Dynamic Service Migration and Workload Scheduling in Micro-Clouds

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

Abstract—Mobile Micro-Clouds provide a promising approach to significantly improve tactical network operations 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 work, 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 demand of real-time situational awareness applications running on mobile devices carried by soldiers is putting a significant burden on the capacity of mobile 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 wireless 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 mobile micro-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 micro-clouds is service migration: Should a service currently running in one of the micro-clouds be migrated as the user locations change, and if yes, where? This question stems from the basic tradeoff 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 micro-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 micro-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. PROBLEM FORMULATION

We consider an micro-cloud system comprised of $M$ distributed micro-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.

---

*R. Urgaonkar and T. He are with the IBM T. J. Watson Research Center, Yorktown Heights, NY, USA. Emails: {urgaon, 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. He was previously with the IBM T. J. Watson Research Center, Yorktown Heights, NY, 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.
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 \( I(t) \). We assume that \( I(t) \) takes values from a finite (but potentially arbitrarily large) set \( \mathcal{L} \). Further, \( I(t) \) is assumed to evolve according to an ergodic discrete time Markov chain (DTMC) over the states in \( \mathcal{L} \) with transition probabilities denoted by \( p_{lt} \) for all \( l, t \in \mathcal{L} \).

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 \( \mathbf{a}(t) \). Similar to \( I(t) \), we assume that \( \mathbf{a}(t) \) 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 \( \mathbf{a}(t) \) is also assumed to evolve according to an ergodic DTMC over the states in \( \mathcal{A} \) with transition probabilities \( p_{aa'} \) for all \( a, a' \in \mathcal{A} \). All application requests generated at each user are routed to a selected subset of the micro-clouds for servicing. These routing decisions incur transmission costs and are subject to certain constraints as discussed below.

User-to-Micro-Cloud Request Routing: Let \( r_{kmn}(t) \) denote the number of application \( k \) requests from user \( n \) that are routed to micro-cloud \( m \) in slot \( t \) and let \( r(t) \) denote the collection \( \{r_{kmn}(t)\} \) for all \( k, n, m \). Routing of these requests incurs a transmission cost of \( r_{kmn}(t)c_{kmn}(t) \) where \( c_{kmn}(t) \) 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 micro-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., \( C(t) = \sum_{kmn} r_{kmn}(t)c_{kmn}(t) \). For each \( (k, m) \), we denote by \( R_{km}(t) \) the total number of application \( k \) requests received by micro-cloud \( m \) in slot \( t \), i.e., \( R_{km}(t) = \sum_{n=1}^{N} r_{kmn}(t) \). We assume that the maximum number of requests for an application \( k \) that can be routed to micro-cloud \( m \) in any slot is upper bounded by \( R_{km}^{\max} \). Given these assumptions, the routing decisions \( r(t) \) are subject to the following constraints

\[
A_{kn}(t) = \sum_{m=1}^{M} r_{kmn}(t) \quad \forall k, n \tag{1}
\]

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

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 micro-clouds where requests from user \( n \) can be routed given its location \( L_n(t) \). Given \( I(t) = l, a(t) = a \), 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 Micro-Clouds: For all \( k, m \), define application placement variables \( H_{km}(t) \) as

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

The collection \( \{H_{km}(t)\} \) is denoted by the vector \( \mathbf{h}(t) \). This defines the application configuration of the micro-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 micro-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 \( \varphi_{km}() \) of the vector \( \mathbf{h}(t) \), i.e., \( \mu_{km}(t) = \varphi_{km}(\mathbf{h}(t)) \). 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 \( \mathbf{h}(t) \) and use \( \mu_{km}(t) \) to mean \( \varphi_{km}(\mathbf{h}(t)) \). Further, we assume that there exist finite constants \( \mu_{km}^{\max} \) such that \( \mu_{km}(t) \leq \mu_{km}^{\max} \) for all \( t \).

