Dynamic Service Migration and Workload Scheduling in Edge-Clouds

Rahul Urgaonkar*, Shiqiang Wang†, Ting He*, Murtaza Zafer‡, Kevin Chan§, and Kin K. Leung†

Abstract

Edge-clouds provide a promising new approach to significantly reduce network operational costs by moving computation closer to the edge. A key challenge in such systems is to decide where and when services should be migrated in response to user mobility and demand variation. The objective is to optimize operational costs while providing rigorous performance guarantees. In this paper, we model this as a sequential decision making Markov Decision Problem (MDP). However, departing from traditional solution methods (such as dynamic programming) that require extensive statistical knowledge and are computationally prohibitive, we develop a novel alternate methodology. First, we establish an interesting decoupling property of the MDP that reduces it to two independent MDPs on disjoint state spaces. Then, using the technique of Lyapunov optimization over renewals, we design an online control algorithm for the decoupled problem that is provably cost-optimal. This algorithm does not require any statistical knowledge of the system parameters and can be implemented efficiently. We validate the performance of our algorithm using extensive trace-driven simulations. Our overall approach is general and can be applied to other MDPs that possess a similar decoupling property.

I. Introduction

The increasing popularity of mobile applications (such as social networking, photo sharing, etc.) running on handheld devices is putting a significant burden on the capacity of cellular and backhaul networks. These applications are generally comprised of a front-end component running on the handheld and a back-end component (that performs data processing and computation) that typically runs on the cloud. While this architecture enables applications to take advantage of the on-demand feature of cloud computing, it also introduces new challenges in the form of increased network overhead and latency. A promising approach to address these challenges is to move such computation closer to the network edge. Here, it is envisioned that entities (such as basestations in a cellular network) closer to the network edge would host smaller-sized cloud-like infrastructure distributed across the network. This idea has been variously termed as Cloudlets [1], Fog Computing [2], Edge Computing [3], and Follow Me Cloud [4], to name a few. The trend towards edge-clouds is expected to accelerate as more users perform a majority of their computations on handhelds and as newer mobile applications get adopted.

One of the key design issues in edge-clouds is service migration: Should a service currently running in one of the edge-clouds be migrated as the user locations change, and if yes, where? This question stems from the basic tradeoff

*R. Urgaonkar and T. He are with the IBM T. J. Watson Research Center, Yorktown Heights, NY, USA. Emails: {rurgaon, the}@us.ibm.com. †S. Wang and K. K. Leung are with the Department of Electrical and Electronic Engineering, Imperial College London, UK. Emails: {shiqiang.wang11, kin.leung}@imperial.ac.uk. ‡M. Zafer is with Nyansa Inc., Palo Alto, CA, USA. Email: murtaza.zafer.us@ieee.org. §K. Chan is with the US Army Research Laboratory, Adelphi, MD, USA. Email: kevin.s.chan.civ@mail.mil.

The final version of this paper has been published at Performance Evaluation, vol. 91, pp. 205-228, Sept. 2015, http://dx.doi.org/10.1016/j.peva.2015.06.013

between the cost of service migration vs. the reduction in network overhead and latency for users that can be achieved after migration. While conceptually simple, it is challenging to make this decision in an optimal manner because of the uncertainty in user mobility and request patterns. Because edge-clouds are distributed at the edge of the network, their performance is closely related to user dynamics. These decisions get even more complicated when the number of users and applications is large and there is heterogeneity across edge-clouds. Note that the service migration decisions affect workload scheduling as well (and vice versa), so that in principle these decisions must be made jointly.

The overall problem of dynamic service migration and workload scheduling to optimize system cost while providing end-user performance guarantees can be formulated as a sequential decision making problem in the framework of MDPs [5], [6]. This approach, although very general, suffers from several drawbacks. First, it requires extensive knowledge of the statistics of the user mobility and request arrival processes that can be impractical to obtain in a dynamic network. Second, even when this is known, the resulting problem can be computationally challenging to solve. Finally, any change in the statistics would make the previous solution suboptimal and require recomputing the optimal solution.

In this paper, we present a new methodology that overcomes these drawbacks. Our approach is inspired by the technique of Lyapunov optimization [7], [8] which is a general framework for designing optimal control algorithms for *non-MDP* problems *without requiring any knowledge* of the transition probabilities. Specifically, these are problems where the cost functions and control decisions are functionals of states that evolve independently of the control actions. However, as we will show later, this condition does not hold for the joint service migration and workload scheduling problem studied in this paper. A key contribution of this work is to develop a methodology that enables us to still apply the Lyapunov optimization technique to this MDP while preserving its attractive features.

II. Related Work

The general problem of resource allocation and workload scheduling in cloud computing systems using the framework of stochastic optimization has been considered in several recent works. Specifically, [9] considers a stochastic model for a cloud computing cluster, where requests for virtual machines (VMs) arrive according to a stochastic process. Each VM request is specified in terms of a vector of resources (such as CPU, memory and storage space) and its duration and must be placed on physical machines (PMs) subject to a set of vector packing constraints. Reference [9] defines the notion of the capacity region of the cloud system and shows that the MaxWeight algorithm is throughput optimal. Virtual machine placement utilizing shadow routing is studied in [10] and [11], where virtual queues are introduced to capture more complicated packing constraints. Reference [12] considers a joint VM placement and route selection problem where in addition to packing constraints, the traffic load between the VMs is also considered. All these works consider a traditional cloud model where the issue of user mobility and resulting dynamics is not considered. As discussed before, this issue becomes crucial in edge-clouds and introduces the need for dynamic service migration that incurs reconfiguration costs. The presence of these costs fundamentally changes the underlying resource allocation problem from a non-MDP to an MDP for which the techniques used in these works are no longer optimal.

The impact of reconfiguration or switching cost has been considered in some works recently. Specifically, [13] considers the problem of dynamic "right-sizing" of data centers where the servers are turned ON/OFF in response to the time-varying workloads. However, such switching incurs cost in terms of the delay associated with switching as well the impact on server life. Reference [13] proposes an online algorithm that is shown to have a 3-competitive ratio while explicitly considering

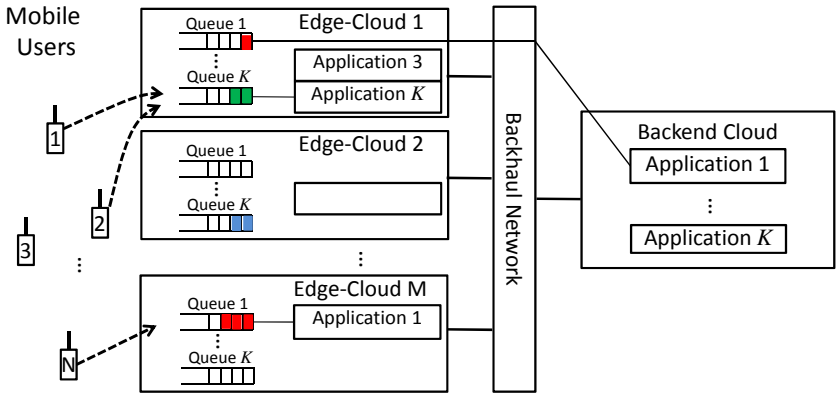


Fig. 1. Illustration of our edge-cloud model showing the collection of edge-clouds, back-end cloud, and mobile users.

switching costs. A similar problem involving geographic load balancing is considered in [14] using a receding horizon control framework. Reference [15] focuses on a wireless scheduling problem with reconfiguration delay while [16] studies the reconfiguration problem from a queuing theory perspective and derives analytical expressions for the performance. All the approaches in [14]–[16] assume knowledge of the statistics of the underlying system while [13] considers a single data center with homogeneous servers. Our work differs from all these because we explicitly consider the reconfiguration costs associated with service migrations while treating a very general model for a distributed edge-cloud system.

The methodology used in this paper is inspired by the framework of Lyapunov optimization over renewals proposed in [17]. This framework extends the Lyapunov optimization approach of [7], [8] to treat constrained MDPs. The basic idea involves converting a constrained MDP into a sequence of *unconstrained stochastic shortest path problems* (SSPs) [5], [6] that are solved over consecutive renewal frames. However, solving the resulting SSPs typically still requires knowledge of the underlying probability distributions and can be computationally prohibitive for large state spaces.

In this work, we also make use of the framework of Lyapunov optimization over renewals. However, instead of directly applying the technique of [17], we first establish a novel decoupling property of our MDP which shows that it can be decoupled into two independent MDPs that evolve on disjoint state spaces. This crucial property enables us to apply the framework of [17] in such a way that the resulting algorithms are simple *deterministic* optimization problems (rather than stochastic shortest path problems) that can be solved efficiently *without any knowledge of the underlying probability distributions*. For example, one of the components of our overall algorithm involves solving a deterministic shortest path problem instead of an SSP every renewal frame. As such, the resulting solution is markedly different from classical dynamic programming based approaches and does not suffer from the associated "curse of dimensionality" or convergence issues.

III. Problem Formulation

We consider an edge-cloud system comprised of M distributed edge-clouds and one back-end cloud that together host K applications (see Fig. 1). The system also consists of N users that generate application requests over time. The collection of edge and back-end clouds supports these applications by providing the computational resources needed to serve user requests. The users are assumed to be mobile while the edge and back-end clouds are static. We assume a time-slotted model and use the notion of "service" and "application" interchangeably in this paper.

A. System Model

Mobility Model: Let $L_n(t)$ denote the location of user n in slot t. The collection of all user locations in slot t is denoted by vector $\boldsymbol{l}(t)$. We assume that $\boldsymbol{l}(t)$ takes values from a finite (but potentially arbitrarily large) set $\mathcal{L}$. Further, $\boldsymbol{l}(t)$ is assumed to evolve according to an ergodic discrete time Markov chain (DTMC) over the states in $\mathcal{L}$ with transition probabilities denoted by $p_{\boldsymbol{ll'}}$ for all $\boldsymbol{l}, \boldsymbol{l'} \in \mathcal{L}$.

Application Request Model: Denote the number of requests for application k generated by user n in slot t by $A_{kn}(t)$ and the collection $\{A_{kn}(t)\}$ for all k, n by vector $\boldsymbol{a}(t)$. Similar to $\boldsymbol{l}(t)$, we assume that $\boldsymbol{a}(t)$ takes values from a finite (but potentially arbitrarily large) set $\mathcal{A}$. We further assume that for all k, n there exist finite constants $A_{kn}^{\max}$ such that $A_{kn}(t) \leq A_{kn}^{\max}$ for all t. The process $\boldsymbol{a}(t)$ is also assumed to evolve according to an ergodic DTMC over the states in $\mathcal{A}$ with transition probabilities $q_{\boldsymbol{aa'}}$ for all $\boldsymbol{a}, \boldsymbol{a'} \in \mathcal{A}$. All application requests generated at each user are routed to a selected subset of the edge-clouds for servicing. These routing decisions incur transmission costs and are subject to certain constraints as discussed below.

User-to-Edge-Cloud Request Routing: Let $r_{knm}(t)$ denote the number of application k requests from user n that are routed to edge-cloud m in slot t and let $\boldsymbol{r}(t)$ denote the collection $\{r_{knm}(t)\}$ for all k, n, m. Routing of these requests incurs a transmission cost of $r_{knm}(t)c_{knm}(t)$ where $c_{knm}(t)$ is the unit transmission cost that can depend on the current location of the user $L_n(t)$, the application index k, as well as the edge-cloud index m. More generally, it could also depend on other "uncontrollable" factors such as background backhaul traffic, wireless fading, etc., but we do not consider these for simplicity. Denote the sum total transmission cost incurred in slot t by $C(t)$, i.e., $C(t) = \sum_{knm} r_{knm}(t)c_{knm}(t)$. For each (k, m), we denote by $R_{km}(t)$ the total number of application k requests received by edge-cloud m in slot t, i.e., $R_{km}(t) = \sum_{n=1}^{N} r_{knm}(t)$. We assume that the maximum number of requests for an application k that can be routed to edge-cloud m in any slot is upper bounded by $R_{km}^{\max}$. Given these assumptions, the routing decisions $\boldsymbol{r}(t)$ are subject to the following constraints

$$A_{kn}(t) = \sum_{m=1}^{M} r_{knm}(t) \ \ \forall k, n \tag{1}$$

$$0 \leq \sum_{n=1}^{N} r_{knm}(t) \leq R_{km}^{\max} \ \ \forall k, m \tag{2}$$

where (1) captures the assumption that no request buffering happens at the users. In addition to (1) and (2), there can be other location-based constraints that limit the set of edge-clouds where requests from user n can be routed given its location $L_n(t)$. Given $\boldsymbol{l}(t) = l, \boldsymbol{a}(t) = a$, denote the feasible request routing set by $\mathcal{R}(l, a)$. We assume that $\mathcal{R}(l, a) \neq \emptyset$ for all $l \in \mathcal{L}, a \in \mathcal{A}$.

Application Configuration of Edge-Clouds: For all k, m, define application placement variables $H_{km}(t)$ as

$$H_{km}(t) = \begin{cases} 1 & \text{if edge-cloud } m \text{ hosts application } k \text{ in slot } t, \\ 0 & \text{else.} \end{cases} \tag{3}$$

The collection $\{H_{km}(t)\}$ is denoted by the vector $\boldsymbol{h}(t)$. This defines the *application configuration* of the edge-clouds in slot t and determines the *local service rates* $\{\mu_{km}(t)\}$ offered by them in that slot. An application's requests can only be serviced by an edge-cloud if it hosts this application in that slot. Thus, $\mu_{km}(t) = 0$ if $H_{km}(t) = 0$. When $H_{km}(t) = 1$,

then $\mu_{km}(t)$ is assumed to be a general non-negative function $\varphi_{km}(\cdot)$ of the vector $\boldsymbol{h}(t)$, i.e., $\mu_{km}(t) = \varphi_{km}(\boldsymbol{h}(t))$. This results in a very general model that can capture correlations between the service rates of co-located applications. A special case is where $\mu_{km}(t)$ depends only on $H_{km}(t)$. For simplicity, we assume that $\mu_{km}(t)$ is a deterministic function of $\boldsymbol{h}(t)$ and use $\mu_{km}(t)$ to mean $\varphi_{km}(\boldsymbol{h}(t))$. Further, we assume that there exist finite constants $\mu_{km}^{\max}$ such that $\mu_{km}(t) \leq \mu_{km}^{\max}$ for all t.

An edge-cloud is typically resource constrained and may not be able to host all applications. In general, hosting an application involves creating a set of virtual machines (VMs) or execution containers (e.g., Docker) and assigning them a vector of computing resources (such as CPU, memory, storage, etc.) from the physical machines (PMs) in the edge-cloud. We say that an application configuration $\boldsymbol{h}(t)$ is feasible if there exists a VM-to-PM mapping that does not violate any resource constraints. The set of all feasible application configurations is denoted by $\mathcal{H}$ and is assumed to be finite. We also assume that there is a system-wide controller that can observe the state of the system and change the application configuration over time by using techniques such as VM migration and replication. This enables the controller to adapt in response to the system dynamics induced by user mobility as well as demand variations. However, such reconfiguration incurs a cost that is a function of the degree of reconfiguration. Given any two configurations $\boldsymbol{a}, \boldsymbol{b} \in \mathcal{H}$, the cost of switching from $\boldsymbol{a}$ to $\boldsymbol{b}$ is denoted by $W_{\boldsymbol{ab}}$. For simplicity, we assume that it is possible to switch between any two configurations $\boldsymbol{a}, \boldsymbol{b} \in \mathcal{H}$ and that $W_{\boldsymbol{ab}}$ is upper bounded by a finite constant $W_{\max}$. We further assume, without loss of generality, that

$$W_{\boldsymbol{ab}} \leq W_{\boldsymbol{ac}} + W_{\boldsymbol{cb}} \; \forall \boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c} \in \mathcal{H}. \tag{4}$$

The last assumption is valid if $W_{\boldsymbol{ab}}$ is the minimum cost required to switch from $\boldsymbol{a}$ to $\boldsymbol{b}$, This is because if $W_{\boldsymbol{ab}} > W_{\boldsymbol{ac}} + W_{\boldsymbol{cb}}$, then we could carry out the reconfiguration from $\boldsymbol{a}$ to $\boldsymbol{b}$ by switching from $\boldsymbol{a}$ to $\boldsymbol{c}$ and then to $\boldsymbol{b}$, achieving lower cost. Denote the switching cost incurred in slot t by $W(t)$. For simplicity, we assume that switching incurs no delay while noting that our model can be extended to consider such delays (for example, by setting the local service rates to zero during those slots when a reconfiguration is underway).

