

Virtualized Control over Fog: Interplay Between Reliability and Latency

Hazer Inaltekin, *Member, IEEE*, Maria Gorlatova, *Member, IEEE*, 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 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 realistic 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—Fog computing, distributed systems, Internet of Things, control, reliability, latency.

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]. Peter Levine, a partner at an A-list venture capital firm Andreessen Horowitz, has recently called fog computing *the next multi-billion dollar tech market* [3]. 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],

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 Engineering, Princeton University, Princeton, NJ 08544, USA (e-mail: mariaag@princeton.edu).

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

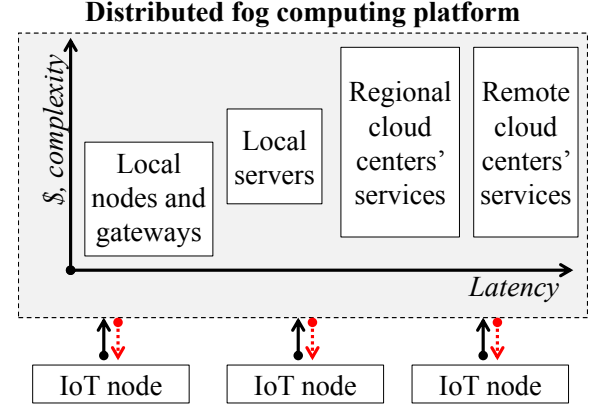


Fig. 1. In a fog computing system, control application can be run at different distributed points and as different services, with different characteristics.

[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 towards addressing this gap in the current 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 all 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 ap-

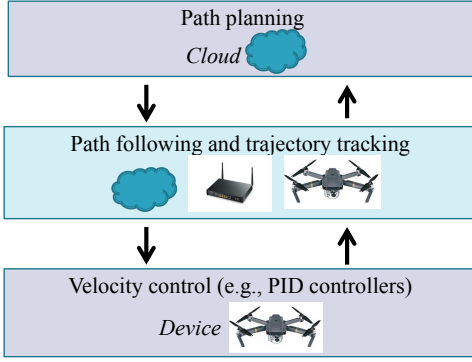


Fig. 2. Layers of control functionality for drones. While path planning belongs on the cloud and velocity control probably belongs on the device itself, trajectory tracking elements could be virtualized on different fog endpoints.

plied 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]. In drone air traffic control, while the highest-level global fleet planning decisions require the involvement of the cloud and low-level high-bandwidth velocity control needs to be done on the drone itself [20], the important trajectory tracking and path following control operations [21], [22] can be executed on multiple locations in a fog network.

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.

- 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 control software located along the cloud-to-things continuum. Under this framework, the min-cost performance of virtualized control services is obtained.
- 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.
- Based on a realistic 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 moving controller software as close as possible to the unmanned vehicle (UV).

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. 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 optimum virtual controller **without latency**, while Section VI contains parallel results for optimum virtual controller with latency. In Section VII, we present our numerical results for the UV trajectory tracking problem. Section VIII concludes the paper, with potential generalizations of our results.

II. RELATED WORK

Our results in this paper are related both to the emerging body of papers in fog computing [12], [13], [23]–[26] and to the more classical literature in control systems [10], [11], [27]–[32]. The papers [12], [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 offloading and associated computing job scheduling. In particular, Tong et al. proposed a hierarchical edge/cloud architecture in [23], and showed that the proposed architecture has a higher chance of serving peak loads from virtualized services. In [24], the authors 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. In [25], the authors investigated a job offloading problem similar to those studied in [23], [24], but by considering 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. In [26], Kosta et al. 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 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 design choice of reliability and latency, we obtain the min-cost performance of the optimum virtual fog controller. These optimum performance values can then be inputted 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], [11] considered virtualized control services over the cloud. In [27], the author 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 architectures that are important for placements of virtualized control services, and present the results of small-scale experiments with fog computing architectures.



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

Fog computing architectures are expected to include different physical links (wired, wireless, 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* [9], [33]). Cloud computing service providers are already offering a full range of service options that differ in the speed, cost, and complexity of execution [34], [35] – 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 minutes per year) [36], 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 [37], [38].

Additionally, virtualized control functionality in fog systems will need to be *placed, tuned, and moved around* automatically, without the inputs from the users. Existing commercial examples of automatic service placements in cloud computing include serverless computing mechanisms [35], [39]–[41],

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

which use auto-provisioning (“autoscaling”) mechanisms to provide robustness to spikes in service request rates at the cost of additional latency [42]. 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-dimensional 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 work. We executed the control application at different AWS Lambda [35] cloud computing service points worldwide, at a Microsoft Azure serverless Functions computing point [40], and on a local consumer-grade hardware device with an Intel Atom single-core 1.6GHz CPU. The local control service was implemented using a popular Flask [43] micro-service development framework over the built-in Flask HTTP server and over a Gunicorn WSGI HTTP Unix Server [44].

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 vary over a wide range. 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 the current 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, the 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 UV, a robotic arm, etc.) whose dynamics evolve *linearly* in discrete-time according to

$$\mathbf{x}_{k+1} = \mathbf{A}_k \mathbf{x}_k + \mathbf{B}_k \mathbf{u}_k + \mathbf{w}_k, \quad (1)$$

where $\mathbf{x}_k \in \mathbb{R}^n$ is the state vector, $\mathbf{u}_k \in \mathbb{R}^s$ is the control signal, $\mathbf{w}_k \in \mathbb{R}^n$ is the zero-mean random disturbance with covariance \mathbf{W}_k , and $\mathbf{A}_k \in \mathbb{R}^{n \times n}$ and $\mathbf{B}_k \in \mathbb{R}^{n \times s}$ are the system matrices that modulate the system states and control signals, respectively, for $k = 0, \dots, N-1$.¹ 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. The states are monitored by on-board sensors and are transmitted over a communication channel to the fog controller. The channel output is given by

$$\mathbf{z}_k = \mathbf{C}_k \mathbf{x}_k + \mathbf{v}_k, \quad (2)$$

where $\mathbf{C}_k \in \mathbb{R}^{m \times n}$ and $\mathbf{v}_k \in \mathbb{R}^m$ is the measurement-plus-channel noise. We note the affinity of this model with the classical multiple-input multiple-output (MIMO) channel model [46].² 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 \mathbf{z}_k marks a *request-for-control* and initiates the generation of a control signal at the fog endpoint.