An micro-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 micro-cloud. We say that an application configuration \( \mathbf{h}(t) \) 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 \( \mathbf{a}, \mathbf{b} \in \mathcal{H} \), the cost of switching from \( \mathbf{a} \) to \( \mathbf{b} \) is denoted by \( W_{ab} \). For simplicity, we assume that it is possible to switch between any two configurations \( \mathbf{a}, \mathbf{b} \in \mathcal{H} \) and that \( W_{ab} \) is upper bounded by a finite constant \( W_{\max} \). We further assume, without loss of generality, that

\[
W_{ab} \leq W_{ac} + W_{cb} \quad \forall \mathbf{a}, \mathbf{b}, \mathbf{c} \in \mathcal{H}. \tag{4}
\]

The last assumption is valid if \( W_{ab} \) is the minimum cost required to switch from \( \mathbf{a} \) to \( \mathbf{b} \). This is because if \( W_{ab} > W_{ac} + W_{cb} \), then we could carry out the reconfiguration from \( \mathbf{a} \) to \( \mathbf{b} \) by switching from \( \mathbf{a} \) to \( \mathbf{c} \) and then to \( \mathbf{b} \), 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 Micro-Clouds: As illustrated in Fig.
1, every micro-cloud $m$ maintains a request queue $U_{km}(t)$ per application $k$ that buffers application $k$ requests from all users that are routed to micro-cloud $m$. Requests in $U_{km}(t)$ can get serviced locally in a slot by micro-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

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

(5)

where $v_{km}(t)$ 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 $U(t)$. 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 $H_{km}(t) = 0$.

Micro-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 micro-clouds. Let $v_{km}(t)$ denote the number of requests from $U_{km}(t)$ that are transmitted to the back-end cloud in slot $t$ and let $v(t)$ denote the collection $\{v_{km}(t)\}$ for all $k,m$. Routing of $v_{km}(t)$ incurs a transmission cost of $c_{km}(t)$ where $c_{km}(t)$ is the unit back-end transmission cost that can depend on the application index as well as the micro-cloud index. Similar to request routing costs $e_{km}(t)$, $c_{km}(t)$ can also depend on other “uncontrollable” factors (such as backhaul traffic), but we only consider the average impact of these for simplicity. Since both the micro-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 $\mu_{max}$ such that $v_{km}(t) \leq v_{max}$ for all $t$. Further, $c_{km}(t) \leq c_{max}$ which models the baseline scenario where all requests are serviced only by the back-end cloud. Denote the set of all $v(t)$ that satisfy these constraints by $V$ and the sum total back-end transmission cost incurred in slot $t$ by $E(t)$, i.e., $E(t) = \sum_{km} v_{km}(t)c_{km}$. We assume that the back-end cloud has sufficient processing capacity such that it can service all requests in $v(t)$ 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 a micro-cloud.

Performance Objective: Given this model, our goal is to design a control algorithm for making request routing decisions at the users and micro-clouds as well as application reconfiguration decisions across the micro-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], which can then be relaxed and decoupled. Due to space limitation, we omit the details on MDP relaxation and decoupling in this paper, and refer readers to [9] for the details.

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 $U(t)$, new arrivals $a(t)$, user locations $l(t)$, and the last configuration $h(t−1)$. Then it makes a reconfiguration decision that transitions the configuration state to $h(t)$ and this determines the local service rates offered in slot $t$. Then the controller makes user to micro-cloud and micro-cloud to back-end cloud routing decisions. The queue backlogs $U(t + 1)$ at the start of the next slot evolve according to (5).

III. 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 $(l(t), a(t), h(t), U(t))$. 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 1) and are similar to the delay-aware queues used in [8].

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

(6)

where $\Xi \triangleq \mu_{km}(t) + v_{km}(t)$, and $0 \leq \sigma_{km} \leq v_{max}$ 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 1. Denote the collection $\{Z_{km}(t)\}$ by $Z(t)$ and the collection $\{\sigma_{km}\}$ by $\sigma$. 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 in [9], 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 $U(t)$ and $Z(t)$. In the following, we describe each of these components in detail. The performance guarantees provided by our algorithm are presented in Section IV.

A. User-to-Micro-Cloud Request Routing

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

$$\text{Minimize} \quad \sum_{km} \sum_n \left( U_{km}(t) + V_{km}(t) \right) r_{km}(t)$$

subject to $\{r_{km}(t)\} \in \mathcal{R}(l, a)$

(7)