Request Queues at the Edge-Clouds: As illustrated in Fig. 1, every edge-cloud m maintains a request queue $U_{km}(t)$ per application k that buffers application k requests from all users that are routed to edge-cloud m. Requests in $U_{km}(t)$ can get serviced locally in a slot by edge-cloud m if it hosts application k in that slot. In addition, buffered requests in $U_{km}(t)$ can also be routed to the back-end cloud which is assumed to host all applications at all times. However, this incurs additional back-end transmission cost as discussed later. The queueing dynamics for $U_{km}(t)$ is given by

$$U_{km}(t+1) = \max[U_{km}(t) - \mu_{km}(t) - \upsilon_{km}(t) + R_{km}(t), 0] \tag{5}$$

where $\upsilon_{km}(t)$ denotes the number of requests from $U_{km}(t)$ that are transmitted to the back-end cloud in slot t and $\mu_{km}(t)$ is the local service rate. It is assumed that the requests in $U_{km}(t)$ are serviced in a FIFO manner. The collection of all queue backlogs $\{U_{km}(t)\}$ is denoted by the vector $\boldsymbol{U}(t)$. From (5), note that requests generated in a slot can get service in the same slot. It should also be noted that requests can be routed to $U_{km}(t)$ in a slot even if $H_{km}(t) = 0$.

Edge-Cloud to Back-End Cloud Request Routing: The back-end cloud is assumed to host all applications at all times. However, transmitting requests to the back-end may incur very high costs and therefore it is desirable to maximize the fraction of requests that can be serviced locally by the edge-clouds. Let $\upsilon_{km}(t)$ denote the number of number of requests

from $U_{km}(t)$ that are transmitted to the back-end cloud in slot t and let $\boldsymbol{\upsilon}(t)$ denote the collection $\{\upsilon_{km}(t)\}$ for all k, m. Routing of $\upsilon_{km}(t)$ incurs a transmission cost of $\upsilon_{km}(t)e_{km}(t)$ where $e_{km}(t)$ is the unit back-end transmission cost that can depend on the application index as well as the edge-cloud index. Similar to request routing costs $c_{knm}(t)$, $e_{km}(t)$ can also depend on other "uncontrollable" factors (such as background backhaul traffic), but we only consider the average impact of these for simplicity. Since both the edge-clouds and the back-end cloud are static, we have $e_{km}(t) = e_{km}$ for all t. We assume that there exist finite constants $\upsilon_{km}^{\max}$ such that $\upsilon_{km}(t) \leq \upsilon_{km}^{\max}$ for all t. Further, $R_{km}^{\max} \leq \upsilon_{km}^{\max}$ which models the baseline scenario where all requests are serviced only by the back-end cloud. Denote the set of all $\boldsymbol{\upsilon}(t)$ that satisfy these constraints by $\mathcal{V}$ and the sum total back-end transmission cost incurred in slot t by $E(t)$, i.e., $E(t) = \sum_{km} \upsilon_{km}(t)e_{km}$. We assume that the back-end cloud has sufficient processing capacity such that it can service all requests in $\boldsymbol{\upsilon}(t)$ with negligible delay. Thus, queueing in the back-end becomes trivial and is ignored. It should be noted that in our model any user request that is eventually serviced by the back-end cloud is transmitted first to an edge-cloud.

Performance Objective: Given this model, our goal is to design a control algorithm for making request routing decisions at the users and edge-clouds as well as application reconfiguration decisions across the edge-clouds so that the time-average overall transmission and reconfiguration costs are minimized while serving all requests with finite delay. Specifically, we assume that the time-average delay for the requests in each queue $U_{km}(t)$ should not exceed d_{avg}, where d_{avg} is a finite constant. This can be formulated as a constrained Markov Decision Problem (MDP) [5], [6] as shown in Section III-B.

Timing of Events in a Slot: We assume the following sequence of events in a slot. At the start of slot t, the controller observes the queue backlogs $\boldsymbol{U}(t)$, new arrivals $\boldsymbol{a}(t)$, user locations $\boldsymbol{l}(t)$, and the last configuration $\boldsymbol{h}(t-1)$. Then it makes a reconfiguration decision that transitions the configuration state to $\boldsymbol{h}(t)$ and this determines the local service rates offered in slot t. Then the controller makes user to edge-cloud and edge-cloud to back-end cloud routing decisions. The queue backlogs $\boldsymbol{U}(t+1)$ at the start of the next slot evolve according to (5).

B. MDP Formulation

The control problem described in Section III-A can be formulated as a constrained MDP over the joint state space $(\boldsymbol{l}(t), \boldsymbol{a}(t), \boldsymbol{h}(t), \boldsymbol{U}(t))$. It is well-known that if this problem is feasible, then an optimal control policy for this MDP can be obtained by searching over the class of stationary, randomized control algorithms that take control actions purely as a function of the system states [5], [6]. Specifically, consider a control algorithm that operates as follows.

First, given last state $\phi' = (\boldsymbol{l}(t-1) = l', \boldsymbol{a}(t-1) = a', \boldsymbol{h}(t-1) = h', \boldsymbol{U}(t-1) = u')$ and current location, arrival, and queue backlog states $\boldsymbol{l}(t) = l, \boldsymbol{a}(t) = a$ and $\boldsymbol{U}(t) = u$, the control algorithm chooses current configuration $\boldsymbol{h}(t) = h$ with probability $z_{\phi'\phi}$, thereby transitioning the state to $\phi = (\boldsymbol{l}(t) = l, \boldsymbol{a}(t) = a, \boldsymbol{h}(t) = h, \boldsymbol{U}(t) = u)$. Note that the routing decisions taken in the last slot together with its configuration h' determine $\boldsymbol{U}(t)$ through the queueing equations in (5). Denote the resulting transition probability from $\boldsymbol{U}(t-1) = u'$ to $\boldsymbol{U}(t) = u$ by $s_{u'u}^{\phi'}$. Then the total expected reconfiguration cost incurred when transitioning from state ϕ' is given by

$$W_{\phi'} = \sum_{\phi} p_{l'l} q_{a'a} s_{u'u}^{\phi'} z_{\phi'\phi} W_{h'h}. \tag{6}$$

Next, given current state $\phi = (\boldsymbol{l}(t) = l, \boldsymbol{a}(t) = a, \boldsymbol{h}(t) = h, \boldsymbol{U}(t) = u)$, the control algorithm chooses routing vector

$\boldsymbol{r}(t) = r$ with probability $x_\phi(r)$ subject to $r \in \mathcal{R}(l, a)$. This incurs a total expected transmission cost given by

$$C_\phi = \sum_{r \in \mathcal{R}(l,a)} x_\phi(r) \sum_{knm} r_{knm} c_{knm} \tag{7}$$

Finally, given current state $\phi = (\boldsymbol{l}(t) = l, \boldsymbol{a}(t) = a, \boldsymbol{h}(t) = h, \boldsymbol{U}(t) = u)$, it chooses back-end routing vector $\boldsymbol{\upsilon}(t) = \upsilon$ with probability $y_\phi(\upsilon)$ subject to $\upsilon \in \mathcal{V}$ and this incurs a total expected back-end transmission cost given by

$$E_\phi = \sum_{\upsilon \in \mathcal{V}} y_\phi(\upsilon) \sum_{km} \upsilon_{km} e_{km} \tag{8}$$

Let us denote the steady state probability of being in state ϕ under this policy by π_ϕ. Then, the overall time-average expected transmission plus reconfiguration cost is given by

$$\overline{C} + \overline{E} + \overline{W} = \sum_\phi \pi_\phi C_\phi + \sum_\phi \pi_\phi E_\phi + \sum_\phi \pi_\phi W_\phi \tag{9}$$

Let $\overline{U}_{km}$ and $\overline{R}_{km}$ denote the time-average expected values of $U_{km}(t)$ and $R_{km}(t)$ under this control algorithm. By Little's Theorem, we have that the average delay $\overline{D}_{km}$ for the requests in queue $U_{km}(t)$ satisfies $\overline{U}_{km} = \overline{R}_{km}\overline{D}_{km}$. In order to meet the average delay constraint, we need that $\overline{D}_{km} = \frac{\overline{U}_{km}}{\overline{R}_{km}} \leq d_{\text{avg}} \forall k, m$.

The constrained MDP optimization searches for an optimal policy that minimizes $\overline{C} + \overline{E} + \overline{W}$ subject to meeting the average delay constraint. Assuming that the problem is feasible, let c^*, e^*, and w^* denote the optimal time-average user-to-edge-cloud transmission cost, edge-cloud-to-back-end cloud transmission cost, and total reconfiguration cost respectively. Solving this optimization is extremely challenging and quickly becomes intractable due to the complexity of the state space under traditional solution methods (such as value iteration [5], [6]). Further, these techniques require knowledge of the transition probabilities of the user mobility and request arrival processes that may not be known a-priori. In the following, we develop an alternate methodology for tackling this problem that overcomes these challenges. Specifically, we take the following approach.

1) We first relax this MDP by replacing the time-average delay constraints by queue stability constraints. This results in an MDP whose state space involves only $(\boldsymbol{l}(t), \boldsymbol{a}(t), \boldsymbol{h}(t))$.
2) For this relaxed MDP, we prove a novel decoupling property which shows that it can be *decoupled* into two independent MDPs that evolve on disjoint state spaces.
3) This decoupling property enables us to develop an online control algorithm that does not require any knowledge of the probability distributions, yet can achieve (arbitrarily) close to optimal cost *while providing worst-case delay guarantees*.

Before proceeding, we make the following assumption about the above (non-relaxed) MDP. Let R^*_{km}, μ^*_{km} and υ^*_{km} respectively denote the time-average request arrival rate, local service rate, and back-end routing rate for queue $U_{km}(t)$ under the optimal control policy. For all k, m for which $R^*_{km} > 0$, define $\epsilon_{km} = \mu^*_{km} + \upsilon^*_{km} - R^*_{km}$ and let $\epsilon = \min_{km} \epsilon_{km}$. Then we assume that ϵ is strictly positive, i.e., $\epsilon > 0$. Note that, in general, in a queue with stochastic arrivals and service rates, if the average arrival rate is not smaller than the service rate, then the average delay becomes unbounded. Therefore this assumption is not very restrictive.

IV. MDP Relaxation and Decoupling

Consider a relaxation of the original MDP discussed in Section III-B where we replace the average delay constraints by the following *queue stability* constraints $\forall k, m$.

$$\overline{\mu}_{km} + \overline{v}_{km} - \overline{R}_{km} \geq \epsilon \text{ if } \overline{R}_{km} > 0 \tag{10}$$

where $\overline{R}_{km}, \overline{\mu}_{km}$ and $\overline{v}_{km}$ respectively denote the time-average expected arrival rate, local service rate and back-end routing rate under any control algorithm. It can be shown that meeting these constraints ensures that all queues are rate stable [8]. Further, we add the constraints that $\overline{C} = c^*, \overline{E} = e^*$, and $\overline{W} = w^*$. That is, we enforce the time-average transmission and switching costs under the relaxed problem to match those under the optimal solution to the original MDP. It is clear that this problem is a relaxation of the original MDP since the solution to the original MDP is feasible for this problem. However, a solution to the relaxed problem will not necessarily satisfy the average delay constraints. An optimal stationary, randomized control algorithm for the relaxed problem can be defined similarly to the original MDP and is described in Appendix A. The motivation for considering this relaxation is that, unlike the original MDP, it suffices to consider the reduced state space defined by $(\boldsymbol{l}(t), \boldsymbol{a}(t), \boldsymbol{h}(t))$ for this problem. This follows by noting that none of the constraints involve the queue backlogs. Further, the relaxed problem has an interesting decoupling property (discussed next) that can be leveraged to design an online control algorithm that can achieve close to optimal cost while providing explicit worst-case delay guarantees (as shown in Section V).

A. Decoupling the Relaxed MDP

We now show that a decoupled control algorithm is optimal for the relaxed MDP defined above. Specifically, under this decoupled algorithm, the control decisions for user request routing are taken purely as a function of $(\boldsymbol{l}(t), \boldsymbol{a}(t))$, those for application reconfiguration are taken purely as a function of $\boldsymbol{h}(t)$, and the back-end routing decisions are taken in i.i.d. manner every slot, independent of all states. As a result, under this algorithm, the states $(\boldsymbol{l}(t), \boldsymbol{a}(t))$ and $\boldsymbol{h}(t)$ become decoupled and evolve independently of each other. Note that, in general, when searching for the optimal policy for the relaxed MDP, one must consider the class of algorithms where the control decisions are taken as a function of the joint state $(\boldsymbol{l}(t), \boldsymbol{a}(t), \boldsymbol{h}(t))$. Under such algorithms, the states $(\boldsymbol{l}(t), \boldsymbol{a}(t))$ and $\boldsymbol{h}(t)$ would be coupled and their evolution would not be independent. Thus, it is noteworthy that such a decoupled policy exists.

We next specify the decoupled control algorithm and show that it achieves the same time-average cost as the relaxed MDP (and hence the original MDP). The decoupled algorithm is defined in terms of the control decision probabilities (and resulting steady-state values) of the optimal solution to the relaxed MDP. We use the superscript "xdp" to indicate the control actions and resulting steady state probabilities of the optimal solution to the relaxed MDP while the superscript "dec" is used for the decoupled control algorithm. We use the shorthand notation $\phi = (\boldsymbol{l}(t) = l, \boldsymbol{a}(t) = a, \boldsymbol{h}(t) = h)$ and $\phi' = (\boldsymbol{l}(t-1) = l', \boldsymbol{a}(t-1) = a', \boldsymbol{h}(t-1) = h')$. Let π_{ϕ}^{xdp} denote the steady-state probability of state ϕ under the optimal solution to the relaxed MDP and $\sum_{la} \pi_{\phi}^{\text{xdp}}$ sums this over all states ϕ for a given configuration h. Define $\mathcal{H}'$ as the set of all configuration states h for which $\sum_{la} \pi_{\phi}^{\text{xdp}} > 0$. The decoupled algorithm has the following components:

Reconfiguration Policy: The reconfiguration policy is defined by probabilities $\theta_{h'h}^{\text{dec}}$ which denotes the probability of

switching to configuration h given that the configuration in the last slot was h'. These probabilities are given by

$$\theta^{\text{dec}}_{h'h} = \begin{cases} \frac{\sum_{l'a'} \pi^{\text{xdp}}_{\phi'} \left(\sum_{la} p_{l'l} q_{a'a} z^{\text{xdp}}_{\phi'\phi} \right)}{\sum_{l'a'} \pi^{\text{xdp}}_{\phi'}} & \text{if } h, h' \in \mathcal{H}', \\ 0 & \text{else.} \end{cases} \tag{11}$$

Routing Policy: Given $\boldsymbol{l}(t) = l, \boldsymbol{a}(t) = a$, choose a routing vector $r \in \mathcal{R}(l, a)$ with probability $\zeta^{\text{dec}}_{la}(r)$ given by

$$\zeta^{\text{dec}}_{la}(r) = \begin{cases} \frac{\sum_h \pi^{\text{xdp}}_{\phi} x^{\text{xdp}}_{\phi}(r)}{\sum_h \pi^{\text{xdp}}_{\phi}} & \text{if } \sum_h \pi^{\text{xdp}}_{\phi} > 0, \\ 0 & \text{else.} \end{cases} \tag{12}$$

where $\sum_h \pi^{\text{xdp}}_{\phi}$ sums π^{xdp}_{ϕ} over all states ϕ for which $\boldsymbol{l}(t) = l$ and $\boldsymbol{a}(t) = a$.

Back-End Routing Policy: In each slot t, choose a back-end routing vector $\upsilon \in \mathcal{V}$ with probability $\vartheta^{\text{dec}}(\upsilon)$ given by

$$\vartheta^{\text{dec}}(\upsilon) = \sum_{\phi} \pi^{\text{xdp}}_{\phi} y^{\text{xdp}}_{\phi}(\upsilon) \tag{13}$$

Let us denote the time-average arrival and service rates for queue $U_{km}(t)$ under the decoupled algorithm by $R^{\text{dec}}_{km}, \mu^{\text{dec}}_{km}$ and $\upsilon^{\text{dec}}_{km}$ respectively. Then we have the following.

Theorem 1: For the decoupled control algorithm defined by (11), (12), and (13), the following hold:

1) The time-average reconfiguration cost is equal to w^*.
2) The time-average transmission cost is equal to c^*.
3) The time-average back-end routing cost is equal to e^*.
4) For each queue $U_{km}(t)$, the time-average arrival and service rates $R^{\text{dec}}_{km}, \mu^{\text{dec}}_{km}$ and $\upsilon^{\text{dec}}_{km}$ are equal to those under the relaxed MDP and satisfy (10), i.e., $\mu^{\text{dec}}_{km} + \upsilon^{\text{dec}}_{km} - R^{\text{dec}}_{km} \geq \epsilon$ if $R^{\text{dec}}_{km} > 0$.