The casual 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 exposition, we explain the rest of the system model for the case of perfect match, and relegate the details of the latter to Section VI.

¹This 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 other several design degrees of freedom such as control quality and precision. For non-linear IoT node processes, it is assumed that Hartman-Grobman theorem holds and the system dynamics can be linearized around an equilibrium point [45].

²Our model can be considered to exemplify uncoded or low-complexity coded transmissions such as simple CRC schemes so that some measurement-plus-channel noise still remains in the received state measurements, which is a practical assumption for IoT nodes limited by computational resources and battery life.

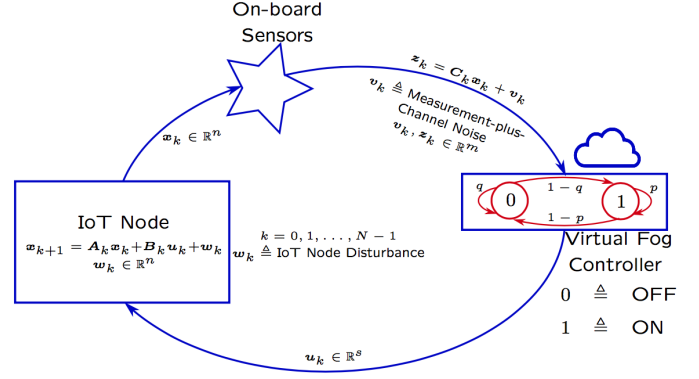


Fig. 3. Model for virtualized control over fog.

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

$$\Pr \{ \tau_{k+1} = 1 | \tau_k = 1 \} = p$$

and

$$\Pr \{ \tau_{k+1} = 1 | \tau_k = 0 \} = 1 - q,$$

where τ_k is the internal state of the fog controller at time $k = 0, \dots, N-1$. Here, the states 1 (i.e., ON state) and 0 (i.e., OFF state) indicate the capacitated occupation status (due to multiple instantiated computing processes) of the fog endpoint for provisioning the requested control service.

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

$$\mathcal{H}_k = \{ \mathbf{z}_i : \tau_i = 1, 0 \leq i \leq k \} \cup \{ \mathbf{u}_i : 0 \leq i \leq k-1 \}. \quad (3)$$

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 \mathbf{u}_k at time k , which is embodied in \mathcal{H}_k . The class of dynamic control policies of interest to us in this paper is introduced in Definition 1 below.

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

We note that $\boldsymbol{\pi}$ is an on-line rule that observes the realizations of system history and determines the control signals to be applied based on these observations. We say that a control policy is *feasible* if $\pi_k(\mathcal{H}_k, 0) = 0$ for all $\mathcal{H}_k \in \mathcal{H}_k$ and $k = 0, \dots, N-1$. That is, a feasible control policy does not output any control when the fog controller is at OFF state. An important subclass of control policies 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 \mathbf{z}_k and τ_k for $k = 0, \dots, N-1$.

B. Cost Minimization Problem

As is standard in control systems [30]–[32], [47], [48], we rank the quality of virtualized control over fog by means of a

quadratic cost function. In particular, we define **per-stage-cost** under control policy $\pi = (\pi_0, \dots, \pi_{N-1})$ at time k as

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

where \mathbf{u}_k is the control generated by π after having observed H_k , and \mathbf{Q}_k is positive semidefinite and \mathbf{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**: (i) **IoT node disturbance**, (ii) **channel noise** and (iii) **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

$$J_{\pi}^{(p,q)}(\mathbf{x}_0, \tau_0) = \mathbb{E}_{(\mathbf{x}_0, \tau_0)} \left[\mathbf{x}_N^\top \mathbf{Q}_N \mathbf{x}_N + \sum_{k=0}^{N-1} g_{\pi_k}(\mathbf{x}_k, H_k, \tau_k) \right], \quad (4)$$

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

$$J^{(p,q)}(\mathbf{x}_0, \tau_0) = \inf_{\pi \in \Pi} J_{\pi}^{(p,q)}(\mathbf{x}_0, \tau_0). \quad (5)$$

We will solve the optimization problem in (5) by utilizing the dynamic programming (DP) approach [16]. To this end, we define cost-to-go functions as

$$J_i^{(p,q)}(\mathbf{x}_i, \tau_i) = \mathbb{E}_{(\mathbf{x}_i, \tau_i)} \left[\mathbf{x}_N^\top \mathbf{Q}_N \mathbf{x}_N + \sum_{k=i}^{N-1} g_{\pi_k}(\mathbf{x}_k, H_k, \tau_k) \right]. \quad (6)$$

We note that $J_i^{(p,q)}(\mathbf{x}_i, \tau_i)$ depends on the i th tail policy π_i for π , which is defined as $\pi_i = (\pi_i, \dots, \pi_{N-1})$. The principle of optimality for DP states that the control policy π^* having optimal tail policies $\pi_i^* = (\pi_i^*, \dots, \pi_{N-1}^*)$ for minimizing the cost-to-go function $J_i^{(p,q)}(\mathbf{x}_i, \tau_i)$ for any starting state $\mathbf{x}_i \in \mathbb{R}^n$ and $\tau_i \in \{0, 1\}$ for all $i \in \{0, \dots, N-1\}$ solves the main DP problem in (5). 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, 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 measurement and channel distortion can be ignored, i.e., $\mathbf{v}_k = \mathbf{0}$ and \mathbf{C}_k is identity for all $k = 0, \dots, N-1$. Hence, the network does not mix-up 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 π^* and the min-cost performance under π^* .

