evaluation of IoT system in real-time," in *Proc. 7th Int. Conf. Wireless Commun. Signal Process. (WCSP)*, 2015, pp. 1–5.
[64] Y. Kawamoto *et al.*, "A feedback control-based crowd dynamics management in IoT system," *IEEE Internet Things J.*, vol. 4, no. 5, pp. 1466–1476, Oct. 2017.
[65] V. Radisavljevic-Gajic, S. Park, and D. Chasaki, "Vulnerabilities of control systems in Internet of Things applications," *IEEE Internet Things J.*, vol. 5, no. 2, pp. 1023–1032, Apr. 2018.
[66] M. Zhang and T.-J. Tarn, "Hybrid control of the pendubot," *IEEE/ASME Trans. Mechatronics*, vol. 7, no. 1, pp. 79–86, Mar. 2002.
[67] A. Ramezani, X. Shi, S.-J. Chung, and S. Hutchinson, "Bat Bot (B2), a biologically inspired flying machine," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, Stockholm, Sweden, 2016, pp. 3219–3226.
[68] H. Anwar and Q. Zhu, "Minimax robust optimal control of multiscale linear-quadratic systems," in *Proc. 51st Annu. Conf. Inf. Sci. Syst. (CISS)*, Baltimore, MD, USA, 2017, pp. 1–6.
[69] M. M. Rana, W. Xiang, E. Wang, X. Li, and B. J. Choi, "Internet of Things infrastructure for wireless power transfer systems," *IEEE Access*, vol. 6, pp. 19295–19303, 2018.
[70] M. Weiner, M. Jorgovanovic, A. Sahai, and B. Nikolié, "Design of a low-latency, high-reliability wireless communication system for control applications," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Sydney, NSW, Australia, 2014, pp. 3829–3835.
[71] M. Luvisotto, Z. Pang, and D. Dzung, "Ultra high performance wireless control for critical applications: Challenges and directions," *IEEE Trans. Ind. Informat.*, vol. 13, no. 3, pp. 1448–1459, Jun. 2017.
[72] C. Li *et al.*, "5G-based systems design for tactile Internet," *Proc. IEEE*, to be published.
[73] M. Gorlatova, H. Inaltekin, and M. Chiang, "Characterizing task completion latencies in fog computing," Dept. Elect. Comput. Eng., Duke Univ., Durham, NC, USA, Rep. arXiv:1811.02638, Nov. 2018. [Online]. Available: https://arxiv.org/abs/1811.02638
[74] W. Li and E. Todorov, "Iterative linear quadratic regulator design for nonlinear biological movement systems," in *Proc. 1st Int. Conf. Informat. Control Autom. Robot. (ICINCO)*, 2004, pp. 222–229.
[75] S. Levine and P. Abbeel, "Learning neural network policies with guided policy search under unknown dynamics," in *Proc. 28th Conf. Neural Inf. Process. Syst. (NIPS)*, Dec. 2014, pp. 1071–1079.
[76] S. Levine, N. Wagener, and P. Abbeel, "Learning contact-rich manipulation skills with guided policy search," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, Seattle, WA, USA, 2015, pp. 156–163.
[77] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *J. Mach. Learn. Res.*, vol. 17, no. 1, pp. 1334–1373, Jan. 2016.

Hazer Inaltekin (S'04–M'06) received the B.S. degree (High Hons.) in electrical and electronics engineering from Boğaziçi University, Istanbul, Turkey, in 2001, and the M.S. and Ph.D. degrees in electrical and computer engineering from Cornell University, Ithaca, NY, USA, in 2006.

He held various researcher and faculty positions in Australia, Europe, and the U.S. He is a Senior Research Fellow with the University of Melbourne, Melbourne, VIC, Australia. His current research interests include fog computing, IoT technology, wireless communications, wireless networks, social networks, game theory, and information theory.

Maria Gorlatova (S'11–GS'12–M'15) received the B.Sc. (*summa cum laude*) and M.Sc. degrees in electrical engineering from the University of Ottawa, Ottawa, ON, Canada, and the Ph.D. degree in electrical engineering from Columbia University, New York, NY, USA.

She is an Assistant Professor of Electrical and Computer Engineering and Computer Science with Duke University, Durham, NC, USA. She has several years of industry experience, and has been affiliated with Telcordia Technologies, Piscataway, NJ, USA, IBM, Armonk, NY, USA, and D. E. Shaw Research, New York, NY, USA. Her current research interests include architectures, algorithms, and protocols for emerging pervasive technologies.

Dr. Gorlatova was a recipient of the Google Anita Borg USA Fellowship, the Canadian Graduate Scholar NSERC Fellowships, and the Columbia University Presidential Fellowship. She was a co-recipient of the 2011 ACM SenSys Best Student Demonstration Award, the 2011 IEEE Communications Society Award for Advances in Communications, and the 2016 IEEE Communications Society Young Author Best Paper Award.

Mung Chiang (S'00–M'03–SM'08–F'12) was the Arthur LeGrand Doty Professor of Electrical Engineering with Princeton University, Princeton, NJ, USA, where he also served as the Director of Keller Center for Innovations in Engineering Education and the inaugural Chairman of Princeton Entrepreneurship Council. He is the John A. Edwardson Dean of the College of Engineering and the Roscoe H. George Professor of Electrical and Computer Engineering with Purdue University, West Lafayette, IN, USA. His textbook *Networked Life*, popular science book *The Power of Networks*, and online courses reached over 400 000 students since 2012. He founded the Princeton EDGE Lab in 2009, which bridges the theory-practice gap in edge networking research by spanning from proofs to prototypes. He also co-founded a few startup companies in mobile data, IoT, and AI, and co-founded the global nonprofit Open Fog Consortium.

Mr. Chiang was a recipient of the 2013 Alan T. Waterman Award for his research on networking, the highest honor to U.S. young scientists and engineers.