Dynamic Service Placement for Mobile Micro-Clouds with Predicted Future Costs

Shiqiang Wang*, Rahul Urgaonkar†, Kevin Chan‡, Ting He†, Murtaza Zafer§¶, and Kin K. Leung*
*Department of Electrical and Electronic Engineering, Imperial College London, United Kingdom
†IBM T. J. Watson Research Center, Yorktown Heights, NY, United States
‡Army Research Laboratory, Adelphi, MD, United States
§Nyansa Inc., Palo Alto, CA, United States
Email: *{shiqiang.wang11, kin.leung}@imperial.ac.uk, †{rurgaon, the}@us.ibm.com,
‡kevin.s.chan.civ@mail.mil, §murtaza.zafer.us@ieee.org

***Abstract*—Seamless computing and data access is enabled by the emerging technology of mobile micro-clouds (MMCs). Different from traditional centralized clouds, an MMC is typically connected directly to a wireless base-station and provides services to a small group of users, which allows users to have instantaneous access to cloud services. Due to the limited coverage area of base-stations and the dynamic nature of mobile users, network background traffic, etc., the question of where to place the services to cope with these dynamics arises. In this paper, we focus on dynamic service placement for MMCs. We consider the case where there is an underlying mechanism to predict the future costs of service hosting and migration, and the prediction error is assumed to be bounded. Our goal is to find the optimal service placement sequence which minimizes the average cost over a given time. To solve this problem, we first propose a method which solves for the optimal placement sequence for a specific look-ahead time-window, based on the predicted costs in this time-window. We show that this problem is equivalent to a shortest-path problem and propose an algorithm with polynomial time-complexity to find its solution. Then, we propose a method to find the optimal look-ahead window size, which minimizes an upper bound of the average cost. Finally, we evaluate the effectiveness of the proposed approach by simulations with real-world user-mobility traces.**

***Index Terms*—Cloud computing, cost prediction, dynamic scheduling, mobile micro-cloud (MMC), mobility, wireless networks**

I. Introduction

Many emerging applications, such as video streaming, real-time face/object recognition, etc., require high data processing capability. However, portable devices (e.g. smartphones) are generally limited by their size and battery life, which makes them incapable to perform complicated computational tasks. A remedy for this is to utilize cloud computing techniques, where the cloud performs the computation for its users. In the traditional setting, cloud services are provided by centralized data-centers that may locate *far away* from end-users, which can be inefficient because the user may suffer from long latency and poor connectivity due to long-distance communication [1]. The idea of mobile micro-clouds (MMCs) is to place the cloud *closer* to end-users, so that users can have fast and reliable access to the service. In an MMC, a server or a small data-center is *directly connected to the wireless base-station*, providing cloud services to users that are either connected to the base-station or are within a reasonable distance from it. Fig. 1 shows an application scenario where MMCs coexist with the centralized cloud. MMCs can be used for many applications that require high reliability or high data processing capability [1]–[3]. This concept has also been termed as Cloudlet [1], Follow Me Cloud [4], edge computing [5], small cell cloud [6], etc. We use the term MMC in this paper.

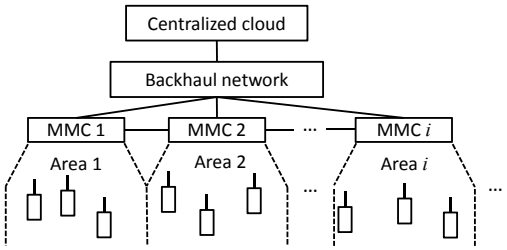


Figure 1. Application scenario with mobile micro-clouds (MMCs).

One important issue in MMCs is to decide which MMC should perform the computation for a particular user, with the presence of user mobility and other dynamic changes in the network. When a user wants to run a service provided by the cloud, it can run it either in the centralized cloud or in one of the MMCs, and the question is how to choose the optimal location to run the service. Besides, the user may frequently switch among different base-stations due to mobility, thus another question is whether we should migrate the service from one cloud (which can be either an MMC or the centralized cloud) to another cloud when the user location or network condition changes.