Proof: See Appendix B. ■

We emphasize that the time-average arrival and service rates $R^{\text{dec}}_{km}, \mu^{\text{dec}}_{km}$, and $\upsilon^{\text{dec}}_{km}$ need not be equal to the corresponding values for the original MDP, i.e., R^*_{km}, μ^*_{km} and υ^*_{km}. Theorem 1 can be intuitively explained by noting that the probability $\theta^{\text{dec}}_{h'h}$ is chosen to be equal to the fraction of time that the relaxed MDP chooses to switch to configuration h' given that the last configuration was h, in steady state. Similarly, $\zeta^{\text{dec}}_{la}(r)$ is chosen to be equal to the fraction of time the relaxed MDP chooses routing vector $r \in \mathcal{R}(l, a)$ given that the current user location and request arrival states are (l, a), in steady state, and the same applies to the back-end routing decisions. Thus, it can be seen that the decoupled control algorithm tries to match the time-average costs of the relaxed policy while meeting the queue stability constraints. Note that under the decoupled control algorithm, the reconfiguration and local servicing decisions are a function only of the configuration state h while the routing is only a function of the user location and arrival states (l, a). It should also be noted that in our model, the latter states (l, a) evolve on their own, independent of the control actions of this algorithm. On the other hand, the evolution of the configuration state is completely governed by this control algorithm. Finally, we note that the decoupled control algorithm is expressed in terms of the steady state probabilities and control actions of the optimal solution of the relaxed MDP which is itself hard to calculate and requires knowledge of the statistics of the mobility or arrival processes that may not be available. *However, our objective is not to calculate this control algorithm explicitly*. Rather, we will use its existence to obtain an alternate online control algorithm that will track the performance of this control algorithm. The online

algorithm does not require any knowledge of the statistics of the mobility or arrival processes and can be implemented efficiently. Further, the online algorithm stabilizes all queues and provides worst-case delay bounds for all requests. Recall that $\mathcal{H}'$ is the set of all configuration states h' for which $\sum_{l'a'} \pi^{\text{xdp}}_{\phi'} > 0$. It can be shown that the reconfiguration policy (11) of the decoupled algorithm results in a finite state Markov chain $\mathcal{M}$ over the states in $\mathcal{H}'$ (see Lemma 2 in Appendix B). Further, all states $h \in \mathcal{H}'$ are positive recurrent. Suppose the Markov chain $\mathcal{M}$ is in configuration h at time t and let T_h^{dec} denote the time spent in other configurations before returning to h (recurrence time). Then, by basic renewal theory [18], the following holds for all t.

$$\frac{\mathbb{E}\left\{\sum_{\tau=t}^{t+T_h^{\text{dec}}-1} \mu^{\text{dec}}_{km}(\tau)\right\}}{\mathbb{E}\left\{T_h^{\text{dec}}\right\}} = \mu^{\text{dec}}_{km}, \quad \frac{\mathbb{E}\left\{\sum_{\tau=t}^{t+T_h^{\text{dec}}-1} W^{\text{dec}}(\tau)\right\}}{\mathbb{E}\left\{T_h^{\text{dec}}\right\}} = w^* \tag{14}$$

Further, the first and second moments of the recurrence times, i.e., $\mathbb{E}\left\{T_h^{\text{dec}}\right\}$ and $\mathbb{E}\left\{(T_h^{\text{dec}})^2\right\}$ are bounded.

V. Online Control Algorithm

We now present an online control algorithm that makes joint request routing and application configuration decisions as a function of the system state $(\boldsymbol{l}(t), \boldsymbol{a}(t), \boldsymbol{h}(t), \boldsymbol{U}(t))$. However, unlike traditional MDP solution approaches such as dynamic programming [5], [6], this algorithm does not require any knowledge of the transition probabilities that govern the system dynamics. In addition to the request queues $U_{km}(t)$, for each (k, m) this algorithm maintains the following "delay-aware" queues that are used to provide worst-case delay guarantees for user requests (as shown later in Theorem 3) and are similar to the delay-aware queues used in [8].

$$Z_{km}(t+1) = \begin{cases} \max[Z_{km}(t) - \mu_{km}(t) - \upsilon_{km}(t) + \sigma_{km}, 0] & \text{if } U_{km}(t) > \mu_{km}(t) + \upsilon_{km}(t), \\ 0 & \text{if } U_{km}(t) \leq \mu_{km}(t) + \upsilon_{km}(t) \end{cases} \tag{15}$$

where $0 \leq \sigma_{km} \leq \upsilon^{\max}_{km}$ are control parameters that affect the delay guarantees offered by this algorithm. Our algorithm also uses a control parameter $V > 0$ that affects a cost-delay tradeoff made precise in Theorem 3. Denote the collection $\{Z_{km}(t)\}$ by $\boldsymbol{Z}(t)$ and the collection $\{\sigma_{km}\}$ by $\boldsymbol{\sigma}$. We assume that all request queues $U_{km}(t)$ and delay-aware queues $Z_{km}(t)$ are initialized to 0 at $t = 0$. As shown in the following, our online algorithm is designed to ensure that all request and delay-aware queues remain bounded for all t and this guarantees a deterministic *worst-case delay bound* for each request. Similar to the decoupled control algorithm defined by (11), (12), and (13), this algorithm consists of decoupled components for routing and reconfiguration decisions. The control decisions in each component are made independently but they are weakly coupled through the queue backlogs $\boldsymbol{U}(t)$ and $\boldsymbol{Z}(t)$. In the following, we describe each of these components in detail. The performance guarantees provided by our algorithm are presented in Section VI.

A. User-to-Edge-Cloud Request Routing

We first describe the user-to-edge-cloud routing component of the algorithm. In each slot t, the routing decisions $\{r_{knm}(t)\}$ are obtained by solving the following optimization problem.

$$\begin{aligned} \text{Minimize} \quad & \sum_{k,m} \sum_{n} \Big(U_{km}(t) + V c_{knm}(t)\Big) r_{knm}(t) \\ \text{subject to} \quad & \{r_{knm}(t)\} \in \mathcal{R}(l, a) \end{aligned} \tag{16}$$

where $\mathcal{R}(l,a)$ is defined by constraints (1), (2), $r_{knm}(t) \in \mathbb{Z}_{\geq 0} \forall k,n,m$, and other location-based constraints as discussed in Sec. III-A. The resulting problem is an integer linear program (ILP) in the variables $r_{knm}(t)$. Further, the problem is separable across k, i.e., it is optimal to solve K such problems separately, one per application k. When $R_{km}^{\max} \geq \sum_{n=1}^{N} A_{kn}(t)$ for all k,m, the above optimization has a particularly simple solution that can be obtained independently for each user n and can be calculated in closed-form as follows. For each (k,n), set $r_{knm^*}(t) = A_{kn}(t)$ for the particular edge-cloud m^* that user n can route to (given its current location $L_n(t)$) and that minimizes $U_{km}(t) + Vc_{knm}(t)$. Set $r_{knm}(t) = 0$ for all $m \neq m^*$. Note that $c_{knm}(t)$ depends on the current user location ($L_n(t)$) as well as the indices of the application (k) and the edge-cloud (m). This algorithm can be viewed as a more general version of the "Join the Shortest Queue" policy which uses only queue lengths. In contrast, here a weighted sum of queue length and transmission cost is used to determine the "shortest" queue.

More generally, (16) can be mapped to variants of matching problems on bipartite graphs. For example, consider the case where $A_{kn}^{\max} = 1$ for all k,n, $R_{km}^{\max} = 1$ for all k,m, and $N \leq M$. Then (16) becomes an instance of the minimum weight matching problem on a bipartite graph formed between the N users and M edge-clouds. This can be solved in polynomial time using well-known methods (such as in [19]). For more general cases, (16) becomes a generalized assignment problem that is NP-hard. However, efficient constant factor approximation algorithms are known for such problems [20]. As we show in Theorem 3, using any such approximation algorithm instead of the optimal solution to (16) ensures that the overall cost of the online algorithm is within the same approximation factor of the optimal cost. It should be noted that the routing component of the control algorithm considers only the current user location, request arrival, and queue backlog states to make decisions and is therefore *myopic*. Further, it does not require any knowledge of the mobility/arrival model. We also note that it does not depend on the application configuration state.

B. Edge-Cloud to Back-End Cloud Request Routing

The back-end routing decisions $\{\upsilon_{km}(t)\}$ are obtained by solving the following optimization problem every slot.

$$\begin{aligned} \text{Minimize} \quad & \sum_{km} \Big(Ve_{km} - U_{km}(t) - Z_{km}(t) \Big) \upsilon_{km}(t) \\ \text{subject to} \quad & 0 \leq \upsilon_{km}(t) \leq \min[U_{km}(t), \upsilon_{km}^{\max}] \ \ \forall k,m \end{aligned} \tag{17}$$

This problem is separable across (k,m) and has a simple solution given by $\upsilon_{km}(t) = \min[U_{km}(t), \upsilon_{km}^{\max}]$ when $U_{km}(t) + Z_{km}(t) > Ve_{km}$ and $\upsilon_{km}(t) = 0$ else. Similar to request routing, the back-end routing algorithm considers only current queue backlogs (as e_{km} is a constant) and does not require any knowledge of the user mobility or request arrival model. Further, it does not depend on the application configuration state. The structure of the back-end routing decisions results in the following bounds on $U_{km}(t)$ and $Z_{km}(t)$.

Lemma 1: Suppose $\upsilon_{km}^{\max} \geq R_{km}^{\max}$ for all k,m. Then, under the back-end routing decisions resulting from (17), the following hold for all t.

$$U_{km}(t) \leq U_{km}^{\max} = Ve_{km} + R_{km}^{\max} \tag{18}$$

$$Z_{km}(t) \leq Z_{km}^{\max} = Ve_{km} + \sigma_{km} \tag{19}$$

Proof: We show that (18) holds using induction. First, (18) holds for $t = 0$ since all queues are initialized to 0. Now suppose $U_{km}(t) \leq U_{km}^{\max}$ for some $t > 0$. Then, we show that $U_{km}(t+1) \leq U_{km}^{\max}$. We have two cases. First, suppose $U_{km}(t) \leq Ve_{km}$. Then, from queueing equation (5), it follows that the maximum value that $U_{km}(t+1)$ can have is $U_{km}(t) + R_{km}^{\max} \leq Ve_{km} + R_{km}^{\max} = U_{km}^{\max}$. Next, suppose $Ve_{km} < U_{km}(t) \leq U_{km}^{\max}$. Then, we have that $U_{km}(t) + Z_{km}(t) > Ve_{km}$ and the solution to (17) chooses $\upsilon_{km}(t) = \min[U_{km}(t), \upsilon_{km}^{\max}]$. Since $\upsilon_{km}^{\max} \geq R_{km}^{\max}$, from queueing equation (5) it follows that $U_{km}(t+1) \leq U_{km}(t) \leq U_{km}^{\max}$. The bound (19) follows similarly and its proof is omitted for brevity. ■

In Theorem 3, we show that for any $\sigma_{km} > 0$, the above bounds result in deterministic worst case delay bounds for any request that gets routed to $U_{km}(t)$.

C. Application Reconfiguration

The third component of the online algorithm performs application reconfigurations over time. We first define the notion of a *renewal state* under this reconfiguration algorithm. Consider any specific state $h_0 \in \mathcal{H}'$ and designate it as the renewal state. The application reconfiguration algorithm presented in this section is designed to operate over variable length renewal frames where each frame starts with the initial configuration h_0 (excluded from the current frame) and ends when it returns to the state h_0 (included in the current frame). All application configuration decisions for a frame are made at the start of the frame and are recalculated for each new frame as a function of the queue backlogs at the start of the frame. Note that the system configuration in the last slot of each frame is h_0. Each visit to h_0 defines a renewal event and initiates a new frame that starts from the next slot and lasts until (and including) the slot when the next renewal event happens as illustrated by an example in Fig. 2. The renewal event and the resulting frame length are fully determined by the configuration decisions of the reconfiguration algorithm, i.e., they are *deterministic functions* of the configuration decisions. In the following, we denote the length of the f^{th} renewal frame by T_f and the starting slot of the f^{th} renewal frame by t_f. Note that $T_f = t_{f+1} - t_f$. For simplicity, we assume $t_0 = 0$.

Recall that $\mathcal{H}'$ is the set of all configuration states h' for which $\sum_{l'a'} \pi_{\phi'}^{\text{xdp}} > 0$. In principle, any state in $\mathcal{H}'$ can be chosen to be the renewal state h_0. However, $\mathcal{H}'$ itself may not be known apriori. Further, in practice, h_0 should be chosen as the configuration that is likely to be used frequently by the optimal policy for the relaxed MDP presented in Section IV. Here, we assume that the reconfiguration algorithm can select a renewal state $h_0 \in \mathcal{H}'$ and leave the determination of optimal selection of h_0 for future work.

Let the collection of queue backlogs at the start of renewal frame f be denoted by $\{U_{km}(t_f)\}$ and $\{Z_{km}(t_f)\}$. Then the reconfiguration algorithm makes decisions on the frame length T_f and the application configurations $[h(t_f), h(t_f+1), \ldots, h(t_f+T_f-1)]$ by solving the following optimization at t_f.

$$\begin{aligned} \text{Minimize} \quad & \frac{1}{T_f} \sum_{\tau=0}^{T_f-1} \Big(J\tau + VW(t_f+\tau) - \sum_{km} G_{km}(t_f, \tau) \Big) \\ \text{subject to} \quad & h(t_f+T_f-1) = h_0 \\ & h(t_f+\tau) \in \mathcal{H} \setminus h_0 \; \forall \tau \in \{0, \ldots, T_f-2\} \\ & T_f \geq 1 \end{aligned} \tag{20}$$

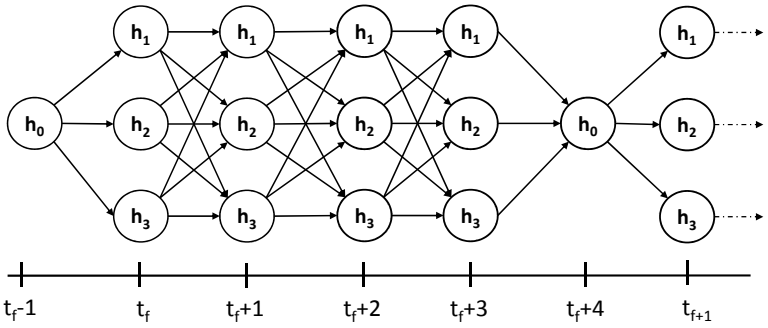


Fig. 2. Illustration of the directed acyclic graph on the application configuration states over a renewal frame. Frame f starts at slot t_f with the configuration changing from h_0 to one of h_1, h_2, h_3 and ends at slot $t_f + 4$ when the configuration becomes h_0 again.

where $G_{km}(\tau, t_f) = \big(U_{km}(t_f) + Z_{km}(t_f)\big)\mu_{km}(t_f + \tau)$ denotes the queue-length weighted service rate, $W(t_f + \tau)$ denotes the reconfiguration cost incurred in slot $(t_f + \tau)$, and $J = \sum_{km} J_{km}$ where J_{km} is a constant defined as $J_{km} \triangleq 2(\mu_{km}^{\max} + \upsilon_{km}^{\max})^2 + \sigma_{km}^2 + (R_{km}^{\max})^2$. Note that the constraint $h(t_f + T_f - 1) = h_0$ enforces the renewal condition. Note also that when the frame starts ($\tau = 0$), the configuration in the previous slot $t_f - 1$ was h_0. The problem above minimizes the ratio of the sum total "penalty" earned in the frame (given by the summation multiplying $1/T_f$ above) to the length of the frame. The penalty term is a sum of V times the reconfiguration costs ($VW(t_f + \tau)$) and the $J\tau$ terms minus the queue-length weighted service rates ($\sum_{km} G_{km}(t_f, \tau)$).

Since the sum of the $J\tau$ terms grows quadratically with frame size, this discourages the use of longer frames. Note also that since the overall objective only uses the queue backlog values at the start of the frame and since all the other terms are deterministic functions of the sequence of configurations $[h(t_f), \ldots, h(t_f + T_f - 1)]$, the optimization problem (20) can be mapped to a *deterministic shortest path problem* involving T_f stages. Specifically, as illustrated in Fig. 2, consider a directed acyclic graph with $T_f + 1$ stages, one node each in the first and last stage (corresponding to configuration h_0), and $|\mathcal{H}| - 1$ nodes per stage in all other stages. For a fixed T_f, the objective in (20) corresponds to finding the minimum cost path from the first to the last node, where the weight of each directed edge (h_i, h_j) that is $\tau + 1$ hops away from the first node is equal to the terms $J\tau + VW(t_f + \tau) - \sum_{km} G_{km}(t_f, \tau)$. Here, $W(t_f + \tau)$ is the switching cost between the configurations h_i and h_j while $\sum_{km} G_{km}(t_f, \tau)$ corresponds to the queue-length weighted service rate achieved using configuration h_j. Given a T_f, this has a complexity $O(|\mathcal{H}|^2 T_f)$ and optimally solving (20) would require searching over all $T_f \geq 1$ since T_f itself is an optimization variable in this problem. We next characterize an important property of the optimal solution to (20) that results in a significantly lower complexity.