where $\mathcal{R}(l, a)$ is defined by constraints (1), (2), $r_{km}(t) \in \mathbb{Z}_{\geq 0}$ for all $k,m$, and other location-based constraints as discussed.
in Sec. II-A. The resulting problem is an integer linear program (ILP) in the variables \( r_{km} \), for all \( k, m \). Further, the problem is separable across \( k \), i.e., it is optimal to solve \( K \) such problems separately, one per application \( k \). When \( R_{km} \geq \sum_{n=1}^{N} A_{kn} \) for all \( k, m \), the above optimization has a particular 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 \( r_{km} = A_{kn} \) for the particular micro-cloud \( m^* \) that user \( n \) can route to (given its current location \( L_n(t) \)) and that minimizes \( U_{km}(t) + V e_{km}(t) \). Set \( r_{km} = 0 \) for all \( m \neq m^* \). Note that \( c_{km} \) depends on the current user location \( (L_n(t)) \) as well as the indices of the application \( (k) \) and the micro-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, (7) can be mapped to variants of matching problems on bipartite graphs. For example, consider the case where \( A_{kn} = 1 \) for all \( k, n \), \( R_{km} = 1 \) for all \( k, m \), and \( N \leq M \). Then (7) becomes an instance of the minimum weight matching problem on a bipartite graph formed between the \( N \) users and \( M \) micro-clouds. This can be solved in polynomial time using well-known methods (such as in [10]). For more general cases, (7) becomes a generalized assignment problem that is NP-hard. However, efficient constant factor approximation algorithms are known for such problems [11].

As we show in Theorem 1, using any such approximation algorithm instead of the optimal solution to (7) 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 overall cost of the online algorithm is within the same approximation factor for any request that gets routed to \( U_{km} \).

Proof: We show that (9) holds with induction. First, (9) holds for \( t = 0 \) since all queues are initialized to 0. Now suppose \( U_{km}(t) \leq U_{km}^{\text{max}} \) for some \( t > 0 \). Then, we show that \( U_{km}(t + 1) \leq U_{km}^{\text{max}} \). We have two cases. First, suppose \( U_{km}(t) \leq V e_{km} \). Then, from queueing equation (5), it follows that the maximum value that \( U_{km}(t + 1) \) can have is \( U_{km}(t) + R_{km} \leq V e_{km} + R_{km} \). Next, suppose \( V e_{km} < U_{km}(t) \leq U_{km}^{\text{max}} \). Then, we have that \( U_{km}(t) + Z_{km}(t) > V e_{km} \) and the solution to (8) chooses \( v_{km}(t) = \min\{U_{km}(t), V e_{km}\} \). Since \( v_{km}^{\text{max}} \geq R_{km} \), from queueing equation (5) it follows that \( U_{km}(t + 1) \leq \max\{U_{km}(t), U_{km}^{\text{max}}\} \). The bound (10) follows similarly and its proof is omitted for brevity.

In Theorem 1, 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 \( U_{km} \).

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. 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 \( \sum_{t=f}^{\infty} \pi_{h} > 0 \). 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 [9]. 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 \( t_f \).

\[
\text{Minimize } \frac{1}{T_f} \sum_{\tau=0}^{T_f-1} \left( J\tau + VW(t_f + \tau) - \sum_{km} G_{km}(t_f, \tau) \right)
\]
subject to \( h(t_f + T_f - 1) = h_0 \).
where $G_{km}(t_f, \tau_f) = (U_{km}(t_f) + Z_{km}(t_f)) \mu_{km}(t_f + \tau)$ denotes the queue-length weighted service rate, $W(t_f + \tau)$ denotes the reconfiguration cost incurred in slot $(t_f + \tau)$, and $J = \sum_{k,m} J_{km}$ where $\sum_{k,m}$ is a constant defined as $J_{km} \triangleq 2(\mu_{km}^\text{max} + \sigma_{km}^2) + \sigma_{km}^2 + (R_{km})^2$. Note that the constraint $h(t_f + T_f - 1) = h_{t_f}$ 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 ($\sum_{k,m} G_{km}(t_f, \tau)$). Details on how to solve (11) is discussed in [9].

IV. PERFORMANCE ANALYSIS

We now analyze the performance of the online control algorithm presented in Section III. This is based on the technique of Lyapunov optimization over renewal periods [8], [12] 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 in [9] 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 $\phi_{km}^\text{stat}(\cdot)$, and $\phi_{km}^\text{stat}(\cdot)$ that are chosen to be equal to $\phi_{km}^\text{dec}(\cdot)$ and $\phi_{km}^\text{dec}(\cdot)$ respectively. If the resulting expected total service rate of any delay-aware queue $Z_{km}(t)$ is less than $\sigma_{km}$, then its back-end request routing is augmented by choosing additional $\nu_{km}(t)$ in an i.i.d. manner such that the expected total service rate becomes $\sigma_{km}$. It can be shown that the resulting time-average back-end routing cost under this algorithm is at most $e^* + \phi(\sigma)$ where $\phi(\sigma) = \sum_{k,m} \max \{\sigma_{km} - \mu_{km}^\text{dec} - \nu_{km}^\text{dec}, 0\} \epsilon_{km}$. By comparing the Lyapunov drift plus cost of the online control algorithm over renewal frames with this stationary algorithm, we have the following.