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

$$J^{(p,q)}(\mathbf{x}_0, \tau_0) = \mathbf{x}_0^\top (\mathbf{L}_0 - \mathbf{A}_0 \mathbf{1}_{\{\tau_0=1\}}) \mathbf{x}_0 + \sum_{k=0}^{N-1} \text{Tr}(\mathbf{K}_{k+1} \mathbf{W}_k) \quad (7)$$

and

$$\pi_k^*(\mathbf{x}_k, \tau_k) = -\mathbf{V}_k \mathbf{x}_k \mathbf{1}_{\{\tau_k=1\}}, \quad (8)$$

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

$$\begin{aligned} \mathbf{K}_N &= \mathbf{Q}_N \\ \mathbf{K}_k &= \mathbf{L}_k - p \mathbf{A}_k \\ \mathbf{L}_k &= \mathbf{Q}_k + \mathbf{A}_k^\top \mathbf{K}_{k+1} \mathbf{A}_k, \text{ and} \\ \mathbf{A}_k &= \mathbf{A}_k^\top \mathbf{K}_{k+1} \mathbf{B}_k \mathbf{V}_k. \end{aligned}$$

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

$$J_N^{(p,q)}(\mathbf{x}_N, \tau_N) = \mathbf{x}_N^\top \mathbf{Q}_N \mathbf{x}_N \quad (9)$$

and

$$J_k^{(p,q)}(\mathbf{x}_k, \tau_k) = \mathbf{x}_k^\top \mathbf{Q}_k \mathbf{x}_k + \min_{\mathbf{u}_k \in \mathbb{R}^s} \left\{ \mathbf{u}_k^\top \mathbf{R}_k \mathbf{u}_k + \mathbb{E}_{(\mathbf{x}_k, \tau_k)} \left[J_{k+1}^{(p,q)}(\mathbf{x}_{k+1}, \tau_{k+1}) | \mathbf{u}_k \right] \right\} \quad (10)$$

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

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

and

$$J_{N-1}^{(p,q)}(\mathbf{x}_{N-1}, 1) = \mathbf{x}_{N-1}^\top (\mathbf{L}_{N-1} - \mathbf{A}_{N-1}) \mathbf{x}_{N-1} + \text{Tr}(\mathbf{K}_N \mathbf{W}_{N-1}).$$

On the other hand, no control is applied and the matrix \mathbf{A}_{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 (8) and the minimum cost-to-go expression below

$$\begin{aligned} J_{N-2}^{*(p,q)}(\mathbf{x}_{N-2}, \tau_{N-2}) \\ = \mathbf{x}_{N-2}^\top (\mathbf{L}_{N-2} - \mathbf{\Lambda}_{N-2} \mathbf{1}_{\{\tau_{N-2}=1\}}) \mathbf{x}_{N-2} \\ + \text{Tr}(\mathbf{K}_{N-1} \mathbf{W}_{N-2}) + \text{Tr}(\mathbf{K}_N \mathbf{W}_{N-1}). \end{aligned}$$

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 nodes. The effect of the fog endpoint reliability appears as a multiplicative coefficient in the definition of \mathbf{K}_k matrices. In particular, as p increases, the additive cost terms $\text{Tr}(\mathbf{K}_{k+1} \mathbf{W}_k)$, i.e., matrix traces, decrease and we start to have a smaller cost value in (7).³ We note that this is also the same virtual control service structure implemented over AWS Lambda, Microsoft Azure, Flask HTTP and Unicorn 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 $k = 0, \dots, N-1$, 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,

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

and

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

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

Proof: Intuitively, decreasing p results in 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 Appendix A. ■

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], [49], which leads to our next theorem.

³This follows from observing the fact that the matrices \mathbf{K}_k , \mathbf{L}_k and $\mathbf{\Lambda}_k$ are positive semi-definite for all $k = 0, \dots, N-1$.

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

$$\begin{aligned} J^{*(p,q)}(\mathbf{x}_0, \tau_0) &= \mathbf{x}_0^\top (\mathbf{L}_0 - \mathbf{\Lambda}_0 \mathbf{1}_{\{\tau_0=1\}}) \mathbf{x}_0 \\ &+ \sum_{k=0}^{N-1} \text{Tr}(\mathbf{K}_{k+1} \mathbf{W}_k) + \mathbb{E}_{(\mathbf{x}_0, \tau_0)} [\epsilon_0^\top \mathbf{\Lambda}_0 \mathbf{1}_{\{\tau_0=1\}} \epsilon_0] \\ &+ p \sum_{k=1}^{N-1} \mathbb{E}_{(\mathbf{x}_0, \tau_0)} [\epsilon_k^\top \mathbf{\Lambda}_k \epsilon_k] \end{aligned} \quad (11)$$

and

$$\pi_k^*(H_k, \tau_k) = -\mathbf{V}_k \mathbb{E}[\mathbf{x}_k | H_k] \mathbf{1}_{\{\tau_k=1\}}, \quad (12)$$

where $\epsilon_k = \mathbf{x}_k - \mathbb{E}[\mathbf{x}_k | H_k]$ and the matrices \mathbf{V}_k , \mathbf{K}_k , $\mathbf{\Lambda}_k$ and \mathbf{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 $\mathbb{E}[\mathbb{E}[X | \mathcal{F}_2] | \mathcal{F}_1] = \mathbb{E}[X | \mathcal{F}_1]$ for any two nested σ -algebras $\mathcal{F}_1 \subseteq \mathcal{F}_2$ from [50]. ■

The first two terms in (11) 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 (12) is similar to the one in Theorem 1, except the non-linear estimator $\mathbb{E}[\mathbf{x}_k | H_k]$ 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 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,

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

and

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

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

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

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: (i) *forward delay* M_F and (ii) *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 optimum control is included in M_B in the sequel. The total delay incurred in both ways is then equal to $M = M_F + M_B$.⁴ 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 $c \triangleq \frac{N-a}{M}$, where $a = N \bmod M$ if N is not an integer multiple of M , and $a = M$ otherwise. Then, the optimum control policy $\pi^* = (\pi_0^*, \dots, \pi_{N-1}^*)$ and $J^{*(p,q)}(x_0, \tau_0)$ are given by

$$J^{*(p,q)}(x_0, \tau_0) = x_0^\top L_0 x_0 + \sum_{k=0}^{N-1} \text{Tr}(K_{k+1} W_k) + p \sum_{k=0}^{cM-1} \text{Tr}(P_{k+1} W_k) \quad (13)$$

and

$$\pi_k^*(\lambda_k, \tau_{k-M_B}) = \begin{cases} 0 & \text{if } k \neq 0 \bmod M \\ -V_k \mathbb{E}[x_k | \lambda_k] 1_{\{\tau_{k-M_B}=1\}} & \text{if } k = 0 \bmod M \end{cases} \quad (14)$$

for $k \in \{1, \dots, N-1\}$, where $\lambda_k = (x_{k-M}, u_{k-M})$ (for $k \geq M$), $V_k = (R_k + B_k^\top K_{k+1} B_k)^{-1} B_k^\top K_{k+1} A_k$ and the matrices K_k and P_k are given recursively as

$$\begin{aligned} K_N &= Q_N \\ K_k &= \begin{cases} L_k & \text{if } k \neq 0 \bmod M \\ L_k - p \Lambda_k & \text{if } k = 0 \bmod M \end{cases} \\ L_k &= Q_k + A_k^\top K_{k+1} A_k, \text{ and} \\ \Lambda_k &= A_k^\top K_{k+1} B_k V_k \end{aligned}$$

⁴Here, 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 becomes deterministic.