1) Complexity Reduction: Consider any solution to (20) that results in a frame length of T_f and a configuration sequence given by $[h(t_f), \ldots, h(t_f + T_f - 2), h_0]$. Then we have the following.

Theorem 2: An optimal solution to (20) can be obtained by restricting to the class of policies that perform either two or no reconfigurations per frame. Further, the reconfigurations (if any) happen only in the first slot and the last slot of the frame.

Proof: First consider the case $1 \leq T_f \leq 2$. By definition the last configuration in the frame must be h_0. Similarly, the configuration in the previous slot before the start of the frame is h_0. There can be at most one more configuration between these. Thus, there can be at most two reconfigurations in the frame.

Next, consider the case $T_f > 2$. Let the queue backlogs at the start of frame be $\boldsymbol{U}(t_f), \boldsymbol{Z}(t_f)$ and suppose the optimal configuration sequence is given by $[h(t_f), \ldots, h(t_f + T_f - 2), h_0]$. Denote the set of configurations in this sequence by $\Omega(\boldsymbol{U}(t_f), \boldsymbol{Z}(t_f))$ and define $h_{\text{opt}}(\boldsymbol{U}(t_f), \boldsymbol{Z}(t_f))$ as the configuration from this sequence that minimizes the following:

$$h_{\text{opt}}(\boldsymbol{U}(t_f), \boldsymbol{Z}(t_f)) = \underset{h \in \Omega(\boldsymbol{U}(t_f), \boldsymbol{Z}(t_f))}{\arg\min} \sum_{km} \big(U_{km}(t_f) + Z_{km}(t_f)\big)\mu_{km}(t_f + \tau) \tag{21}$$

Now consider an alternate configuration sequence given by $[h_{\text{opt}}(u_f), \ \ldots, \ h_{\text{opt}}(u_f), h_0]$. The value of the summation in the objective of (20) under this sequence cannot be larger than that under the sequence $[h(t_f), \ldots, h(t_f + T_f - 2), h_0]$. This is because $h_{\text{opt}}(\boldsymbol{U}(t_f), \boldsymbol{Z}(t_f))$ minimizes the terms $-\sum_{km} G_{km}(t_f, \tau)$ by (21) while the total reconfiguration cost $\sum_{\tau=0}^{T_f-1} VW(t_f + \tau)$ in the alternate sequence cannot exceed the total reconfiguration cost under $[h(t_f), \ldots, h(t_f + T_f - 2), h_0]$ by property (4). The theorem follows by noting that at most two reconfigurations are needed in the alternate sequence, one at the beginning and one at the end of the frame. ■

For a given frame length T_f, Theorem 2 reduces the complexity of solving (20) from $O(|\mathcal{H}|^2 T_f)$ to $O(|\mathcal{H}|)$ since we only need to search for one configuration per frame. In Section V-C2, we show that the reconfiguration algorithm can be further simplified by finding a *closed-form expression* for the optimal frame length given a configuration h. This frame length is $O\Big(\sqrt{\sum_{km} U_{km}(t_f) + Z_{km}(t_f)}\Big)$ which shows that the frame length is always bounded, given that $\{U_{km}(t_f)\}$ and $\{Z_{km}(t_f)\}$ are bounded (see Lemma 1). We also discuss special cases where (20) can be mapped to bipartite graph matching problems. We will show that similar to the routing component, using any approximation algorithm instead of the optimal solution to (20) still ensures that the overall cost of the reconfiguration algorithm is within the same constant factor of the optimal cost (Theorem 3, part 3).

We analyze the performance of the overall control algorithm, including the components for request routing (Sections V-A and V-B) and the component for application reconfiguration in Section VI. Before proceeding, we note that similar to the routing components, the reconfiguration algorithm does not require any statistical knowledge of the request arrival or user mobility processes. Further, it depends only on the queue backlog at the start of the frame, thereby decoupling it from the routing decisions in that frame. However, unlike the routing components that make decisions on a per slot basis (using current system states), the reconfiguration algorithm computes the sequence of configurations once per frame (at the start of the frame) and implements it over the course of the frame.

2) Calculating the Optimal Frame Length: Given that a configuration state h is used in a frame, we show that the optimal frame length $T_{\text{opt}}(h)$ can be easily calculated, thereby further reducing complexity. Specifically, let us denote the values of various terms in the objective of (20) when configuration h_0 or h are used as follows: $\Theta(h_0) = -\sum_{km} G^{h_0}_{km}(t_f, \tau) = -\sum_{km}(U_{km}(t_f) + Z_{km}(t_f))\mu^{h_0}_{km}, \Theta(h) = -\sum_{km} G^{h}_{km}(t_f, \tau) = -\sum_{km}(U_{km}(t_f) + Z_{km}(t_f))\mu^{h}_{km}, W_{\text{sum}} = W_{h_0 h} + W_{h h_0}$, where $\mu^{h_0}_{km}, \mu^{h}_{km}$ denote the service rate μ_{km} when configurations h_0 and h are used. Also, let $\hat{B} = \sum_{km} J_{km}/2$. Then, in order to calculate the optimal frame length, we have two cases. If no reconfiguration is done, then frame length is 1 and the objective of (20) becomes $\Theta(h_0)$. Else, if a reconfiguration is done, the frame length is at least 2 and the objective of (20) can rewritten as

$$\min_{T_f \geq 2} \frac{\Theta(h_0) + \Theta(h)(T_f - 1) + \hat{B}T_f(T_f - 1) + VW_{\text{sum}}}{T_f}$$

Ignoring the constant terms, the above can be simplified to

$$\min_{T_f \geq 2} \hat{B} T_f + \frac{V W_{\text{sum}} + \Theta(h_0) - \Theta(h)}{T_f}$$

If $V W_{\text{sum}} + \Theta(h_0) - \Theta(h) \leq 0$, then the optimal frame length is $T_f = 2$. Else, by taking derivative, we get that the optimal T_f is one of the two integers closest to $\sqrt{\frac{V W_{\text{sum}} + \Theta(h_0) - \Theta(h)}{\hat{B}}}$. Define $T_f^*(h)$ as the optimal frame length given that a switching to configuration h is done. Then, to calculate the overall optimal frame length $T_{\text{opt}}(h)$, we compare the value of the objective of (20) under $T_f^*(h)$ with that under frame length one (i.e., $\Theta(h_0)$) and select the one that results in a smaller objective.

3) Bipartite Matching: Note that a complexity of $O(|\mathcal{H}|)$ can still be high if there are a large number of possible configurations. For example, consider the case where at most one application can be hosted by any edge-cloud and where exactly one instance of each application is allowed per slot across the edge-clouds. Assume $M \geq K$. In this case, $|\mathcal{H}| = \frac{M!}{(M-K)!}$ which is exponential in M, K. Now assume that the total reconfiguration cost is separable across applications, the service rate of each edge-cloud is independent of configurations at the other edge-clouds, and the other settings are the same as in the above example. Then (20) can be reduced to a maximum weight matching problem on a bipartite graph formed between K applications and M edge-clouds. Let m'_k denote the edge-cloud hosting application k in the renewal state h_0. Then in the bipartite graph, the weight for any edge (k, m) (when $m \neq m'_k$) for a given frame length T_f becomes $(U_{km'_k}(t_f) + Z_{km'_k}(t_f))\mu_{km'_k} + (T_f - 1)(U_{km}(t_f) + Z_{km}(t_f))\mu_{km} - V(W_{k,m'_k,m} + W_{k,m,m'_k}) - \frac{J_{km} T_f (T_f - 1)}{2}$ where $W_{k,m'_k,m}$ (W_{k,m,m'_k}) denotes the reconfiguration cost associated with moving application k from edge-cloud m'_k (m) to edge-cloud m (m'_k). When $m = m'_k$, the weight for the edge (k, m) is simply $(U_{km'_k}(t_f) + Z_{km'_k}(t_f))\mu_{km'_k}$.

For a given T_f, the optimal configuration that solves (20) can be obtained by finding the maximum weight matching on this bipartite graph and this is polynomial in M, K [19]. This is in contrast to simply searching over all $\mathcal{H}$ that is exponential in M, K. For more general cases where each application can have multiple instances and multiple application instances are allowed on each edge-cloud, the problem becomes a generalized assignment problem and constant factor approximation algorithms exist [20], as long as different application instances can be considered separately.

VI. Performance Analysis

We now analyze the performance of the online control algorithm presented in Section V. This is based on the technique of Lyapunov optimization over renewal periods [8], [17] where we compare the ratio of a weighted combination of the Lyapunov drift and costs over a renewal period and the length of the period under the online algorithm with the same ratio under a stationary algorithm that is queue backlog independent. This stationary algorithm is defined similarly to the decoupled control algorithm given by (11), (12) and (13) and we use the subscript "stat" to denote its control actions and the resulting service rates and costs. Then the reconfiguration and routing decisions are defined by probabilities $\theta^{\text{stat}}_{hh'}, \zeta^{\text{stat}}_{la}(r)$, and $\vartheta^{\text{stat}}(\upsilon)$ that are chosen to be equal to $\theta^{\text{dec}}_{hh'}, \zeta^{\text{dec}}_{la}(r)$, and $\vartheta^{\text{dec}}(\upsilon)$ respectively. If the resulting expected total service rate of any delay-aware queue $Z_{km}(t)$ is less than σ_{km}, then its back-end request routing is augmented by choosing additional $\upsilon_{km}(t)$ in an i.i.d. manner such that the expected total service rate becomes σ_{km}. It can be shown that the resulting time-average back-end routing cost under this algorithm is at most $e^* + \phi(\boldsymbol{\sigma})$ where $\phi(\boldsymbol{\sigma}) = \sum_{km} \max[\sigma_{km} - \mu^{\text{dec}}_{km} - \upsilon^{\text{dec}}_{km}, 0] e_{km}$. By comparing the Lyapunov drift plus cost of the online control algorithm over renewal frames with this stationary algorithm,

we have the following.

Theorem 3: Suppose the online control algorithm defined by (16), (17), and (20) is implemented with a renewal state $h_0 \in \mathcal{H}'$ using control parameters $V > 0$ and $0 \leq \sigma_{km} \leq \upsilon_{km}^{\max}$ for all k, m. Denote the resulting sequence of renewal times by t_f where $f \in \{0, 1, 2, \ldots\}$ and let $T_f = t_{f+1} - t_f$ denote the length of frame f. Assume $t_0 = 0$ and that $U_{km}(t_0) = 0, Z_{km}(t_0) = 0$ for all k, m. Then the following bounds hold.

1) The time-average expected transmission plus reconfiguration costs satisfy

$$\lim_{F\to\infty} \frac{\sum_{f=0}^{F-1} \mathbb{E}\left\{\sum_{\tau=t_f}^{t_{f+1}-1} C(\tau) + W(\tau) + E(\tau)\right\}}{\sum_{f=0}^{F-1} \mathbb{E}\left\{T_f\right\}} \leq c^* + w^* + e^* + \phi(\boldsymbol{\sigma}) + \frac{1 + \sum_{km} B_{km}}{V} \tag{22}$$

where $B_{km} = (1 + \Upsilon_{h_0}^{\text{dec}})J_{km}/2 + \delta(R_{km}^{\max})^2$, $\Upsilon_{h_0}^{\text{dec}} = \frac{\mathbb{E}\left\{T_{h_0}^{\text{dec}}(T_{h_0}^{\text{dec}}-1)\right\}}{\mathbb{E}\left\{T_{h_0}^{\text{dec}}\right\}}$, $\phi(\boldsymbol{\sigma}) = \sum_{km} \max[\sigma_{km} - \mu_{km}^{\text{dec}} - \upsilon_{km}^{\text{dec}}, 0]e_{km}$ and δ is an $O(\log V)$ parameter that is a function of the mixing time of the Markov chain defined by the user location and request arrival processes while all other terms are constants (independent of V).

2) For all k, m, the worst-case delay $d_{km}^{\max}$ for any request routed to queue $U_{km}(t)$ is upper bounded by

$$d_{km}^{\max} \leq \frac{U_{km}^{\max} + Z_{km}^{\max}}{\sigma_{km}} = \frac{2Ve_{km} + R_{km}^{\max} + \sigma_{km}}{\sigma_{km}} \tag{23}$$

3) Suppose we implement an algorithm that approximately solves (16) and (20) resulting in the following bound for all slots for some $\rho \geq 1$

$$\sum_{km}\sum_{n} \Big(U_{km}(t) + Vc_{knm}(t)\Big) r_{knm}^{\text{apx}}(t) \leq \rho \sum_{km}\sum_{n} \Big(U_{km}(t) + Vc_{knm}(t)\Big) r_{knm}^{\text{opt}}(t) \tag{24}$$

and the following bound every renewal frame

$$\begin{aligned}
&\frac{1}{T_f^{\text{apx}}} \sum_{\tau=0}^{T_f^{\text{apx}}-1} \Big(\sum_{km} \rho J_{km}\tau - \big(U_{km}(t_f) + Z_{km}(t_f)\big)\mu_{km}^{\text{apx}}(t_f+\tau)\Big) + VW^{\text{apx}}(t_f+\tau) \\
&\leq B' + \frac{\rho}{T_f^{\text{opt}}} \sum_{\tau=0}^{T_f^{\text{opt}}-1} \Big(\sum_{km} J_{km}\tau - \big(U_{km}(t_f) + Z_{km}(t_f)\big)\mu_{km}^{\text{opt}}(t_f+\tau)\Big) + VW^{\text{opt}}(t_f+\tau)
\end{aligned} \tag{25}$$

for some constant B' where the subscripts "apx" and "opt" denote the control decisions and resulting costs under the approximate algorithm and the optimal solution to (16) and (20) respectively. Then the time-average expected transmission plus reconfiguration costs under this approximation algorithm is at most

$$\rho\Big(c^* + w^* + e^* + \phi(\boldsymbol{\sigma}) + \frac{1 + \sum_{km} B_{km} + (R_{km}^{\max} + \sigma_{km})\upsilon_{km}^{\max}}{V}\Big) + \frac{B' - \sum_{km}(R_{km}^{\max} + \sigma_{km})\upsilon_{km}^{\max}}{V} \tag{26}$$

while the delay bounds remain the same as (23).

Proof: See Appendix C. ∎

Discussion on Performance Tradeoffs: Our control algorithm offers tradeoffs between cost and delay performance guarantees through the control parameters V and $\boldsymbol{\sigma}$. For a given V and $\boldsymbol{\sigma}$, the bound in (22) implies that the time-average expected transmission plus reconfiguration costs are within an additive term $\phi(\boldsymbol{\sigma}) + O(\log V/V)$ term of the optimal cost while (23) bounds the worst case delay by $O(V/\sigma_{km})$. This shows that by increasing V and decreasing $\boldsymbol{\sigma}$, the time-average cost can be pushed arbitrarily close to optimal at the cost of an increase in delay. This cost-delay tradeoff

is similar to the results in [7], [8] for non-MDP problems. Thus, it is noteworthy that we can achieve similar tradeoff in an MDP setting. Note that if there exist $\sigma_{km} > 0 \ \forall k, m$ such that $\sigma_{km} \leq \mu_{km}^{\text{dec}} + \upsilon_{km}^{\text{dec}}$, then $\phi(\boldsymbol{\sigma}) = 0$ and the tradeoff can be expressed purely in terms of V. Also note that since the average delay is upper bounded by the worst case delay, our algorithm provides an additive approximation with respect to the cost $c^* + w^* + e^*$ and a multiplicative approximation with respect to the average delay d_{avg} of the optimal solution to the original MDP defined in Section III-B.

In practice, setting $\sigma_{km} = 0 \ \forall k, m$ should yield good delay performance even though (23) becomes unbounded. This is because our control algorithm ensures that all queues remain bounded even when $\sigma_{km} = 0$ (see Lemma 1). This hypothesis is confirmed by the simulation results in the next section.

VII. Evaluations

We evaluate the performance of our control algorithm using simulations. To show both the theoretical and real-world behaviors of the algorithm, we consider two types of user mobility traces. The first is a set of synthetic traces obtained from a random-walk user mobility model while the second is a set of real-world traces of San Francisco taxis [21]. We assume that the edge-clouds are co-located with a subset of the basestations of a cellular network. A hexagonal symmetric cellular structure is assumed with 91 cells in total as shown in Fig. 3. Out of the 91 basestations, 10 host edge-clouds and there are 5 applications in total. For simplicity, each edge-cloud can host at most one application in any slot in the simulation. Further, there can be only one active instance of any application in a slot.