**Theorem 1:** Suppose the online control algorithm defined by (7), (8), and (11) is implemented with a renewal state $h_0 \in \mathcal{H}$ using control parameters $V > 0$ and $0 \leq \sigma_{km} \leq \nu_{km}^\text{max}$ 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

$$
\lim_{F \to \infty} \sum_{f=1}^{F-1} \frac{C(\tau_f) + W(\tau_f) + E(\tau_f)}{\sum_{f=1}^{F-1} E(T_f)} \leq e^* + w^* + e^* + \phi(\sigma) + 1 + \sum_{k,m} B_{km} \frac{\nu_{km}^\text{dec}}{V} \leq e^* + w^* + e^* + \phi(\sigma) + 1 + \sum_{k,m} B_{km} \frac{\nu_{km}^\text{dec}}{V} \leq e^* + w^* + e^* + \phi(\sigma) + 1 + \sum_{k,m} B_{km} \frac{\nu_{km}^\text{dec}}{V} \leq e^* + w^* + e^* + \phi(\sigma) + 1 + \sum_{k,m} B_{km} \frac{\nu_{km}^\text{dec}}{V}
$$

where $B_{km} = (1 + T_{h_{km}}(\tau_{h_{km}}) / 2 + \delta (R_{km})^2, \ Y_{h_{km}} = \frac{\E[T_{h_{km}}(\tau_{h_{km}}) - 1]}{\E[T_{h_{km}}]}, \ \phi(\sigma) = \sum_{k,m} \max \{\sigma_{km} - \mu_{km}^\text{dec} - \nu_{km}^\text{dec}, \nu_{km}^\text{dec}, 0\} \epsilon_{km}$ and $\delta$ 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 $d_{km}^\text{max}$ for any request routed to queue $U_{km}(t)$ is upper bounded by

$$
d_{km}^\text{max} \leq \frac{U_{km}^\text{max} + Z_{km}^\text{max}}{\sigma_{km}} = 2V_{km} + R_{km}^\text{max} + \sigma_{km} \leq \frac{U_{km}^\text{max} + Z_{km}^\text{max}}{\sigma_{km}}
$$

3) Suppose we implement an algorithm that approximately solves (7) and (11) resulting in the following bound for all slots for some $\rho \geq 1$

$$
\sum_{k,m} \sum_{n} \left( U_{km}(t) + V_{km}(t) \right) r_{kmn}(t) \leq \rho \sum_{k,m} \sum_{n} \left( U_{km}(t) + V_{km}(t) \right) r_{kmn}(t)
$$

and the following bound every renewal frame

$$
\frac{1}{T_f} \sum_{\tau=0}^{T_f-1} \left( \sum_{k,m} \rho J_{km} \tau - (U_{km}(t_f) + Z_{km}(t_f)) + \nu_{km}^\text{dec} \right) + W_{km}(t_f + \tau) \leq B' + \frac{\rho}{T_f} \sum_{\tau=0}^{T_f-1} \left( \sum_{k,m} J_{km} \tau - (U_{km}(t_f) + Z_{km}(t_f)) + \nu_{km}^\text{dec} \right) + W_{km}(t_f + \tau)
$$

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 (7) and (11) respectively. Then the time-average expected transmission plus reconfiguration costs under this approximation algorithm is at most

$$
\rho \left( e^* + w^* + e^* + \phi(\sigma) + \frac{1 + \sum_{k,m} B_{km} + (R_{km}^\text{max} + \sigma_{km}) \nu_{km}^\text{max}}{V} \right) + \frac{B' - \sum_{k,m} (R_{km}^\text{max} + \sigma_{km}) \nu_{km}^\text{max}}{V}
$$

while the delay bounds remain the same as (13).

**Proof:** See Appendix C in [9].

**Discusison on Performance Tradeoffs:** Our control algorithm offers tradeoffs between cost and delay performance guarantees through the control parameters $V$ and $\sigma$. For a given $V$ and $\sigma$, the bound in (12) implies that the time-average expected transmission plus reconfiguration costs are within an additive term $\phi(\sigma) + O(\log V/V)$ term of the optimal cost while (13) bounds the worst case delay by $O(V/\sigma_{km})$. This shows that by increasing $V$ and decreasing $\sigma$, 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 systems. Thus, it is noteworthy that we can achieve similar tradeoff in an MDP setting. Note that if there exist $\sigma_{km} \geq 0 \forall k,m$ such that $\sigma_{km} \leq \mu_{km}^\text{dec} + \nu_{km}^\text{dec}$, then $\phi(\sigma) = 0$ 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 $e^* + w^* + e^*$ and a multiplicative approximation with respect to the average delay $d_{\text{avg}}$ of the optimal solution to the original MDP defined in [9].