The abovementioned problems are related to application/workload placement problems in cloud environments. Although existing work has studied such problems under complicated network topologies [7], [8], they mainly focused on static network conditions and fixed resource demands. The presence of dynamically changing resource availability that is related to user mobility has not been sufficiently considered. When user mobility exists, it is necessary to consider real-time service migration. For example, it can be beneficial to migrate

¶ Contributions of the author to this work are not related to his current employment at Nyansa Inc.

the service to a location closer to the user. Only a few existing papers in the literature have studied this problem [9], [10]. The main approach in [9] and [10] is to formulate the mobility-driven service migration problem as a Markov decision process (MDP). Such a formulation is suitable where users follow a mobility model that can be described by a Markov chain. However, there are cases where the Markovian assumption is not valid [11]. Besides, [9] and [10] assume specific structures of the cost function that are related to the service and user locations. Such cost structures can inapplicable when the load on different MMCs are imbalanced or when we consider the centralized cloud as a service placement option.

In this paper, we consider a more general setting which allows heterogeneity in cost values, network structure, and mobility models. We focus on the case where there is an underlying mechanism to predict the future costs of running the service in each MMC (or the centralized cloud) as well as the costs of service migration. The cost may be related to the service/user location, network condition, and other factors such as user preference. Specific methods for predicting future costs are beyond the scope of this paper, but we anticipate that existing approaches such as [12], [13] and [14] can be applied. We assume that the prediction mechanism provides the most likely future cost sequence and an upper bound on possible deviation of the actual cost from the predicted cost value. Such an assumption is valid for many prediction methods that provide guarantees on the prediction accuracy. Unlike the MDP-based approaches in [9] and [10], we do not require knowledge on the probability distribution of the costs.

Based on the above model, we formulate a problem with the goal of finding the optimal service placement sequence that minimizes the average cost over a given time. We define a look-ahead window to represent the amount of time that we look (predict) into the future. We first propose an algorithm to find the optimal placement sequence for a specific look-ahead window size. Then, we propose a method to find the optimal window size, which considers the existence of prediction errors and minimizes an upper bound of the average cost. The effectiveness of the proposed approach is evaluated by simulations with real-world user-mobility traces.

The remainder of this paper is organized as follows. The problem formulation is described in Section II. Section III proposes an algorithm to find the optimal service placement sequence with given look-ahead window size. Section IV proposes a method to find the optimal look-ahead window size. Section V discusses the simulation results and Section VI draws conclusions.

II. Problem Formulation

We consider a cloud computing system as shown in Fig. 1, where the clouds are indexed by $m \in \{1, 2, ..., M\}$. The MMCs have indexes $m \in \{1, 2, ..., M-1\}$, and the centralized cloud has index $m = M$. All MMCs together with the centralized cloud can host services. A time-slotted system as shown in Fig. 2 is considered, where the slots $t \in \{1, 2, ..., T_{\max}\}$ can be either evenly or unevenly spaced. The choice of the length

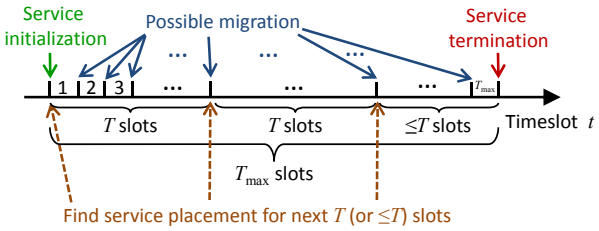


Figure 2. Timing of the proposed approach.

of each timeslot is related to how fast the costs change and the complexity of finding the optimal service placement sequence. The total number of slots $T_{\max}$ matches with the time duration that the service runs.

For simplicity, we consider a single user running one non-decomposable service in our formulation. This can be *easily extended to multiple users* running the same service, by defining the cost to be related to multiple users. The case with multiple users running independent services can also be captured by appropriately defining the cost functions to take into account the load from other users, as we do in the simulations in Section V. The service is initialized at the beginning of the first timeslot $t = 1$, and it can be migrated from one cloud to another cloud at the beginning of all subsequent timeslots $t \in \{2, 3, ..., T_{\max}\}$.

A. Cost Definition

We consider two types of costs. The *local cost* $u(t, m)$ specifies the cost of data transmission and processing in timeslot t when the service is placed at cloud m. Its value can depend on many factors, such as the user location, network condition, and load of the cloud in slot t. The local cost in slot $t = 1$ also includes the cost of initial placement. We then define the *migration cost* $w(t, n, m)$, which specifies the cost of service migration from cloud n to cloud m *at the beginning of slot* t. We define $w(\cdot, m, m) = 0$, because there is no migration cost if we do not migrate. There is also no migration cost in the first timeslot, thus we have $w(1, \cdot, \cdot) = 0$.