The transmission and reconfiguration costs are defined as a function of the distance (measured by the smallest number of hops) between different cells. When a user n in cell l routes its request to the edge-cloud in cell l', we define its transmission cost as

$$\text{trans}_n(l, l') = \begin{cases} 1 + 0.1 \cdot \text{dist}(l, l'), & \text{if } l \neq l' \\ 0, & \text{if } l = l' \end{cases} \tag{27}$$

where $\text{dist}(l, l')$ is the number of hops between cells l and l'. The reconfiguration cost of different applications is assumed to be independent. For any application k that is moved from the edge-cloud in cell l to the edge-cloud in cell l', the reconfiguration cost for this specific application is defined as

$$\text{recon}_k(l, l') = \begin{cases} \kappa(1 + 0.1 \cdot \text{dist}(l, l')), & \text{if } l \neq l' \\ 0, & \text{if } l = l' \end{cases} \tag{28}$$

where κ is a weighting factor to compare the reconfiguration cost to the transmission cost. The total reconfiguration cost is the sum of reconfiguration costs across all k. In the simulations, we consider two cases in which κ takes the values 0.5 and 1.5 respectively, to represent cases where the reconfiguration cost is smaller/larger than the transmission cost. Both cases can occur in practice depending on the amount of state information the application has to transfer during reconfiguration. The back-end routing cost is fixed as a constant 2 for each request.

Each user generates requests for an application according to a fixed probability λ per slot. However, the number of active users in the system can change over time. Thus, the aggregate request arrival rate across all users for an application varies as a function of the number of active users in a slot. In our study of synthetic mobility traces, we assume that the number of users is fixed to 10 and all of them are active. However, the real-world mobility trace has a time-varying number of

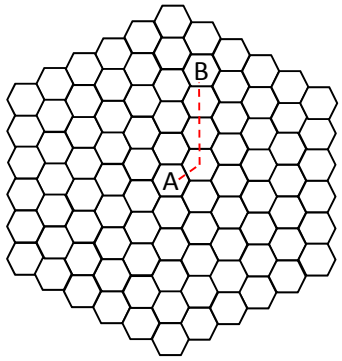


Fig. 3. Illustration of the hexagonal cellular structure showing distance between 2 cells.

active users. In both cases, λ is the time-average (over the simulation duration) aggregate arrival rate per application per slot, while the edge cloud service rate for an active application instance is 1 per slot, and the back-end cloud service rate for each application is 2 per slot. The request arrivals are assumed to be independent and identically distributed among different users, and they are also independent of the past arrivals and user locations.

We note that optimally solving the original or even relaxed MDP for this network is highly challenging. Therefore, we compare the performance of our algorithm with three alternate approaches that include never/always migrate policies and a myopic policy. In the never migrate policy, each application is initially placed at one particular edge-cloud and reconfiguration never happens. User requests are always routed to the edge-cloud that hosts the corresponding application. In the always migrate policy, user requests are always routed to the edge-cloud that is closest to the user and reconfiguration is performed in such a way that the queues with the largest backlogs are served first (subject to the constraint that each edge-cloud can only host one application). We also assume that the request arrival rate λ is known in the never and always migrate policies. If $\lambda > 1$, the arrival rate exceeds the edge-cloud capacity, and the requests that are queued in edge-clouds are probabilistically routed to the back-end cloud, where the probability is chosen such that the average arrival rate to edge-clouds does not exceed the service rate at edge clouds. Finally, the myopic policy considers the transmission, reconfiguration, and back-end routing costs *jointly* in every slot. Specifically, in each slot, it calculates a routing and configuration option that minimizes the sum of these three types of costs in a *single* slot, where it is assumed that a user routes its request either to the back-end cloud or to the edge-cloud that hosts the application after possible reconfiguration.

The online algorithm itself is implemented by making use of the structure of the optimal solution as discussed in Section V. Specifically, we implement the request routing part (16) by solving the bipartite max-weight matching problem as discussed in Section V-A while the application reconfiguration part (20) uses the techniques in Section V-C3 and Section V-C2. Because the proposed online algorithm is obtained using a (loose) upper bound on the drift-plus-penalty terms, the actual drift-plus-penalty value can be much smaller than the upper bound. We take into account this fact by adjusting the constant terms J_{km} in (20). We set $J_{km} = 0.2$ in the simulation which is a reasonably good number that we found experimentally.

A. Synthetic Traces

We first evaluate the performance of our algorithm along with the three alternate approaches on synthetic mobility traces. The synthetic traces are obtained assuming random-walk user mobility. Specifically, at the beginning of each slot, a user

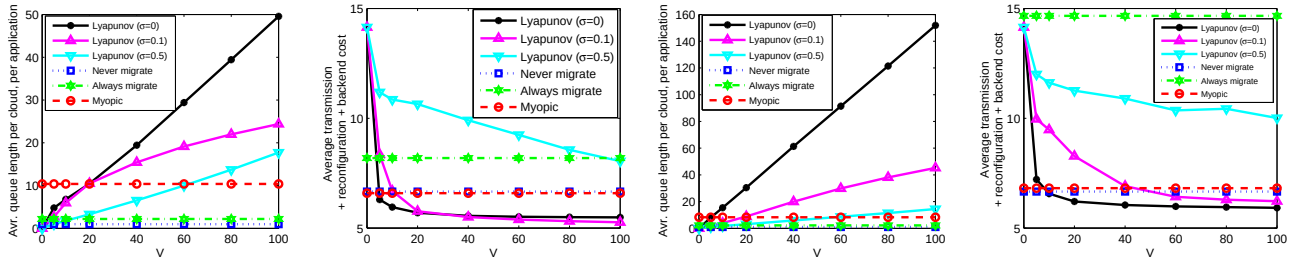


Fig. 4. Average queue lengths and costs for synthetic user mobility with different V and σ values. Subfigures (a) and (b) are results for $\kappa = 0.5$, and subfigures (c) and (d) are results for $\kappa = 1.5$.

moves to one of its neighboring cells with probability $1/7$ for each cell, and it stays in the same cell with probability $1/7$. When the number of neighboring cells is less than six, the corresponding probability is added to the probability of staying in the same cell. Such a mobility model can be described as a Markov chain and therefore our theoretical analysis applies.

There are 10 users in this simulation, and we simulate the system for $100,000$ slots. The average queue length and the average transmission plus reconfiguration plus back-end routing costs over the entire simulation duration are first studied for different values of the control parameters V as well as $\{\sigma_{km}\}$. Specifically, we set all σ_{km} to the same value σ which is chosen from $\sigma \in \{0, 0.1, 0.5\}$. The performance results for all four algorithms under these scenarios are shown in Fig. 4 for both values of κ, where we set $\lambda = 0.95$.

We can see from the results that, for each fixed σ, the queue lengths and cost values under the Lyapunov algorithm follow the $O(V, \log V/V)$ trend as suggested by the bounds (22) and (23). The impact of the value of σ is also as predicted by these bounds. Namely, a smaller value of σ yields larger queue lengths and lower costs, while a larger value of σ yields smaller queue lengths and higher costs. When comparing all four algorithms in Fig. 4(a), (b) where $\kappa = 0.5$, it can be seen that while the never/always migrate policies have smaller queue backlogs, they incur more cost than the Lyapunov algorithm. Note that, unlike the Lyapunov algorithm, none of the alternate approaches offers a mechanism to trade off queue backlog (and hence average delay) performance for a reduction in cost. For the case $\kappa = 1.5$, similar behavior is seen as illustrated by Fig. 4(c), (d).

We next study the queue lengths and costs under different values of the arrival rate λ, where we fix $V = 100$ and $\sigma = 0$. Results are shown in Fig. 5. We can see that with the myopic policy, the queue lengths are very large and in fact become unbounded. This is because the myopic policy does not try to match the edge-cloud arrival rate with its service rate, and it is also independent of the queue backlog. Because the one-slot cost of routing to an edge-cloud is usually lower than routing to the back-end cloud, an excessive amount of requests are routed to edge-clouds exceeding their service capacity. The never and always migrate policies have low queue backlogs because we matched the request routing with the service rate of edge-clouds, as explained earlier. However, they incur higher costs as shown in Fig. 5(b), (d). More importantly, they require prior knowledge on the arrival rate, which it is usually difficult to obtain in practice.

B. Real-World Mobility

To study the performance under more realistic user mobility, we use real-world traces of San Francisco taxis [21] that is a collection of GPS coordinates of approximately 500 taxis collected over 24 days in the San Francisco Bay Area. In our simulation, we select a subset of this data that corresponds to a period of 5 consecutive days. We set the distance

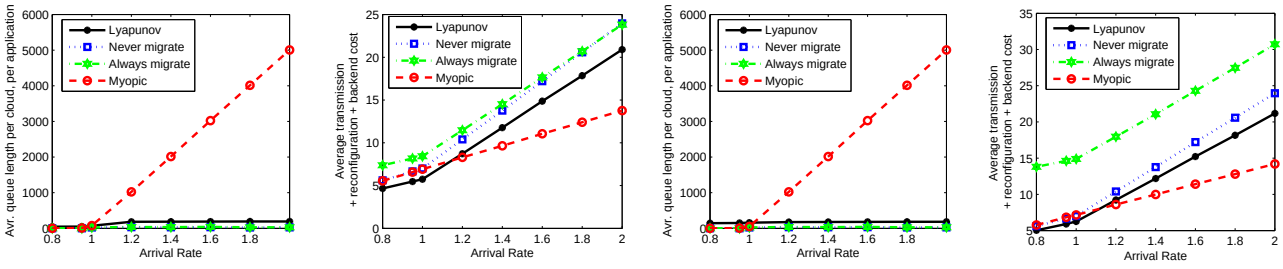


Fig. 5. Average queue lengths and costs for synthetic user mobility with different λ values. Subfigures (a) and (b) are results for $\kappa = 0.5$, and subfigures (c) and (d) are results for $\kappa = 1.5$.

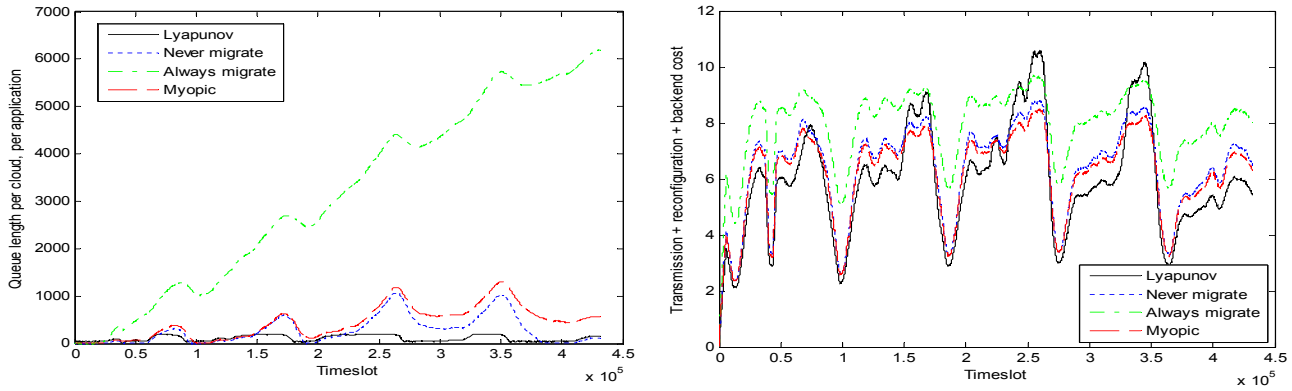
between basestations (center of cell) to 1000 meters, and the hexagon structure is placed onto the geographical location. User locations are then mapped to the cell location by considering which cell the user lies in. In this dataset, there are 536 unique users in total, and not all of them are active at a given time. The number of active users at any time ranges from 0 to 409, and 278 users are active on average. We assume that only active users generate requests such that the average arrival rate over the entire duration is $\lambda = 0.95$ for each application. With this model, when the number of active users is large (small), the instantaneous arrival rate can be higher (lower) than the edge-cloud service rate. The underlying mobility pattern in this scenario can be quite different from a stationary Markov model and exhibits non-stationary behavior.

We set the timeslot length as 1 second and fix $V = 100$, $\sigma = 0$ for the Lyapunov algorithm. The purpose of this simulation is to study the temporal behavior of queue lengths and cost values under our algorithm and compare with the alternate approaches. We find that while the queue lengths change relatively slowly, the per slot costs fluctuate rapidly. Therefore, we measure the moving average of the costs over an interval of size 6000 seconds for all algorithms. Figs. 6 and 7 show the results respectively for the case $\kappa = 0.5$ and $\kappa = 1.5$, and the average values across the entire time duration are given in Table I.

There are several noteworthy observations. From Table I, we can see that even though $\sigma = 0$, the average queue length under the Lyapunov approach is significantly lower than all other approaches, while the cost of the Lyapunov approach is lower than all other approaches when $\kappa = 0.5$ and only slightly higher than the never migrate and myopic policies when $\kappa = 1.5$. This confirms that the proposed Lyapunov algorithm has promising performance with real-world user traces. As shown in Figs. 6 and 7, the cost results show a noticeable diurnal behavior with 5 peaks and valleys that matches with the 5 day simulation period. The cost of the Lyapunov algorithm becomes higher than some other approaches at peaks, which is mainly due to the presence of back-end routing. At the same time, however, the difference between the queue length of the Lyapunov algorithm and the other approaches is also larger at such peaks. We see that the Lyapunov approach has the lowest variation in its queue length, which is a consequence of our design goal of bounding the worst-case delay. The queue lengths of the other approaches fluctuate more, and the always migrate policy appears to be unstable as the queue backlogs grow unbounded.

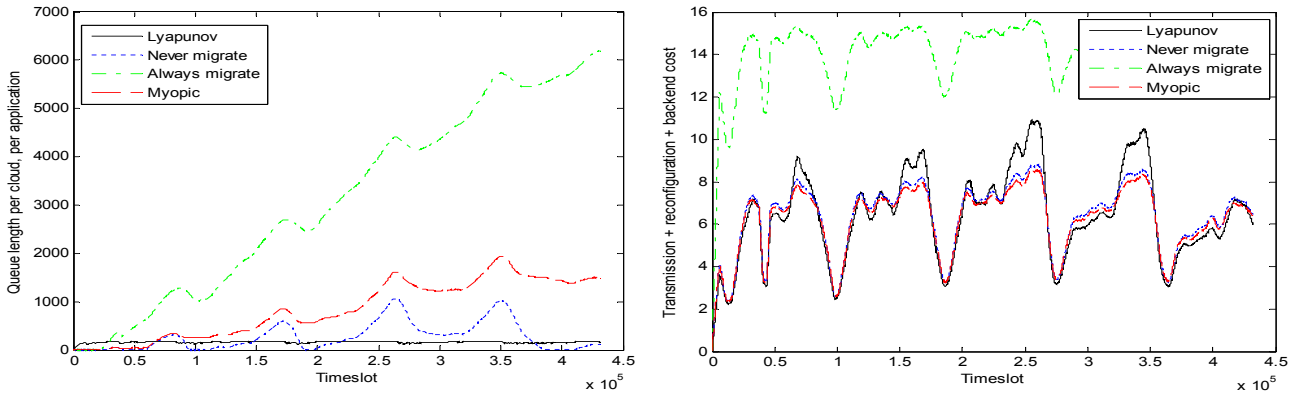
TABLE I. AVERAGE VALUES FOR TRACE-DRIVEN SIMULATION

Policy	Queue lengths ($\kappa = 0.5$)	Costs ($\kappa = 0.5$)	Queue lengths ($\kappa = 1.5$)	Costs ($\kappa = 1.5$)
Lyapunov	106.5	5.954	160.2	6.446
Never migrate	268.7	6.414	268.7	6.414
Always migrate	3117	7.983	3117	14.15
Myopic	437.4	6.228	851	6.268



(a) (b)

Fig. 6. Instantaneous queue lengths and moving average of costs for trace-driven simulation with $\kappa = 0.5$.



(a) (b)

Fig. 7. Instantaneous queue lengths and moving average of costs for trace-driven simulation with $\kappa = 1.5$.

VIII. Conclusions

In this paper, we have developed a new approach for solving a class of constrained MDPs that possess a decoupling property. When this property holds, our approach enables the design of simple online control algorithms that do not require any knowledge of the underlying statistics of the MDPs, yet are provably optimal. The resulting solution is markedly different from classical dynamic programming based approaches and does not suffer from the associated "curse of dimensionality" or convergence issues. We applied this technique to the problem of dynamic service migration and workload scheduling in the emerging area of edge-clouds and showed how it results in an efficient control algorithm for this problem. Our overall approach is promising and could be useful in a variety of other contexts.

Acknowledgements

This research was sponsored in part by the U.S. Army Research Laboratory and the U.K. Ministry of Defence and was accomplished under Agreement Number W911NF-06-3-0001. The views and conclusions contained in this document are those of the author(s) and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Army Research Laboratory, the U.S. Government, the U.K. Ministry of Defence or the U.K. Government. The U.S. and U.K. Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

REFERENCES

[1] M. Satyanarayanan, "Cloudlets: At the leading edge of cloud-mobile convergence," in *Proceedings of the 9th International ACM Sigsoft Conference on Quality of Software Architectures*, ser. QoSA '13. New York, NY, USA: ACM, 2013, pp. 1–2.