In practice, setting $\sigma_{km} = 0 \forall k, m$ should yield good delay performance even though (13) 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.

V. 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 [13]. We assume that the micro-clouds are co-located with a subset of the basestations of a wireless network. A hexagonal symmetric cellular structure is assumed with 91 cells in total as shown in Fig. 2. Out of the 91 basestations, 10 host micro-clouds and there are 5 applications in total. For simplicity, each micro-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 micro-cloud in cell $l'$, we define its transmission cost as

$$\text{trans}_n(l, l') = \begin{cases} 1 + 0.1 \cdot \text{dist}(l, l'), & \text{if } l \neq l' \\ 0, & \text{if } l = l' \end{cases} \quad (17)$$

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 micro-cloud in cell $l$ to the micro-cloud in cell $l'$, the reconfiguration cost for this specific application is defined as

$$\text{recon}_k(l, l') = \begin{cases} \kappa(1 + 0.1 \cdot \text{dist}(l, l')), & \text{if } l \neq l' \\ 0, & \text{if } l = l' \end{cases} \quad (18)$$

where $\kappa$ 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 $\kappa$ 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 $\lambda$ 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 active

users. In both cases, $\lambda$ is the time-average (over the simulation duration) aggregate arrival rate per application per slot, while the micro-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 micro-cloud and reconfiguration never happens. User requests are always routed to the micro-cloud that hosts the corresponding application. In the always migrate policy, user requests are always routed to the micro-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 micro-cloud can only host one application). We also assume that the request arrival rate $\lambda$ is known in the never and always migrate policies. If $\lambda > 1$, the arrival rate exceeds the micro-cloud capacity, and the requests that are queued in micro-clouds are probabilistically routed to the back-end cloud, where the probability is chosen such that the average arrival rate to micro-clouds does not exceed the service rate at micro-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 micro-cloud that hosts the application after possible reconfiguration.

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 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 $\sigma_{km}$ to the same value $\sigma$ 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. 3 for both values of $\kappa$, where we set $\lambda = 0.95$.

We can see from the results that, for each fixed $\sigma$, the queue lengths and cost values under the Lyapunov algorithm follow the $O(V, \log V/V)$ trend as suggested by the bounds (12) and (13). The impact of the value of $\sigma$ is also as predicted by these bounds. Namely, a smaller value of $\sigma$ yields larger queue lengths and lower costs, while a larger value of $\sigma$ yields smaller queue lengths and higher costs. When comparing all four algorithms in Fig. 3(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. 3(c), (d).

We next study the queue lengths and costs under different values of the arrival rate $\lambda$, where we fix $V = 100$ and $\sigma = 0$. Results are shown in Fig. 4. 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 micro-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 micro-cloud is usually lower than routing to the back-end cloud, an excessive amount of requests are routed to micro-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 micro-clouds, as explained earlier. However, they incur higher costs as shown in Fig. 4(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 [13] 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 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 micro-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. 5 and 6 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. 5 and 6, 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.

### VI. 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 micro-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.

### ACKNOWLEDGMENT

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


### TABLE I

<table>
<thead>
<tr>
<th>Policy</th>
<th>Queue lengths ($\kappa = 0.5$)</th>
<th>Costs ($\kappa = 0.5$)</th>
<th>Queue lengths ($\kappa = 1.5$)</th>
<th>Costs ($\kappa = 1.5$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lyapunov</td>
<td>106.5</td>
<td>5.954</td>
<td>160.2</td>
<td>6.446</td>
</tr>
<tr>
<td>Never migrate</td>
<td>268.7</td>
<td>6.414</td>
<td>268.7</td>
<td>6.414</td>
</tr>
<tr>
<td>Always migrate</td>
<td>3117</td>
<td>7.983</td>
<td>3117</td>
<td>14.15</td>
</tr>
<tr>
<td>Myopic</td>
<td>437.4</td>
<td>6.228</td>
<td>851</td>
<td>6.268</td>
</tr>
</tbody>
</table>

**Average Values for Trace-Driven Simulation**