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

$$\begin{aligned} P_{cM} &= \Lambda_{cM} \\ P_k &= \begin{cases} A_k^\top P_{k+1} A_k & \text{if } k \neq 0 \bmod M \\ \Lambda_k & \text{if } k = 0 \bmod M \end{cases} \end{aligned}$$

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

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 arrives from time $cM + 1$ to N . Hence, we can write

$$J_i^{*(p,q)}(x_i, \tau_i) = x_i^\top L_i x_i + \sum_{k=i}^{N-1} \text{Tr}(K_{k+1} W_k) \quad (15)$$

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

$$\min_{u \in \mathbb{R}^s} \left\{ u^\top R_i u + \mathbb{E} \left[x_{i+1}^\top J_{i+1}^{*(p,q)}(x_{i+1}, \tau_{i+1}) | u \right] | \lambda_i \right\}$$

for $i = cM$. Using (15) and system linearity, the above minimization leads to

$$\pi_{cM}^*(\lambda_{cM}, \tau_{cM-M_B}) = -V_{cM} \mathbb{E}[x_{cM} | \lambda_{cM}] 1_{\{\tau_{cM-M_B}=1\}}$$

and, with a slight abuse of notation⁵,

$$\begin{aligned} J_{cM}^{*(p,q)}(x_{cM}, \tau_{cM}) &= x_{cM}^\top (L_{cM} - \Lambda_{cM} 1_{\{\tau_{cM-M_B}=1\}}) x_{cM} \\ &+ \sum_{k=cM}^{N-1} \text{Tr}(K_{k+1} W_k) + \epsilon_{cM}^\top P_{cM} 1_{\{\tau_{cM-M_B}=1\}} \epsilon_{cM}, \end{aligned}$$

where $\epsilon_{cM} = x_{cM} - \mathbb{E}[x_{cM} | \lambda_{cM}]$. Now observing that no control arrives at time $cM - 1$ and using the symmetry assumption as well as system linearity, we have

$$\begin{aligned} J_i^{*(p,q)}(x_i, \tau_i) &= x_i^\top L_i x_i + \sum_{k=i}^{N-1} \text{Tr}(K_{k+1} W_k) \\ &+ p \text{Tr}(P_{i+1} W_i) + p \epsilon_i^\top P_i \epsilon_i \end{aligned}$$

for $i = cM - 1$ and $\epsilon_{cM-1} = x_{cM-1} - \mathbb{E}[x_{cM-1} | \lambda_{cM}]$, which is the *residual* error from estimation at time cM .

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

$$\begin{aligned} J_i^{*(p,q)}(x_i, \tau_i) &= x_i^\top (L_i - \Lambda_i 1_{\{\tau_{i-M_B}=1\}}) x_i + \sum_{k=i}^{N-1} \text{Tr}(K_{k+1} W_k) \\ &+ p \sum_{k=i}^{cM-1} \text{Tr}(P_{k+1} W_k) + \epsilon_i^\top P_i 1_{\{\tau_{i-M_B}=1\}} \epsilon_i, \end{aligned}$$

⁵ $J_{cM}^{*(p,q)}(x_{cM}, \tau_{cM})$ actually depends on τ_{cM-M_B} but we have chosen to use above notation for the sake notational consistency.

where $\epsilon_i = x_i - E[x_i | \lambda_i]$ 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 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 (12), 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 (13) represents the collateral impact of latency on the min-cost performance due to delayed state measurements, which would disappear for an hypothetical perfect estimator.⁶

Virtual Controller Placement: Equations (7) and (13) in Theorems 1 and 5 provide a pair of quantitative 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,

$$J^{*(p,q)}(x_0, \tau_0) \leq J^{*(1-q,q)}(x_0, \tau_0)$$

and

$$J^{*(p,q)}(x_0, \tau_0) \geq J^{*(p,1-p)}(x_0, \tau_0).$$

Moreover, the optimum control achieving $J^{*(1-q,q)}(x_0, \tau_0)$ with *full* state information and *imperfect* match also attains a performance in between these two bounds.

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

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 (3) in an obvious way to include only delayed measurements and repeating the same steps above, we obtain the following theorems.

⁶ $J^{*(p,q)}(x_0, \tau_0)$ in (13) does not depend on τ_0 since the system forgets about the starting fog node state after one unit delay due to symmetry assumption.

Theorem 7: Assume $p = 1 - q$, $N \geq M \geq 1$ and the measured states can be *partially* observed with *imperfect* match. Define $c \triangleq \frac{N-a}{M}$, where $a = N \bmod M$ if N is not an integer multiple of M , and $a = M$ otherwise. Then, the optimum control policy $\pi^* = (\pi_0^*, \dots, \pi_{N-1}^*)$ and $J^{*(p,q)}(x_0, \tau_0)$ are given by

$$\begin{aligned} J^{*(p,q)}(x_0, \tau_0) &= x_0^\top L_0 x_0 + \sum_{k=0}^{N-1} \text{Tr}(K_{k+1} W_k) + p \sum_{k=0}^{cM-1} \text{Tr}(P_{k+1} W_k) \\ &\quad + p \sum_{k=1}^{c-1} E_{(x_0, \tau_0)} [\epsilon_{kM}^\top P_{kM} \epsilon_{kM}] \end{aligned} \quad (16)$$

and

$$\begin{aligned} \pi_k^*(H_k, \tau_{k-M_B}) &= \begin{cases} 0 & \text{if } k \neq 0 \bmod M \\ -V_k E[x_k | H_k] 1_{\{\tau_{k-M_B}=1\}} & \text{if } k = 0 \bmod M \end{cases} \end{aligned} \quad (17)$$

for $k \in \{1, \dots, N-1\}$, where the matrices V_k , K_k , Λ_k , L_k and P_k are as defined in Theorem 5, and $\epsilon_{kM} = x_{kM} - E[x_{kM} | H_{(k+1)M}]$ for $k \in \{1, \dots, c-1\}$.