[2] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*. ACM, 2012, pp. 13–16.

[3] S. Davy, J. Famaey, J. Serrat-Fernandez, J. Gorricho, A. Miron, M. Dramitinos, P. Neves, S. Latre, and E. Goshen, "Challenges to support edge-as-a-service," *IEEE Communications Magazine*, vol. 52, no. 1, pp. 132–139, Jan. 2014.

[4] T. Taleb and A. Ksentini, "Follow me cloud: interworking federated clouds and distributed mobile networks," *IEEE Network*, vol. 27, no. 5, pp. 12–19, Sept. 2013.

[5] D. P. Bertsekas, *Dynamic Programming and Optimal Control*, 2nd ed. Athena Scientific, 2000.

[6] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*, 1st ed. John Wiley & Sons, Inc., 1994.

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

[8] M. J. Neely, *Stochastic Network Optimization with Application to Communication and Queueing Systems*. Morgan and Claypool Publishers, 2010.

[9] S. Maguluri, R. Srikant, and L. Ying, "Stochastic models of load balancing and scheduling in cloud computing clusters," in *INFOCOM, 2012 Proceedings IEEE*, March 2012, pp. 702–710.

[10] Y. Guo, A. Stolyar, and A. Walid, "Shadow-routing based dynamic algorithms for virtual machine placement in a network cloud," in *INFOCOM, 2013 Proceedings IEEE*, April 2013, pp. 620–628.

[11] Y. Guo, A. L. Stolyar, and A. Walid, "Online algorithms for joint application-vm-physical-machine auto-scaling in a cloud," in *The 2014 ACM International Conference on Measurement and Modeling of Computer Systems*, ser. SIGMETRICS '14. ACM, 2014, pp. 589–590.

[12] J. W. Jiang, T. Lan, S. Ha, M. Chen, and M. Chiang, "Joint vm placement and routing for data center traffic engineering." in *INFOCOM*. IEEE, 2012, pp. 2876–2880.

[13] M. Lin, A. Wierman, L. L. H. Andrew, and E. Thereska, "Dynamic right-sizing for power-proportional data centers," *IEEE/ACM Trans. Netw.*, vol. 21, no. 5, pp. 1378–1391, Oct. 2013.

[14] M. Lin, Z. Liu, A. Wierman, and L. L. H. Andrew, "Online algorithms for geographical load balancing," in *Proceedings of the 2012 International Green Computing Conference (IGCC)*, ser. IGCC '12, Washington, DC, USA, 2012, pp. 1–10.

[15] G. Celik and E. Modiano, "Scheduling in networks with time-varying channels and reconfiguration delay," *Networking, IEEE/ACM Transactions on*, vol. 23, no. 1, pp. 99–113, Feb 2015.

[16] A. Gandhi, S. Doroudi, M. Harchol-Balter, and A. Scheller-Wolf, "Exact analysis of the m/m/k/setup class of markov chains via recursive renewal reward," *SIGMETRICS Perform. Eval. Rev.*, vol. 41, no. 1, pp. 153–166, Jun. 2013.

[17] M. Neely, "Dynamic optimization and learning for renewal systems," *Auto. Control, IEEE Trans.*, vol. 58, no. 1, pp. 32–46, Jan 2013.

[18] S. M. Ross, *Introduction to Probability Models, Ninth Edition*. Orlando, FL, USA: Academic Press, Inc., 2006.

[19] T. H. Cormen, C. Stein, R. L. Rivest, and C. E. Leiserson, *Introduction to Algorithms*, 2nd ed. McGraw-Hill Higher Education, 2001.

[20] D. B. Shmoys and E. Tardos, "An approximation algorithm for the generalized assignment problem," *Math. Program.*, vol. 62, no. 3, pp. 461–474, Dec. 1993.

[21] M. Piorkowski, N. Sarafijanovic-Djukic, and M. Grossglauser, "A parsimonious model of mobile partitioned networks with clustering," in *Communication Systems and Networks and Workshops, 2009. COMSNETS 2009. First International*, Jan 2009, pp. 1–10.

[22] R. Urgaonkar and M. J. Neely, "Opportunistic scheduling with reliability guarantees in cognitive radio networks," *IEEE Transactions on Mobile Computing*, vol. 8, no. 6, pp. 766–777, Jun. 2009.

APPENDIX A
RELAXED MDP FORMULATION

An optimal stationary, randomized control algorithm for the relaxed MDP can be defined as follows. First, given last state $\phi' = (\boldsymbol{l}(t-1) = l', \boldsymbol{a}(t-1) = a', \boldsymbol{h}(t-1) = h')$ and current states $\boldsymbol{l}(t) = l, \boldsymbol{a}(t) = a$, it chooses configuration $\boldsymbol{h}(t) = h$ with probability $z_{\phi'\phi}$ where $\phi = (\boldsymbol{l}(t) = l, \boldsymbol{a}(t) = a, \boldsymbol{h}(t) = h)$. The total expected reconfiguration cost incurred when transitioning from state ϕ' is given by

$$W_{\phi'} = \sum_{\phi} p_{l'l} q_{a'a} z_{\phi'\phi} W_{h'h}. \tag{29}$$

Next, given current state $\phi = (\boldsymbol{l}(t) = l, \boldsymbol{a}(t) = a, \boldsymbol{h}(t) = h)$, it chooses a routing vector $\boldsymbol{r}(t) = r$ with probability $x_\phi(r)$ subject to the constraint that $r \in \mathcal{R}(l, a)$ and incurs a total expected transmission cost of

$$C_\phi = \sum_{r \in \mathcal{R}(l,a)} x_\phi(r) \sum_{knm} r_{knm} c_{knm} \tag{30}$$

Finally, given current state $\phi = (\boldsymbol{l}(t) = l, \boldsymbol{a}(t) = a, \boldsymbol{h}(t) = h)$, it chooses a back-end routing vector $\boldsymbol{\upsilon}(t) = \upsilon$ with probability $y_\phi(\upsilon)$ subject to $\upsilon \in \mathcal{V}$ and this incurs a total expected back-end transmission cost given by

$$E_\phi = \sum_{\upsilon \in \mathcal{V}} y_\phi(\upsilon) \sum_{km} \upsilon_{km} e_{km} \tag{31}$$

Let us denote the steady state probability of being in state ϕ under this policy by π_ϕ. Then, the overall time-average expected transmission plus reconfiguration cost satisfies

$$\overline{C} + \overline{E} + \overline{W} = \sum_{\phi} \pi_\phi C_\phi + \sum_{\phi} \pi_\phi E_\phi + \sum_{\phi} \pi_\phi W_\phi = c^* + e^* + w^* \tag{32}$$

Note that the relaxed MDP still has a high-dimensional state space making it impractical to solve using standard techniques. Further, we still need knowledge of all transition probabilities.

APPENDIX B
COST OPTIMALITY OF DECOUPLED MDP

Here we prove Theorem 1 through a series of lemmas. For notational convenience, we use ϕ and lah interchangeably throughout this section. Recall that $\mathcal{H}'$ is the set of all configuration states h' for which $\sum_{l'a'} \pi^{\text{xdp}}_{\phi'} > 0$. Then for all $h', h \in \mathcal{H}'$, we define matrix $F = (\theta^{\text{dec}}_{h'h})$ where $\theta^{\text{dec}}_{h'h}$ is given by (11).

Lemma 2: F is a stochastic matrix.

Proof: In order for $F = (\theta^{\text{dec}}_{h'h})$ to be a stochastic matrix, we need $\sum_{h \in \mathcal{H}'} \theta^{\text{dec}}_{h'h} = 1$ for all $h' \in \mathcal{H}'$. Fix an $h' \in \mathcal{H}'$. Using (11), we have

$$\sum_{h \in \mathcal{H}'} \theta^{\text{dec}}_{h'h} = \sum_{h \in \mathcal{H}} \theta^{\text{dec}}_{h'h} = \frac{\sum_{h \in \mathcal{H}} \sum_{l'a'} \pi^{\text{xdp}}_{\phi'} \Big(\sum_{la} p_{l'l} q_{a'a} z^{\text{xdp}}_{\phi'\phi} \Big)}{\sum_{l'a'} \pi^{\text{xdp}}_{\phi'}} = \frac{\sum_{l'a'} \pi^{\text{xdp}}_{\phi'} \Big(\sum_{\phi} p_{l'l} q_{a'a} z^{\text{xdp}}_{\phi'\phi} \Big)}{\sum_{l'a'} \pi^{\text{xdp}}_{\phi'}} = \frac{\sum_{la} \pi^{\text{xdp}}_{\phi}}{\sum_{la} \pi^{\text{xdp}}_{\phi}} = 1$$

Since this holds for all $h' \in \mathcal{H}'$, the lemma follows. ■

From Lemma 2, it follows that F can be thought of as the transition probability matrix for the Markov chain over the set $\mathcal{H}'$ of configuration states that results from the reconfiguration policy (11) of the decoupled control algorithm.

Lemma 3: For all $h \in \mathcal{H}$, the fraction of time spent in configuration h under the reconfiguration policy (11) of the decoupled control algorithm is equal to the fraction of time in configuration h under the relaxed MDP.

Proof: First, consider any state $h \in \mathcal{H}'$. Let π_h^F denote the steady state probability for state h in the Markov chain defined by transition probability matrix F. This is well-defined and exists according to Lemma 2. It is well known that these steady state probabilities can be obtained as the unique solution to the system of equations $\pi_h^F = \sum_{h' \in \mathcal{H}'} \pi_{h'}^F \theta_{h'h}^{\text{dec}}$ for all h. We now show that choosing $\pi_{h'}^F = \sum_{l'a'} \pi_{l'a'h'}^{\text{xdp}}$ for all $h' \in \mathcal{H}'$ satisfies this system of equations. Upon substituting $\pi_{h'}^F = \sum_{l'a'} \pi_{l'a'h'}^{\text{xdp}}$, the RHS $\sum_{h' \in \mathcal{H}'} \pi_{h'}^F \theta_{h'h}^{\text{dec}}$ becomes

$$\sum_{h' \in \mathcal{H}'} \pi_{h'}^F \theta_{h'h}^{\text{dec}} = \sum_{h' \in \mathcal{H}'} \Big(\sum_{l'a'} \pi_{l'a'h'}^{\text{xdp}} \Big) \theta_{h'h}^{\text{dec}} = \sum_{h' \in \mathcal{H}'} \Big(\sum_{l'a'} \pi_{l'a'h'}^{\text{xdp}} \Big) \frac{\sum_{l'a'} \pi_{l'a'h'}^{\text{xdp}} \Big(\sum_{la} p_{l'l} q_{a'a} z_{l'a'h',lah}^{\text{xdp}} \Big)}{\sum_{l'a'} \pi_{l'a'h'}^{\text{xdp}}}$$
$$= \sum_{h' \in \mathcal{H}'} \sum_{l'a'} \pi_{l'a'h'}^{\text{xdp}} \Big(\sum_{la} p_{l'l} q_{a'a} z_{l'a'h',lah}^{\text{xdp}} \Big) = \sum_{la} \sum_{l'a'} \sum_{h' \in \mathcal{H}'} \pi_{l'a'h'}^{\text{xdp}} p_{l'l} q_{a'a} z_{l'a'h',lah}^{\text{xdp}} = \sum_{la} \pi_{lah}^{\text{xdp}}$$

where we used (11) in the second step. Similarly, upon substituting $\pi_{h'}^F = \sum_{l'a'} \pi_{l'a'h'}^{\text{xdp}}$, the LHS $\pi_{h'}^F$ becomes: $\pi_{h'}^F = \sum_{l'a'} \pi_{l'a'h'}^{\text{xdp}} = \sum_{la} \pi_{lah}^{\text{xdp}}$. This implies that $\pi_h^F = \sum_{la} \pi_{lah}^{\text{xdp}}$ for all $h \in \mathcal{H}'$. This shows that the fraction of time spent in configuration h under the reconfiguration policy (11) of the decoupled control algorithm, i.e., π_h^F is equal to $\sum_{la} \pi_{lah}^{\text{xdp}}$ which is the fraction of time in configuration h under the relaxed MDP. For the case when $h \in \mathcal{H} \setminus \mathcal{H}'$, Lemma 3 follows by noting that the fraction of time spent in configuration h in both cases is 0. ■

Lemma 4: The time-average expected reconfiguration cost under the decoupled control algorithm is equal to the time-average expected reconfiguration cost of the relaxed MDP, i.e., w^*.

Proof: The time-average expected reconfiguration cost under policy (11) is $\sum_{h \in \mathcal{H}} \pi_h^F \sum_{h' \in \mathcal{H}} \theta_{hh'}^{\text{dec}} W_{hh'}$. Substituting $\pi_h^F = \sum_{la} \pi_{lah}^{\text{xdp}}$ (from Lemma 3) and using (11), we have:

$$\sum_{h \in \mathcal{H}} \pi_h^F \sum_{h' \in \mathcal{H}} \theta_{hh'}^{\text{dec}} W_{hh'} = \sum_{h \in \mathcal{H}} \Big(\sum_{la} \pi_{lah}^{\text{xdp}} \Big) \sum_{h' \in \mathcal{H}} \frac{\sum_{la} \pi_{lah}^{\text{xdp}} \Big(\sum_{l'a'} p_{ll'} q_{aa'} z_{lah,l'a'h'}^{\text{xdp}} \Big)}{\sum_{la} \pi_{lah}^{\text{xdp}}} W_{hh'}$$
$$= \sum_{h \in \mathcal{H}} \sum_{h' \in \mathcal{H}} \sum_{la} \pi_{lah}^{\text{xdp}} \Big(\sum_{l'a'} p_{ll'} q_{aa'} z_{lah,l'a'h'}^{\text{xdp}} \Big) W_{hh'} = \sum_{lah} \pi_{lah}^{\text{xdp}} \Big(\sum_{l'a'h'} p_{ll'} q_{aa'} z_{lah,l'a'h'}^{\text{xdp}} W_{hh'} \Big) = \sum_{lah} \pi_{lah}^{\text{xdp}} W_{\phi}^{\text{xdp}} = w^*$$

where the last step follows from (29) and (32). ■

Theorem 1 part 1 follows from Lemma 4.

Lemma 5: Under the routing policy defined by (12), the time-average transmission cost of the decoupled algorithm is equal to that under the relaxed MDP, i.e., c^*.

Proof: Let π_{la} denote the fraction of time the decoupled MDP is in state $(\boldsymbol{l}(t) = l, \boldsymbol{a}(t) = a)$. The time-average transmission cost under the routing policy defined by the probabilities in (12) is given by

$$\sum_{la} \pi_{la} \sum_{r \in \mathcal{R}(l,a)} \zeta_{la}^{\text{dec}}(r) \sum_{knm} r_{knm} c_{knm} = \sum_{la} \pi_{la} \sum_{r \in \mathcal{R}(l,a)} \frac{\sum_h \pi_{\phi}^{\text{xdp}} x_{\phi}^{\text{xdp}}(r) \big(\sum_{knm} r_{knm} c_{knm} \big)}{\sum_h \pi_{\phi}^{\text{xdp}}}$$
$$= \sum_{la} \pi_{la} \frac{\sum_h \pi_{\phi}^{\text{xdp}} \sum_{r \in \mathcal{R}(l,a)} x_{\phi}^{\text{xdp}}(r) \big(\sum_{knm} r_{knm} c_{knm} \big)}{\sum_h \pi_{\phi}^{\text{xdp}}} = \sum_{la} \pi_{la} \frac{\sum_h \pi_{\phi}^{\text{xdp}} C_{\phi}^{\text{xdp}}}{\sum_h \pi_{\phi}^{\text{xdp}}} = \sum_{la} \sum_h \pi_{\phi}^{\text{xdp}} C_{\phi}^{\text{xdp}} = c^*$$

where in the second step we sum over only those (l,a) for which $\pi_{la}>0$. In the second last step above, we used the fact that $\pi_{la}=\sum_{h\in\mathcal{H}}\pi_{\phi}^{\text{xdp}}$, i.e., the fraction of time the relaxed MDP is in state $(\boldsymbol{l}(t)=l,\boldsymbol{a}(t)=a)$ is equal to π_{la}. This is because these states evolve independent of the control actions of either control algorithms. ■

This shows Theorem 1 part 2.

Lemma 6: Under the back-end routing policy defined by (13), the time-average back-end transmission cost of the decoupled algorithm is equal to that under the relaxed MDP, i.e., e^*.

Proof: The time-average transmission cost under the back-end routing policy defined by (13) is given by

$$\sum_{\upsilon\in\mathcal{V}}\vartheta^{\text{dec}}(\upsilon)\sum_{km}\upsilon_{km}e_{km}=\sum_{\upsilon\in\mathcal{V}}\big(\sum_{\phi}\pi_{\phi}^{\text{xdp}}y_{\phi}^{\text{xdp}}(\upsilon)\big)\sum_{km}\upsilon_{km}e_{km}=\sum_{\phi}\pi_{\phi}^{\text{xdp}}\sum_{\upsilon\in\mathcal{V}}y_{\phi}^{\text{xdp}}(\upsilon)\sum_{km}\upsilon_{km}e_{km}=\sum_{\phi}\pi_{\phi}^{\text{xdp}}E_{\phi}^{\text{xdp}}=e^*$$

where we used (31) in the second last step. ■

This shows Theorem 1 part 3.

Finally, we show that time-average arrival and service rates are equal to those under the relaxed MDP.

Lemma 7: The time-average arrival and service rates for each queue $U_{km}(t)$ under the decoupled MDP are equal to those under the relaxed MDP.

Proof: The fraction of time a routing vector $r\in\mathcal{R}(l,a)$ is chosen under the relaxed MDP is given by $\sum_h\pi_{\phi}^{\text{xdp}}x_{\phi}^{\text{xdp}}(r)$. Under the decoupled MDP, this becomes

$$\pi_{la}\zeta_{la}^{\text{dec}}(r)=\pi_{la}\times\frac{\sum_h\pi_{\phi}^{\text{xdp}}x_{\phi}^{\text{xdp}}(r)}{\sum_h\pi_{\phi}^{\text{xdp}}}=\sum_h\pi_{\phi}^{\text{xdp}}x_{\phi}^{\text{xdp}}(r)$$

This shows that the time-average arrival rate to any queue is the same under both algorithms. A similar result holds for average local service rates since these are only a function of the fraction of time spent in any configuration h which is the same under both MDPs. Finally, the fraction of time a back-end routing vector $\upsilon\in\mathcal{V}$ is chosen under the relaxed MDP is given by $\sum_{\phi}\pi_{\phi}^{\text{xdp}}y_{\phi}^{\text{xdp}}(\upsilon)$. Under the decoupled MDP, this is equal to $\vartheta^{\text{dec}}(\upsilon)=\sum_{\phi}\pi_{\phi}^{\text{xdp}}y_{\phi}^{\text{xdp}}(\upsilon)$ by definition (13). ■

Theorem 1 part 4 follows by noting that the time-average arrival and service rates to each queue $U_{km}(t)$ under the decoupled MDP satisfy (10).

APPENDIX C
PROOF OF THEOREM 3

Let $\boldsymbol{Q}(t)=(\boldsymbol{U}(t),\boldsymbol{Z}(t))$ denote the collection of queue backlogs in slot t. Define $L(\boldsymbol{Q}(t))\triangleq\frac{1}{2}\sum_{km}\big(U_{km}^2(t)+Z_{km}^2(t)\big)$ as a Lyapunov function of $\boldsymbol{Q}(t)$. Using (5) and (15), we can bound the one-slot difference $L(\boldsymbol{Q}(t+1))-L(\boldsymbol{Q}(t))$ as

$$L(\boldsymbol{Q}(t+1))-L(\boldsymbol{Q}(t))\leq\sum_{km}\Big(B_{km}^u+B_{km}^z-U_{km}(t)\big(\mu_{km}(t)+\upsilon_{km}(t)-R_{km}(t)\big)-Z_{km}(t)\big(\mu_{km}(t)+\upsilon_{km}(t)-\sigma_{km}\big)\Big)$$

where $B_{km}^u=((\mu_{km}^{\max}+\upsilon_{km}^{\max})^2+(R_{km}^{\max})^2)/2$ and $B_{km}^z=((\mu_{km}^{\max}+\upsilon_{km}^{\max})^2+\sigma_{km}^2)/2$. Note that at any renewal time t_f, the application configuration $h(t_f)=h_0$. For ease of notation, we denote the collection of all queue backlogs, user locations, and arrival vectors at the start of renewal frame f by $\boldsymbol{\Psi}(t_f)$. Define the *frame-based conditional Lyapunov drift* $\Delta(t_f)$ as the expected change in the value of $L(\boldsymbol{Q}(t))$ over the f^{th} frame given initial state $\boldsymbol{\Psi}(t_f)$, i.e., $\Delta(t_f)\triangleq\mathbb{E}\left\{L(\boldsymbol{Q}(t_{f+1}))-L(\boldsymbol{Q}(t_f))|\boldsymbol{\Psi}(t_f)\right\}$. In the following, we use the shorthand notation $\mathbb{E}_c\left\{.\right\}$ to represent the conditional expectation $\mathbb{E}\left\{.|\boldsymbol{\Psi}(t_f)\right\}$.

Following the methodology in [8], [17] and using the one-slot difference bound along with delayed queueing equations, we can obtain the following upper bound on the frame-based conditional Lyapunov drift plus V times the conditional expected transmission plus reconfiguration cost, i.e., $\Delta(t_f) + V\mathbb{E}_c\left\{\sum_{\tau=t_f}^{t_{f+1}-1} C(\tau)+W(\tau)+E(\tau)\right\}$ in terms of the control decisions under any algorithm (see Appendix D).

$$\sum_{km}\mathbb{E}_c\left\{X_{km}(T_f)\right\} - \sum_{km}\big(U_{km}(t_f)+Z_{km}(t_f)\big)\mathbb{E}_c\left\{\sum_{\tau=t_f}^{t_{f+1}-1}\mu_{km}(\tau)\right\} - \sum_{km}\mathbb{E}_c\left\{\sum_{\tau=t_f}^{t_{f+1}-1}\big(U_{km}(\tau)+Z_{km}(\tau)\big)\upsilon_{km}(\tau)\right\}$$
$$+\sum_{km}\mathbb{E}_c\left\{\sum_{\tau=t_f}^{t_{f+1}-1}Z_{km}(\tau)\sigma_{km}\right\}+\sum_{km}\mathbb{E}_c\left\{\sum_{\tau=t_f}^{t_{f+1}-1}U_{km}(\tau)R_{km}(\tau)\right\}+V\mathbb{E}_c\left\{\sum_{\tau=t_f}^{t_{f+1}-1}C(\tau)+W(\tau)+E(\tau)\right\} \quad (33)$$

where $X_{km}(T_f)\triangleq(B^u_{km}+B^z_{km})T_f + T_f(T_f-1)(\mu^{\max}_{km}+\upsilon^{\max}_{km})\mu^{\max}_{km}$. Given any location and arrival states and queue backlogs, the routing decisions under the online control algorithm minimize $VC(t)+\sum_{km}U_{km}(t)R_{km}(t)$ in every slot over all other algorithms, including the backlog independent stationary algorithm. Similarly, the back-end routing decisions under the online control algorithm minimize $VE(t)-\sum_{km}(U_{km}(t)+Z_{km}(t))\upsilon_{km}(t)$ in every slot over all other algorithms, including the stationary algorithm. Denoting these decisions under the stationary algorithm by $R^{\text{stat}}_{km}(t)$ and $\upsilon^{\text{stat}}_{km}(t)$ and the resulting costs by $C^{\text{stat}}(t)$ and $E^{\text{stat}}(t)$, it follows that we can upper bound the value of $\Delta(t_f)+V\mathbb{E}_c\left\{\sum_{\tau=t_f}^{t_{f+1}-1}C(\tau)+W(\tau)+E(\tau)\right\}$ under the online control algorithm by

$$\sum_{km}\mathbb{E}_c\left\{X_{km}(T_f)\right\} - \sum_{km}\big(U_{km}(t_f)+Z_{km}(t_f)\big)\mathbb{E}_c\left\{\sum_{\tau=t_f}^{t_{f+1}-1}\mu_{km}(\tau)\right\} - \sum_{km}\mathbb{E}_c\left\{\sum_{\tau=t_f}^{t_{f+1}-1}\big(U_{km}(\tau)+Z_{km}(\tau)\big)\upsilon^{\text{stat}}_{km}(\tau)\right\}+$$
$$\sum_{km}\mathbb{E}_c\left\{\sum_{\tau=t_f}^{t_{f+1}-1}Z_{km}(\tau)\sigma_{km}\right\}+\sum_{km}\mathbb{E}_c\left\{\sum_{\tau=t_f}^{t_{f+1}-1}U_{km}(\tau)R^{\text{stat}}_{km}(\tau)\right\}+V\mathbb{E}_c\left\{\sum_{\tau=t_f}^{t_{f+1}-1}C^{\text{stat}}(\tau)+W(\tau)+E^{\text{stat}}(\tau)\right\} \quad (34)$$

Next, we use the following bounds obtained using delayed queue equations (details in Appendix E):

$$-\sum_{km}\mathbb{E}_c\left\{\sum_{\tau=t_f}^{t_{f+1}-1}U_{km}(\tau)\upsilon^{\text{stat}}_{km}(\tau)\right\} \le -\sum_{km}U_{km}(t_f)\upsilon^{\text{stat}}_{km}\mathbb{E}_c\left\{T_f\right\}+\sum_{km}\mathbb{E}_c\left\{\frac{T_f(T_f-1)}{2}(\mu^{\max}_{km}+\upsilon^{\max}_{km})\upsilon^{\max}_{km}\right\} \quad (35)$$

$$-\sum_{km}\mathbb{E}_c\left\{\sum_{\tau=t_f}^{t_{f+1}-1}Z_{km}(\tau)\upsilon^{\text{stat}}_{km}(\tau)\right\} \le -\sum_{km}Z_{km}(t_f)\upsilon^{\text{stat}}_{km}\mathbb{E}_c\left\{T_f\right\}+\sum_{km}\mathbb{E}_c\left\{\frac{T_f(T_f-1)}{2}(\mu^{\max}_{km}+\upsilon^{\max}_{km})\upsilon^{\max}_{km}\right\} \quad (36)$$

$$\sum_{km}\mathbb{E}_c\left\{\sum_{\tau=t_f}^{t_{f+1}-1}Z_{km}(\tau)\sigma_{km}\right\} \le \sum_{km}Z_{km}(t_f)\sigma_{km}\mathbb{E}_c\left\{T_f\right\}+\sum_{km}\mathbb{E}_c\left\{\frac{T_f(T_f-1)}{2}\sigma^2_{km}\right\} \quad (37)$$

$$\sum_{km}\mathbb{E}_c\left\{\sum_{\tau=t_f}^{t_{f+1}-1}U_{km}(\tau)R^{\text{stat}}_{km}(\tau)\right\} \le \sum_{km}U_{km}(t_f)(R^{\text{stat}}_{km}+\alpha\gamma^{\delta})\mathbb{E}_c\left\{T_f\right\}+\sum_{km}\mathbb{E}_c\left\{\delta T_f(R^{\max}_{km})^2+\frac{T_f(T_f-1)}{2}(R^{\max}_{km})^2\right\}$$
$$+\sum_{km}\mathbb{E}_c\left\{x_{km}(t_f,t_{f+1})\right\} \quad (38)$$

$$\mathbb{E}_c\left\{\sum_{\tau=t_f}^{t_{f+1}-1}C^{\text{stat}}(\tau)\right\} \le (c^*+\beta\gamma^{\delta})\mathbb{E}_c\left\{T_f\right\}+\mathbb{E}_c\left\{y(t_f,t_{f+1})\right\} \quad (39)$$

$$\mathbb{E}_c\left\{\sum_{\tau=t_f}^{t_{f+1}-1}E^{\text{stat}}(\tau)\right\} \le (e^*+\phi(\boldsymbol{\sigma}))\mathbb{E}_c\left\{T_f\right\} \quad (40)$$

where $x_{km}(t_f, t_{f+1})$ and $y(t_f, t_{f+1})$ are defined as

$$x_{km}(t_f,t_{f+1}) = \sum_{\tau=t_f}^{t_f+\delta-1} U_{km}(\tau)R_{km}^{\text{stat}}(\tau) - \sum_{\tau=t_{f+1}}^{t_{f+1}+\delta-1} U_{km}(\tau)R_{km}^{\text{stat}}(\tau), \;\; y(t_f,t_{f+1}) = \sum_{\tau=t_f}^{t_f+\delta-1} C^{\text{stat}}(\tau) - \sum_{\tau=t_{f+1}}^{t_{f+1}+\delta-1} C^{\text{stat}}(\tau)$$

and $\alpha \geq 0, \beta \geq 0, 0 < \gamma < 1$ are constants that characterize how fast the Markov chain defined by the user location and request arrival processes mixes to its steady-state distribution and $\delta \geq 0$ is an integer whose value will be specified in the following. Substituting these, we get

$$\begin{aligned}
&\Delta(t_f) + V\mathbb{E}_c\left\{\sum_{\tau=t_f}^{t_{f+1}-1} C(\tau) + W(\tau) + E(\tau)\right\} \leq \sum_{km}\mathbb{E}_c\left\{X'_{km}(T_f)\right\} - \sum_{km}\big(U_{km}(t_f) + Z_{km}(t_f)\big)\mathbb{E}_c\left\{\sum_{\tau=t_f}^{t_{f+1}-1}\mu_{km}(\tau)\right\} \\
&- \sum_{km}\big(U_{km}(t_f) + Z_{km}(t_f)\big)\upsilon_{km}^{\text{stat}}\mathbb{E}_c\left\{T_f\right\} + \sum_{km} Z_{km}(t_f)\sigma_{km}\mathbb{E}_c\left\{T_f\right\} + \sum_{km} U_{km}(t_f)(R_{km}^{\text{stat}} + \alpha\gamma^{\delta})\mathbb{E}_c\left\{T_f\right\} \\
&+ V(c^* + e^* + \beta\gamma^{\delta} + \phi(\boldsymbol{\sigma}))\mathbb{E}_c\left\{T_f\right\} + V\mathbb{E}_c\left\{\sum_{\tau=t_f}^{t_{f+1}-1} W(\tau)\right\} + \sum_{km}\mathbb{E}_c\left\{x_{km}(t_f,t_{f+1})\right\} + V\mathbb{E}_c\left\{y(t_f,t_{f+1})\right\} \qquad (41)
\end{aligned}$$

where $X'_{km}(T_f) = \big(T_f + T_f(T_f-1)\big)J_{km}/2 + \delta T_f(R_{km}^{\max})^2$. Next, note that under the application reconfiguration and servicing decisions of the online control algorithm (20), the following holds for any $\boldsymbol{\Psi}(t_f)$.

$$\begin{aligned}
&\sum_{km}\mathbb{E}_c\left\{X'_{km}(T_f)\right\} - \sum_{km}\big(U_{km}(t_f) + Z_{km}(t_f)\big)\mathbb{E}_c\left\{\sum_{\tau=t_f}^{t_{f+1}-1}\mu_{km}(\tau)\right\} + V\mathbb{E}_c\left\{\sum_{\tau=t_f}^{t_{f+1}-1} W(\tau)\right\} \\
&\leq \eta\sum_{km}\mathbb{E}_c\left\{X'_{km}(T_f^{\text{stat}})\right\} - \eta\sum_{km}\big(U_{km}(t_f) + Z_{km}(t_f)\big)\mathbb{E}_c\left\{\sum_{\tau=t_f}^{t_f+T_f^{\text{stat}}-1}\mu_{km}^{\text{stat}}(\tau)\right\} + \eta V\mathbb{E}_c\left\{\sum_{\tau=t_f}^{t_f+T_f^{\text{stat}}-1} W^{\text{stat}}(\tau)\right\} \qquad (42)
\end{aligned}$$

where $\eta = \frac{\mathbb{E}_c\{T_f\}}{\mathbb{E}_c\{T_f^{\text{stat}}\}}$. Here, $\mu_{km}^{\text{stat}}(\tau), W^{\text{stat}}(\tau), T_f^{\text{stat}}$ represent the control decisions and the resulting reconfiguration cost and frame length under the backlog independent stationary algorithm defined earlier. This follows by noting that the online control algorithm is designed to *minimize the ratio* of the left hand side to the frame length over all possible algorithms. Next, note that the reconfiguration policy of the stationary algorithm is dependent only on the configuration state and is independent of all states in $\boldsymbol{\Psi}(t_f)$. Thus, the conditional expectation can be replaced by regular expectation and the relationships in (14) can be used since the reconfiguration policies of the decoupled control algorithm and the stationary algorithm are identical. Plugging these, the above can be rewritten as

$$\begin{aligned}
&\sum_{km}\mathbb{E}_c\left\{X'_{km}(T_f)\right\} - \sum_{km}\big(U_{km}(t_f) + Z_{km}(t_f)\big)\mathbb{E}_c\left\{\sum_{\tau=t_f}^{t_{f+1}-1}\mu_{km}(\tau)\right\} + V\mathbb{E}_c\left\{\sum_{\tau=t_f}^{t_{f+1}-1} W(\tau)\right\} \\
&\leq \eta\sum_{km}\mathbb{E}_c\left\{X'_{km}(T_f^{\text{stat}})\right\} - \sum_{km}\big(U_{km}(t_f) + Z_{km}(t_f)\big)\mu_{km}^{\text{stat}}\mathbb{E}_c\left\{T_f\right\} + Vw^*\mathbb{E}_c\left\{T_f\right\} \qquad (43)
\end{aligned}$$