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 $E[E[X | \mathcal{F}_2] | \mathcal{F}_1] = E[X | \mathcal{F}_1]$ for any two nested σ -algebras $\mathcal{F}_1 \subseteq \mathcal{F}_2$ from [50]. ■

The last summation in (16) 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 (17) is non-linear, which can be replaced with its linear version through practical Kalman filtering implementation at the expense of some loss of optimality.

Virtual Controller Placement: Equations (11) and (16) in Theorems 3 and 7 provide a pair of quantitative 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,

$$J^{*(p,q)}(x_0, \tau_0) \leq J^{*(1-q,q)}(x_0, \tau_0)$$

and

$$J^{*(p,q)}(x_0, \tau_0) \geq J^{*(p,1-p)}(x_0, \tau_0).$$

Moreover, the optimum control achieving $J^{*(1-q,q)}(x_0, \tau_0)$ with *partial* state information and *imperfect* match also attains a performance in between these two bounds.

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

VII. UV 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 UVs. While our work 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 [51] 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 UV Control

We consider a planar motion at some certain altitude determined through a sequence of way-points $\{\bar{x}_k\}_{k=0}^N \subset \mathbb{R}^2$. \bar{x}_k represents the desired position of the UV at time $k\Delta t$. Here, Δt is our basic discrete-time unit to communicate location information (obtained through GPS sensors) and control signals between the fog server and UV.⁷

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

$$\begin{pmatrix} \mathbf{x}_{k+1} \\ \mathbf{v}_{k+1} \end{pmatrix} = \begin{pmatrix} \mathbf{x}_k + \Delta t(\mathbf{v}_k + \mathbf{u}_k) \\ \mathbf{v}_k + \mathbf{u}_k \end{pmatrix} + \begin{pmatrix} \mathbf{w}_k^x \\ \mathbf{w}_k^v \end{pmatrix}, \quad (18)$$

where \mathbf{w}_k^x and \mathbf{w}_k^v are random (possibly correlated) disturbances affecting the location and velocity of the UV due to environmental conditions such as wind and rain, respectively. After some manipulations, (18) can be written as

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

where $\mathbf{e}_k = \mathbf{x}_k - \bar{\mathbf{x}}_k$, $\mathbf{A}_k = \begin{pmatrix} \mathbf{I}_2 & \Delta t \mathbf{I}_2 \\ \mathbf{0} & \mathbf{I}_2 \end{pmatrix}$ and $\mathbf{B}_k = \begin{pmatrix} \Delta t \mathbf{I}_2 \\ \mathbf{I}_2 \end{pmatrix}$ for $k = 0, \dots, N-1$.⁸ This is similar to the linear IoT node model in Section IV, with drift terms providing desired trajectory information. Hence, the optimum control will direct the UV along the desired path while trying to minimize drifted state measurements. To this end, we minimize the following cost $J = \sum_{k=0}^N \mathbb{E} [|\mathbf{e}_k|^2 + \alpha (|\mathbf{v}_k|^2 + |\mathbf{u}_k|^2)]$ in an attempt to balance trajectory deviations and energy expenditure, which leads to $\mathbf{Q}_k = \begin{pmatrix} \mathbf{I}_2 & \mathbf{0} \\ \mathbf{0} & \alpha \mathbf{I}_2 \end{pmatrix}$, $\mathbf{R}_k = \alpha \mathbf{I}_2$ and $\alpha \geq 0$ being a design parameter. In this cost expression,

⁷The model can be extended to include time-varying discrete-time units. However, we do not pursue this direction in the current paper for the sake of notational simplicity.

⁸More explicitly, the randomly varying velocity process of the UV can be written as $\mathbf{V}(t) = \mathbf{v}_k + \mathbf{u}_k + \mathbf{W}(t)$ for $k\Delta t < t \leq (k+1)\Delta t$, where $\mathbf{W}(t)$ is the stochastic disturbance process inflicting the UV motion. Then, \mathbf{w}_k^x is the integral of $\mathbf{W}(t)$ from $k\Delta t$ to $(k+1)\Delta t$, and \mathbf{w}_k^v is the value of $\mathbf{W}(t)$ at time $(k+1)\Delta t$.

we note that the terms containing $|\mathbf{e}_k|^2$ 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 UV 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 $\Sigma = \begin{pmatrix} \sigma_x^2 \mathbf{I}_2 & \rho \mathbf{I}_2 \\ \rho \mathbf{I}_2 & \sigma_v^2 \mathbf{I}_2 \end{pmatrix}$.

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_v 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 UV 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 UV almost perfectly tracks the desired trajectory for values of p above 0.5, which can be 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 UV around the desired trajectory starting from $p = 0.5$ and downwards 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 UV 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. In the upper figures, we again observe that our optimum fog controller is able to stabilize the UV 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 UV 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. 4, on the other hand, we observe that the jitters due to environmental disturbances are more pronounced for the fog controller with delay. The UV motion almost resembles a random walk around the desired trajectory, which is certainly not desirable for many mission critical applications. This observation is mainly