Using this, the right hand side of (41) can be bounded as

$$\begin{aligned}
&\eta\sum_{km}\mathbb{E}_c\left\{X'_{km}(T_f^{\text{stat}})\right\} - \sum_{km} U_{km}(t_f)(\mu_{km}^{\text{stat}} + \upsilon_{km}^{\text{stat}} - R_{km}^{\text{stat}} - \alpha\gamma^{\delta})\mathbb{E}_c\left\{T_f\right\} - \sum_{km} Z_{km}(t_f)(\mu_{km}^{\text{stat}} + \upsilon_{km}^{\text{stat}} - \sigma_{km})\mathbb{E}_c\left\{T_f\right\} \\
&+ V(c^* + e^* + w^* + \beta\gamma^{\delta} + \phi(\boldsymbol{\sigma}))\mathbb{E}_c\left\{T_f\right\} + \sum_{km}\mathbb{E}_c\left\{x_{km}(t_f,t_{f+1})\right\} + V\mathbb{E}_c\left\{y(t_f,t_{f+1})\right\} \qquad (44)
\end{aligned}$$

Note that the frame length T_f^{stat} under the stationary algorithm is independent of the initial state $\boldsymbol{\Psi}(t_f)$. Further, the recon-

figuration policy for the stationary algorithm and the decoupled algorithm are identical. Thus, the conditional expectation can been removed and the first term can be rewritten as

$$\eta \sum_{km} \mathbb{E}_c \left\{ X'_{km}(T_f^{\text{stat}}) \right\} = \sum_{km} \frac{\mathbb{E}\left\{ X'_{km}(T_f^{\text{stat}}) \right\}}{\mathbb{E}\left\{ T_f^{\text{stat}} \right\}} \mathbb{E}_c \left\{ T_f \right\} = \sum_{km} \frac{\mathbb{E}\left\{ X'_{km}(T_{h_0}^{\text{dec}}) \right\}}{\mathbb{E}\left\{ T_{h_0}^{\text{dec}} \right\}} \mathbb{E}_c \left\{ T_f \right\}$$
$$= \sum_{km} \left((1 + \Upsilon_{h_0}^{\text{dec}}) J_{km}/2 + \delta (R_{km}^{\max})^2 \right) \mathbb{E}_c \left\{ T_f \right\} \tag{45}$$

where $\Upsilon_{h_0}^{\text{dec}} = \frac{\mathbb{E}\left\{ T_{h_0}^{\text{dec}} (T_{h_0}^{\text{dec}} - 1) \right\}}{\mathbb{E}\left\{ T_{h_0}^{\text{dec}} \right\}}$. Denoting $B_{km} = (1 + \Upsilon_{h_0}^{\text{dec}}) J_{km}/2 + \delta (R_{km}^{\max})^2$, (44) can be rewritten as

$$\sum_{km} B_{km} \mathbb{E}_c \left\{ T_f \right\} - \sum_{km} U_{km}(t_f) (\mu_{km}^{\text{stat}} + \upsilon_{km}^{\text{stat}} - R_{km}^{\text{stat}} - \alpha \gamma^{\delta}) \mathbb{E}_c \left\{ T_f \right\} - \sum_{km} Z_{km}(t_f) (\mu_{km}^{\text{stat}} + \upsilon_{km}^{\text{stat}} - \sigma_{km}) \mathbb{E}_c \left\{ T_f \right\}$$
$$+ V(c^* + e^* + w^* + \beta \gamma^{\delta} + \phi(\boldsymbol{\sigma})) \mathbb{E}_c \left\{ T_f \right\} + \sum_{km} \mathbb{E}_c \left\{ x_{km}(t_f, t_{f+1}) \right\} + V \mathbb{E}_c \left\{ y(t_f, t_{f+1}) \right\} \tag{46}$$

Note that under the stationary algorithm, if $R_{km}^{\text{stat}} > 0$, then $\mu_{km}^{\text{stat}} + \upsilon_{km}^{\text{stat}} - R_{km}^{\text{stat}} \geq \mu_{km}^{\text{dec}} + \upsilon_{km}^{\text{dec}} - R_{km}^{\text{dec}} \geq \epsilon > 0$. This follows from Theorem 1 part 4 and the fact that $\upsilon_{km}^{\text{stat}} \geq \upsilon_{km}^{\text{dec}}$. Thus, by choosing $\delta \geq \log \frac{2\alpha/\epsilon}{1/\gamma}$, we have that $\mu_{km}^{\text{stat}} - R_{km}^{\text{stat}} - \alpha \gamma^{\delta} \geq \epsilon/2$ if $R_{km}^{\text{stat}} > 0$. For those k, m for which $\mu_{km}^{\text{stat}} = R_{km}^{\text{stat}} = 0$, the α can be set to 0. Also, the stationary algorithm is designed such that $\mu_{km}^{\text{stat}} + \upsilon_{km}^{\text{stat}} \geq \sigma_{km}$. Thus, we have

$$\Delta(t_f) + V \mathbb{E}_c \left\{ \sum_{\tau=t_f}^{t_{f+1}-1} C(\tau) + W(\tau) + E(\tau) \right\} \leq \sum_{km} B_{km} \mathbb{E}_c \left\{ T_f \right\} + V(c^* + e^* + w^* + \beta \gamma^{\delta} + \phi(\boldsymbol{\sigma})) \mathbb{E}_c \left\{ T_f \right\}$$
$$+ \sum_{km} \mathbb{E}_c \left\{ x_{km}(t_f, t_{f+1}) \right\} + V \mathbb{E}_c \left\{ y(t_f, t_{f+1}) \right\} \tag{47}$$

Taking the expectation of both sides and summing over $f \in \{0, 1, 2, ..., F-1\}$ yields

$$\mathbb{E}\left\{ \Phi(\boldsymbol{U}(t_F)) \right\} + V \sum_{f=0}^{F-1} \mathbb{E}\left\{ \sum_{\tau=t_f}^{t_{f+1}-1} C(\tau) + W(\tau) + E(\tau) \right\} \leq \sum_{km} \sum_{f=0}^{F-1} B_{km} \mathbb{E}\left\{ T_f \right\}$$
$$+ V(c^* + e^* + w^* + \beta \gamma^{\delta} + \phi(\boldsymbol{\sigma})) \sum_{f=0}^{F-1} \mathbb{E}\left\{ T_f \right\} + \sum_{km} \sum_{f=0}^{F-1} \mathbb{E}\left\{ x_{km}(t_f, t_{f+1}) \right\} + V \sum_{f=0}^{F-1} \mathbb{E}\left\{ y(t_f, t_{f+1}) \right\} \tag{48}$$

This can be rearranged to get

$$\sum_{f=0}^{F-1} \mathbb{E}\left\{ \sum_{\tau=t_f}^{t_{f+1}-1} C(\tau) + W(\tau) + E(\tau) \right\} \leq (c^* + e^* + w^* + \beta \gamma^{\delta} + \phi(\boldsymbol{\sigma})) \sum_{f=0}^{F-1} \mathbb{E}\left\{ T_f \right\} + \sum_{km} \frac{B_{km}}{V} \sum_{f=0}^{F-1} \mathbb{E}\left\{ T_f \right\}$$
$$+ \sum_{km} \sum_{f=0}^{F-1} \frac{\mathbb{E}\left\{ x_{km}(t_f, t_{f+1}) \right\}}{V} + \sum_{f=0}^{F-1} \mathbb{E}\left\{ y(t_f, t_{f+1}) \right\}$$

The last two terms vanish when we divide both sides by $\sum_{f=0}^{F-1} \mathbb{E}\left\{ T_f \right\}$ and take limit $F \to \infty$. The bound in Theorem 3 part 1 follows by choosing $\delta \geq \frac{\log V\beta}{\log 1/\gamma}$. It can be seen that this term is $O(\log V)$.

Part 2 of Theorem 3 follows by a direct application of Lemma 5.5 from [8].

Finally, part 3 of Theorem 3 is obtained by performing a frame-based conditional Lyapunov drift analysis for the approximate algorithm similar to the analysis above while using the relations (24), (25) and is omitted for brevity.

APPENDIX D
FRAME-BASED LYAPUNOV DRIFT BOUND

The one-slot difference $L(\boldsymbol{Q}(\tau+1)) - L(\boldsymbol{Q}(\tau))$ is upper bounded by

$$\sum_{km}\Big(B_{km}^u + B_{km}^z - U_{km}(t)\big(\mu_{km}(t) + \upsilon_{km}(t) - R_{km}(t)\big) - Z_{km}(t)\big(\mu_{km}(t) + \upsilon_{km}(t) - \sigma_{km}\big)\Big)$$

Summing over $t \in \{t_f, \ldots, t_{f+1}-1\}$ yields the following bound on the frame-level difference $L(\boldsymbol{Q}(t_{f+1})) - L(\boldsymbol{Q}(t_f))$:

$$\sum_{km}\Big((B_{km}^u + B_{km}^z)T_f - \sum_{\tau=t_f}^{t_{f+1}-1}(U_{km}(\tau)+Z_{km}(\tau))\mu_{km}(\tau) + \sum_{\tau=t_f}^{t_{f+1}-1}U_{km}(\tau)R_{km}(\tau) - \sum_{\tau=t_f}^{t_{f+1}-1}(U_{km}(\tau)+Z_{km}(\tau))\upsilon_{km}(\tau) + \sum_{\tau=t_f}^{t_{f+1}-1}Z_{km}(\tau)\sigma_{km}\Big) \tag{49}$$

From (5), it follows that $U_{km}(\tau) \geq U_{km}(t_f) - (\tau - t_f)(\mu_{km}^{\max} + \upsilon_{km}^{\max})$ for all $\tau \geq t_f$. Similarly from (15), it follows that $Z_{km}(\tau) \geq Z_{km}(t_f) - (\tau - t_f)(\mu_{km}^{\max} + \upsilon_{km}^{\max})$ for all $\tau \geq t_f$. Using this, the second term above can be lower bounded as

$$\begin{aligned}\sum_{\tau=t_f}^{t_{f+1}-1}(U_{km}(\tau)+Z_{km}(\tau))\mu_{km}(\tau) &\geq \sum_{\tau=t_f}^{t_{f+1}-1}(U_{km}(t_f)+Z_{km}(t_f))\mu_{km}(\tau) - \sum_{\tau=t_f}^{t_{f+1}-1}(\tau-t_f)(\mu_{km}^{\max}+\upsilon_{km}^{\max})\mu_{km}(\tau)\\ &\geq \sum_{\tau=t_f}^{t_{f+1}-1}(U_{km}(t_f)+Z_{km}(t_f))\mu_{km}(\tau) - T_f(T_f-1)(\mu_{km}^{\max}+\upsilon_{km}^{\max})\mu_{km}^{\max}\end{aligned}$$

(33) follows by using the above in (49), adding the $V\sum_{\tau=t_f}^{t_{f+1}-1}(C(\tau)+W(\tau)+E(\tau))$ term to both sides, and taking conditional expectation given $\boldsymbol{\Psi}(t_f)$.

APPENDIX E
DELAYED BACKLOG BOUNDS

We first show (35). From (5), it follows that $U_{km}(\tau) \geq U_{km}(t_f) - (\tau - t_f)(\mu_{km}^{\max} + \upsilon_{km}^{\max})$ for all $\tau \geq t_f$. Using this,

$$\begin{aligned}-\sum_{\tau=t_f}^{t_{f+1}-1}U_{km}(\tau)\upsilon_{km}^{\text{stat}}(\tau) &\leq -\sum_{\tau=t_f}^{t_{f+1}-1}U_{km}(t_f)\upsilon_{km}^{\text{stat}}(\tau) + \sum_{\tau=t_f}^{t_{f+1}-1}(\tau-t_f)(\mu_{km}^{\max}+\upsilon_{km}^{\max})\upsilon_{km}^{\text{stat}}(\tau)\\ &\leq -\sum_{\tau=t_f}^{t_{f+1}-1}U_{km}(t_f)\upsilon_{km}^{\text{stat}}(\tau) + \frac{T_f(T_f-1)}{2}(\mu_{km}^{\max}+\upsilon_{km}^{\max})\upsilon_{km}^{\max}\end{aligned} \tag{50}$$

The bound in (35) is obtained by summing (50) over all k, m, taking the conditional expectation of both sides and using the fact that the back-end routing decisions under the stationary algorithm are i.i.d. every slot. The bounds (36) and (37) can be shown using a similar procedure and we omit the details for brevity.

Next, we show (38). For any integer $\delta \geq 0$, we can express $\sum_{\tau=t_f}^{t_{f+1}-1}U_{km}(\tau)R_{km}^{\text{stat}}(\tau)$ in terms of δ-shifted terms as

$$\begin{aligned}\sum_{\tau=t_f}^{t_{f+1}-1}U_{km}(\tau)R_{km}^{\text{stat}}(\tau) &= \sum_{\tau=t_f}^{t_f+\delta-1}U_{km}(\tau)R_{km}^{\text{stat}}(\tau) + \sum_{\tau=t_f+\delta}^{t_{f+1}+\delta-1}U_{km}(\tau)R_{km}^{\text{stat}}(\tau) - \sum_{\tau=t_{f+1}}^{t_{f+1}+\delta-1}U_{km}(\tau)R_{km}^{\text{stat}}(\tau)\\ &= x_{km}(t_f,t_{f+1}) + \sum_{\tau=t_f+\delta}^{t_{f+1}+\delta-1}U_{km}(\tau)R_{km}^{\text{stat}}(\tau)\end{aligned} \tag{51}$$

From (5), it follows that $U_{km}(\tau) \leq U_{km}(t_f) + (\tau - t_f)R_{km}^{\max}$ for all $\tau \geq t_f$. Using this, the second term above can be upper bounded as

$$\sum_{\tau=t_f+\delta}^{t_{f+1}+\delta-1} U_{km}(t_f)R_{km}^{\text{stat}}(\tau) + \sum_{\tau=t_f+\delta}^{t_{f+1}+\delta-1} (\tau - t_f)R_{km}^{\max}R_{km}^{\text{stat}}(\tau) \leq \sum_{\tau=t_f+\delta}^{t_{f+1}+\delta-1} U_{km}(t_f)R_{km}^{\text{stat}}(\tau) + \Big(\delta T_f + \frac{T_f(T_f-1)}{2}\Big)(R_{km}^{\max})^2$$

Now consider the conditional expectation of the first term on the RHS above, i.e., $\sum_{\tau=t_f+\delta}^{t_{f+1}+\delta-1} U_{km}(t_f)R_{km}^{\text{stat}}(\tau)$. We have for any integer $\delta \geq 0$

$$\mathbb{E}_c\left\{\sum_{\tau=t_f+\delta}^{t_{f+1}+\delta-1} U_{km}(t_f)R_{km}^{\text{stat}}(\tau)\right\} = U_{km}(t_f)\mathbb{E}_c\left\{\sum_{\tau=t_f+\delta}^{t_{f+1}+\delta-1} R_{km}^{\text{stat}}(\tau)\right\} \leq U_{km}(t_f)(R_{km}^{\text{stat}} + \alpha\gamma^{\delta})\mathbb{E}_c\left\{T_f\right\} \tag{52}$$

where the last step follows from the application of Lemma 8 (below) and using the fact that the control decisions $R_{km}^{\text{stat}}(\tau)$ under the routing policy of the stationary algorithm are functions purely of the Markov chain defined by the user location and request arrival processes. (38) is obtained by summing (51) over all k, m, taking the conditional expectation, and applying the bound in (52). A similar approach can be used to show (39). Finally, (40) follows by noting that the back-end routing decisions under the stationary algorithm are taken in an i.i.d. manner every slot and the resulting time-average back-end routing cost is at most $e^* + \phi(\boldsymbol{\sigma})$.

Lemma 8: (Markov chain Convergence) [22] Let $Z(t)$ be a finite state, discrete time ergodic Markov chain. Let $\mathcal{S}$ denote its state space and let $\{\pi_s\}_{s\in\mathcal{S}}$ be the steady state probability distribution. Let $f(Z(t))$ be a positive random function of $Z(t)$. Define $\overline{f} = \sum_{j\in\mathcal{S}} \pi_j m_j$ where $m_j \triangleq \mathbb{E}\left\{f(Z(t))|Z(t)=j\right\}$. Then there exist constants α, γ such that for all integers $d \geq 0$, we have

$$\mathbb{E}\left\{f(Z(t))|Z(t-d)=i\right\} \leq \overline{f} + sm_{max}\alpha\gamma^d$$

where $m_{max} \triangleq \max_{j\in\mathcal{S}} m_j$ and $s = \text{card}\{\mathcal{S}\}$.