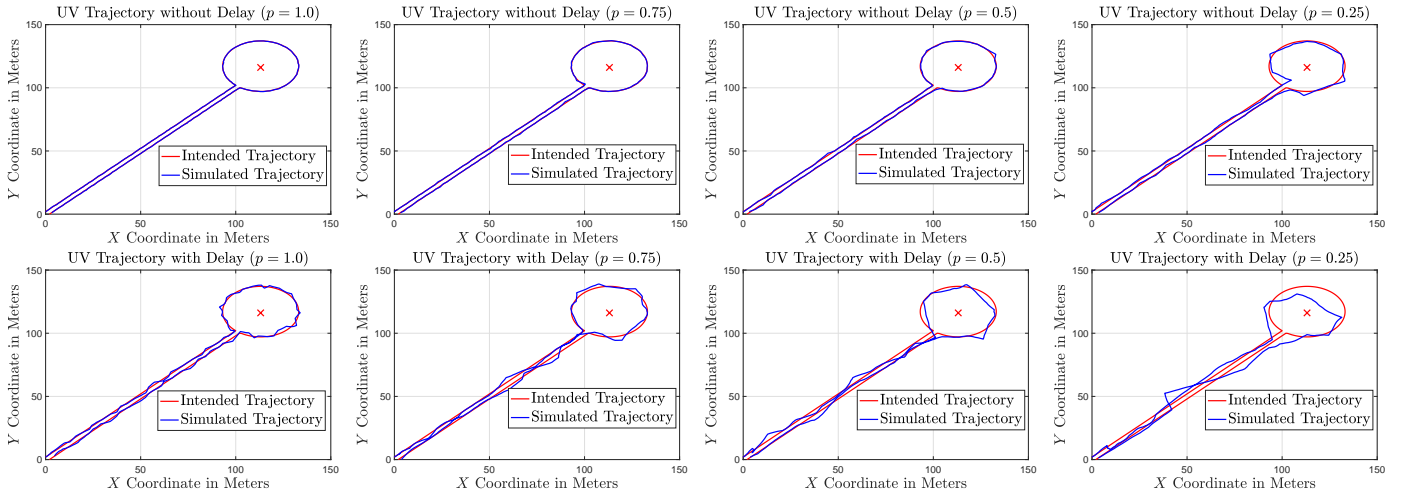


Fig. 4. Effect of reliability on the trajectory tracking performance of a UV fog controller. ($\alpha = 0.1$, $\sigma_x = 0.1$, $\sigma_v = 0.1$ and $\rho = \frac{\sigma_x \sigma_v}{2}$ for all figures. The upper figures are for the case without delay, whereas the lower figures are for the case with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

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 UV 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 UV 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 UV 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 UV tracks its trajectory. This change of perspective results in deteriorations in the trajectory tracking performance of the UV controlled over fog, as illustrated in Fig. 6. In particular, the already poor tracking performance of the UV fog controller with delay becomes much worse as such the UV deviates significantly from the desired trajectory with increasing values of α (i.e., smaller 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 UV. An important engineering implication of these results is the more evident importance of the delay when the energy is at stake to control a UV over the fog.

VIII. CONCLUSIONS, FUTURE WORK AND DISCUSSIONS

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 work offers an initial step to discover the utility of fog computing for virtual control applications, with several important future research directions remaining. Firstly, this paper does not consider how to provision control services for multi-tenant 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. Secondly, 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 non-linear IoT node processes are also of interest for control applications in which the linearity assumption in (3) does not hold or a reasonable linear approximation for the IoT node process cannot be obtained.

APPENDIX A PROOF OF THEOREM 2

Consider two different systems, the first one with Markov transition probabilities $1 - q$ and q , and the second one

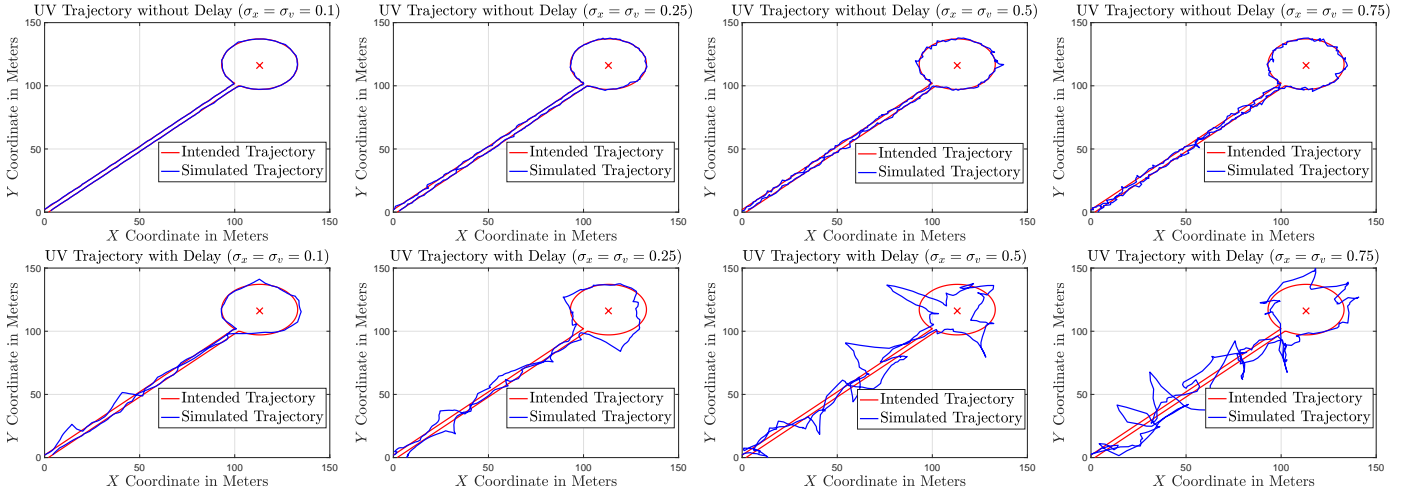


Fig. 5. Effect of environmental disturbances on the trajectory tracking performance of a UV fog controller. ($p = 0.75$, $\alpha = 0.1$ and $\rho = \frac{\sigma_x \sigma_v}{2}$ for all figures. The upper figures are for the case without delay, whereas the lower figures are for the case with delay $M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

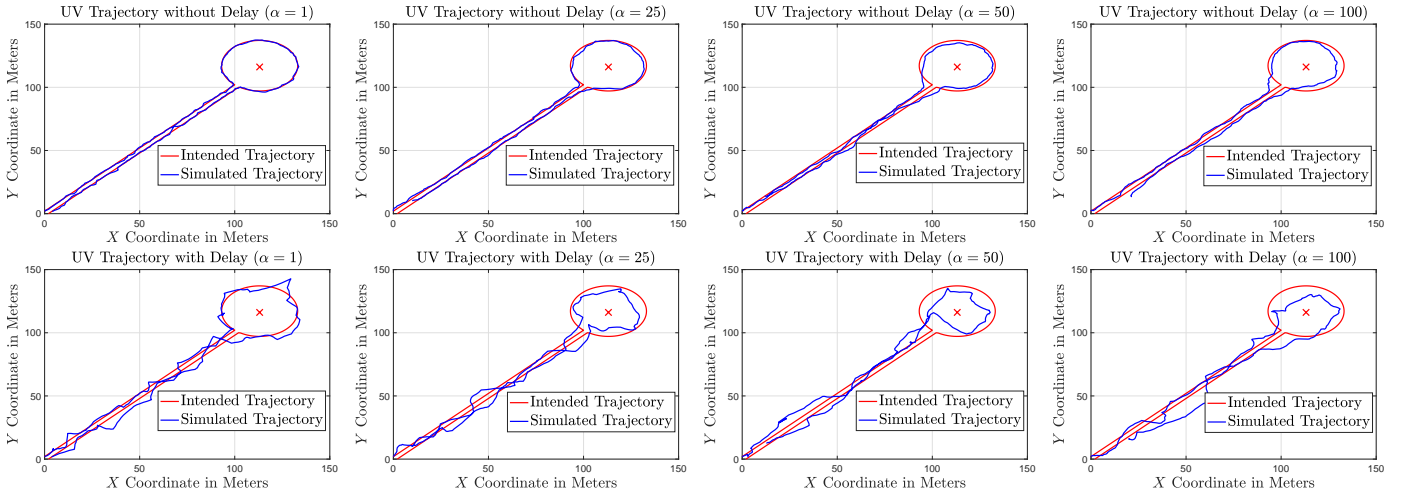


Fig. 6. Effect of the energy weight parameter α on the trajectory tracking performance of a UAV fog controller. ($p = 0.75$, $\sigma_x = 0.25$, $\sigma_v = 0.25$ and $\rho = \frac{\sigma_x \sigma_v}{2}$ for all figures. The upper figures are for the case without delay, whereas the lower figures are for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

with p and q . Consider the optimum control π^* achieving $J^{*(1-q,q)}(x_0, \tau_0)$ for the first system. We apply the same control to System 2, although being sub-optimum, to obtain the upper bound. Let $J_k^{*(1-q,q)}(x_k, \tau_k)$ and $J_k^{(p,q)}(x_k, \tau_k)$ be the cost-to-go values for systems 1 and 2 under π^* , starting at x_k and τ_k . It is easy to see that

$$J_k^{(p,q)}(x_k, 1) \leq J_k^{*(1-q,q)}(x_k, 1) \quad (19)$$

and

$$J_k^{(p,q)}(x_k, 0) \leq J_k^{*(1-q,q)}(x_k, 0) \quad (20)$$

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

$$\begin{aligned} J_k^{(p,q)}(x_k, 1) &= x_k^\top Q_k x_k + (u_k^*)^\top R_k u_k^* \\ &\quad + p E_{(x_k, 1)} \left[J_{k+1}^{(p,q)}(x_{k+1}, 1) | u_k^* \right] \\ &\quad + (1-p) E_{(x_k, 1)} \left[J_{k+1}^{(p,q)}(x_{k+1}, 0) | u_k^* \right], \end{aligned}$$

where u_k^* is the optimum control applied to System 1 at time k after observing x_k . Using the inductive argument and observing that $J_k^{*(1-q,q)}(x_k, 1) \leq J_k^{*(1-q,q)}(x_k, 0)$, we have

$$\begin{aligned} J_k^{(p,q)}(x_k, 1) &\leq x_k^\top Q_k x_k + (u_k^*)^\top R_k u_k^* \\ &\quad + q E_{(x_k, 1)} \left[J_{k+1}^{*(1-q,q)}(x_{k+1}, 0) | u_k^* \right] \\ &\quad + (1-q) E_{(x_k, 1)} \left[J_{k+1}^{*(1-q,q)}(x_{k+1}, 1) | u_k^* \right] \\ &= J_k^{*(1-q,q)}(x_k, 1). \end{aligned}$$

Repeating the same steps for $J_k^{(p,q)}(x_k, 0)$ shows that $J_k^{*(p,q)}(x_k, \tau_k) \leq J_k^{*(1-q,q)}(x_k, \tau_k)$ since $J_k^{*(p,q)}(x_k, \tau_k) \leq J_k^{(p,q)}(x_k, \tau_k)$, 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'12*, Aug. 2012.

- [2] M. Chiang and T. Zhang, "Fog and IoT: An overview of research opportunities," *IEEE Internet Things J.*, vol. 3, pp. 854–864, Dec. 2016.
- [3] P. Levine, "Return to the edge and the end of cloud computing," a16z.com/2016/12/16/the-end-of-cloud-computing/, Dec. 2016.
- [4] "AWS Greengrass," aws.amazon.com/greengrass. Accessed Jan. 2018.
- [5] "Azure IoT edge," github.com/Azure/iot-edge. Accessed Jan. 2018.
- [6] OpenFog Consortium, "OpenFog Reference Architecture." Available online: www.openfogconsortium.org/ra/, Feb. 2017.
- [7] ETSI Industry Specification Group, "Mobile edge computing (MEC): Framework and reference architecture." www.etsi.org/technologies-clusters/technologies/multi-access-edge-computing/. Accessed Jan. 2018.
- [8] H. Esen, M. Adachi, D. Bernardini, A. Bemporad, D. Rost, and J. Knodel, "Control as a service (CaaS): Cloud-based software architecture for automotive control applications," in *Proc. ACM SWEC'15*, Apr. 2015.
- [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 *IEEE AMC'16*, Apr. 2016.
- [10] A. E. Abdelaal, T. Hegazy, and M. Hefeeda, "Event-based control as a cloud service," in *Proc. 2017 American Control Conference*, (Seattle, USA), May 2017.
- [11] A. Vick, J. Guhl, and J. Kruger, "Model predictive control as a service concept and architecture for use in cloud-based robot control," in *Proc. IEEE MMAR'16*, (Miedzyzdroje, Poland), Aug. 2016.
- [12] M. Yannuzzi, F. van Lingen, A. Jain, O. L. Parellada, M. M. Flores, D. Carrera, J. L. Pérez, D. Montero, P. Chacin, A. Corsaro, and A. Olive, "A new era for cities with fog computing," *IEEE Internet Comput.*, vol. 21, pp. 54–67, Mar 2017.
- [13] M. A. A. Faruque and K. Vatanparvar, "Energy management-as-a-service over fog computing platform," *IEEE Internet Things J.*, vol. 3, pp. 161–169, Apr. 2016.
- [14] Y. Gao, W. Hu, K. Ha, B. Amos, P. Pillai, and M. Satyanarayanan, "Are cloudlets necessary?," Tech. Rep. CMU-CS-15-139, CMU School of Computer Science, Oct. 2015.
- [15] K. J. Aström and R. M. Murray, *Feedback systems: an introduction for scientists and engineers*. Princeton University Press, 2010.
- [16] D. Bertsekas, *Dynamic Programming and Optimal Control - Volume I*. Belmont, MA, USA: Athena Scientific, 1995.
- [17] "Amazon prime air," www.amazon.com/Amazon-Prime-Air/b?node=8037720011. Accessed Jan. 2018.
- [18] A. Khan, E. Yanmaz, and B. Rinner, "Information exchange and decision making in micro aerial vehicle networks for cooperative search," *IEEE Trans. Control of Network Systems*, vol. 2, pp. 335–347, Dec. 2015.
- [19] X. Wang, A. Chowdhery, and M. Chiang, "SkyEyes: Adaptive video streaming from UAVs," in *Proc. ACM HotWireless'16*, Sept. 2016.
- [20] E. Bregu, N. Casamassima, D. Cantoni, L. Mottola, and K. Whitehouse, "Reactive control of autonomous drones," in *Proc. ACM MobiSys'16*, Jun. 2016.
- [21] P. B. Sujit, S. Saripalli, and J. B. Sousa, "An evaluation of UAV path following algorithms," in *Proc. IEEE ECC'13*, Jul. 2013.
- [22] S. Kukreti, M. Kumar, and K. Cohen, "Genetically tuned LQR-based path following for UAVs under wind disturbance," in *Proc. IEEE ICUAS'16*, Jun. 2016.
- [23] L. Tong, Y. Li, and W. Gao, "A hierarchical edge cloud architecture for mobile computing," in *Proc. IEEE INFOCOM'16*, May 2016.
- [24] H. Tan, Z. Han, X.-Y. Li, and F. C. Lau, "Online job dispatching and scheduling in edge-clouds," in *Proc. IEEE INFOCOM'17*, May 2017.
- [25] Y. Xiao and M. Krunz, "QoE and power efficiency tradeoff for fog computing networks with fog node cooperation," in *Proc. IEEE INFOCOM'17*, May 2017.
- [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'12*, Mar. 2012.
- [27] V. Liberatore, "Integrated play-back, sensing, and networked control," in *Proc. IEEE INFOCOM'06*, Apr. 2006.
- [28] N. J. Ploplys, P. A. Kawka, and A. G. Alleyne, "Closed-loop control over wireless networks," *IEEE Control Syst. Mag.*, vol. 24, pp. 58–71, Jun. 2004.
- [29] K. Gatsis, A. Ribeiro, and G. J. Pappas, "Control with random access wireless sensors," in *IEEE CDC'15*, pp. 318–323, Dec. 2015.
- [30] K. Gatsis, M. Pajic, A. Ribeiro, and G. J. Pappas, "Opportunistic control over shared wireless channels," *IEEE Trans. Autom. Control*, vol. 60, pp. 3140–3155, Dec. 2015.
- [31] O. C. Imer, S. Yüksel, and T. Basar, "Optimal control of lti systems over unreliable communication links," *Automatica*, vol. 42, pp. 1429–1439, Sept. 2006.
- [32] V. Gupta, B. Hassibi, and R. M. Murray, "Optimal LQG control across packet-dropping links," *Systems & Control Letters*, vol. 56, pp. 439–446, Jun. 2007.
- [33] H. Esen, "Control as a service (caas) : Cloud-based software architecture for automotive control applications," April 2015.
- [34] "AWS EC2 instance types," aws.amazon.com/ec2/instance-types/. Accessed Jan. 2018.
- [35] "AWS Lambda," aws.amazon.com/lambda/. Accessed Jan. 2018.
- [36] E. A. Brewer, "Lessons from giant-scale services," *IEEE Internet Computing*, vol. 5, pp. 46–55, Jul 2001.
- [37] R. Margolies, M. Gorlatova, J. Sarik, G. Stanje, J. Zhu, B. Vignanam, M. Szczodrak, L. Carloni, P. Kinget, I. Kymissis, and G. Zussman, "Energy harvesting active networked tags (EnHANTs): Prototyping and experimentation," *ACM Transactions on Sensor Networking (ACM TOSN)*, vol. 11, Nov. 2015.
- [38] B. Lucia, V. Balaji, A. Colin, K. Maeng, and E. Ruppel, "Intermittent Computing: Challenges and Opportunities," in *SNAPL 2017* (B. S. Lerner, R. Bodík, and S. Krishnamurthi, eds.), vol. 71 of *Leibniz International Proceedings in Informatics (LIPIcs)*, (Dagstuhl, Germany), pp. 8:1–8:14, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2017.
- [39] "Google cloud functions," cloud.google.com/functions/. Accessed Jan. 2018.
- [40] "Azure functions," azure.microsoft.com/en-us/services/functions/. Accessed Jan. 2018.
- [41] "Bluemix OpenWhisk," www.ibm.com/cloud-computing/bluemix/openwhisk. Accessed Jan. 2018.
- [42] M. Fowler, "Serverless architectures," martinfowler.com/articles/serverless.html. Accessed Jan. 2018.
- [43] A. Ronacher, "Flask: Web development, one drop at a time," flask.pocoo.org/. Accessed Jan. 2018.
- [44] "Green Unicorn: Python WSGI HTTP Server for UNIX," gunicorn.org. Accessed Jan. 2018.
- [45] S. H. Strogatz, *Nonlinear Dynamics and Chaos with Applications to Physics, Biology, Chemistry, and Engineering*. Cambridge, MA, USA: Perseus Books Publishing, 1995.
- [46] D. N. C. Tse and P. Viswanath, *Fundamentals of Wireless Communications*. New York, NY, USA: Cambridge University Press, 2005.
- [47] S. Yüksel and T. Basar, *Stochastic Networked Control Systems: Stabilization and Optimization under Information Constraints*. New York, NY, USA: Birkhauser, 2013.
- [48] S. Tatikonda, A. Sahai, and S. Mitter, "Stochastic linear control over a communication channel," *IEEE Trans. Autom. Control*, vol. 49, pp. 1549–1561, Sep. 2004.
- [49] Y. B. Shalom and E. Tse, "Dual effect, certainty equivalence, and separation in stochastic control," *IEEE Trans. Autom. Control*, vol. 19, pp. 494–500, Oct. 1974.
- [50] R. Durrett, *Probability: Theory and Examples*. Belmont, CA, USA: Duxbury Press, 2nd ed., 1996.
- [51] DJI, "DJI Matrice 100 Quadcopter," www.dji.com/matrice100/info. Accessed Jan. 2018.