Let vector $\boldsymbol{\pi}$ denote the sequence of service placements for all timeslots $t \in \{1, 2, ..., T_{\max}\}$, where the tth element of $\boldsymbol{\pi}$ (denoted by $\pi(t)$) specifies the index of the cloud that the service is placed on in slot t. The sum of local and migration costs in slot t when following placement sequence $\boldsymbol{\pi}$ is

$$C_{\boldsymbol{\pi}}(t) \triangleq u(t, \pi(t)) + w(t, \pi(t-1), \pi(t)) \qquad (1)$$

Note that $C_{\boldsymbol{\pi}}(t)$ is only dependent on $\pi(t-1)$ and $\pi(t)$, but we use vector $\boldsymbol{\pi}$ as subscript for simplicity.

B. Actual and Predicted Costs

To distinguish between the actual and predicted cost values, we let $A_{\boldsymbol{\pi}}(t)$ denote the *Actual* value of $C_{\boldsymbol{\pi}}(t)$, and let $D^{t_0}_{\boldsymbol{\pi}}(t)$ denote the *preDicted* most likely value of $C_{\boldsymbol{\pi}}(t)$, when prediction is performed at the beginning of slot t_0. For completeness of notations, we define $D^{t_0}_{\boldsymbol{\pi}}(t) = A_{\boldsymbol{\pi}}(t)$ for $t < t_0$, because at the beginning of t_0, the costs of all past timeslots are known. For $t \geq t_0$, we assume that the absolute difference between $A_{\boldsymbol{\pi}}(t)$ and $D^{t_0}_{\boldsymbol{\pi}}(t)$ is at most

$\epsilon(\tau) \triangleq \max_{\boldsymbol{\pi},t_0} |A_{\boldsymbol{\pi}}(t_0+\tau) - D^{t_0}_{\boldsymbol{\pi}}(t_0+\tau)|$, which represents the maximum error when looking ahead for τ slots, among all possible placement sequences $\boldsymbol{\pi}$ and all possible prediction instances t_0. The function $\epsilon(\tau)$ is assumed to be non-decreasing with τ, because we generally cannot have lower error when we look farther ahead into the future. The specific value of $\epsilon(\tau)$ is provided by the cost prediction module. We also define a constant $\delta \triangleq \max_{t,n,m} w_a(t,n,m)$ to represent the maximum value of the actual migration cost in any slot, where $w_a(t,n,m)$ denotes the actual cost of migrating from n to m at the beginning of slot t. The value of δ is system-specific and is related to the cost definition.

C. Our Goal

Our ultimate goal is to find the optimal placement sequence $\boldsymbol{\pi}^*$ that minimizes the *actual* time average cost over the entire duration that the service runs, i.e.

$$\boldsymbol{\pi}^* = \arg\min_{\boldsymbol{\pi}} \frac{\sum_{t=1}^{T_{\max}} A_{\boldsymbol{\pi}}(t)}{T_{\max}} \tag{2}$$

However, it is impractical to find the optimal solution to (2), because we cannot precisely predict the future costs. Therefore, we focus on obtaining an approximate solution to (2) by utilizing *predicted* cost values.

D. Approximate Solution from Predicted Costs

In the approximate solution, as shown in Fig. 2, we predict the costs for the next T timeslots, and we define these T slots as a *look-ahead window*. The service placement sequence is found at the beginning of each look-ahead window, such that the average *predicted* cost within the window is minimized. We denote the placement sequence found from the predicted costs (for all slots $t \in \{1,2,...,T_{\max}\}$) by $\boldsymbol{\pi}_T$. The window size T is chosen so that the average *actual* cost when placing according to $\boldsymbol{\pi}_T$ is closest (in an upper-bound sense) to the average *actual* cost when placing according to $\boldsymbol{\pi}^*$. Details will be given in Sections III and IV.

III. Service Placement Based on Predicted Costs with Given Look-Ahead Window Size

We start with illustrating the high-level procedure of finding $\boldsymbol{\pi}_T$. When the look-ahead window size T is given, the placement sequence $\boldsymbol{\pi}_T$ is found according to the following steps (see also Fig. 2):

1) Initialize $t_0 = 1$.
2) Let $t_e = \min\{t_0 + T - 1; T_{\max}\}$. At the beginning of slot t_0, find

$$\boldsymbol{\pi}_T(t_0,...,t_e) = \arg\min_{\boldsymbol{\pi}(t_0,...,t_e)} \sum_{t=t_0}^{t_e} D^{t_0}_{\boldsymbol{\pi}(t_0,...,t_e)}(t) \tag{3}$$

   where $\boldsymbol{\pi}(t_0,...,t_e)$ denotes the placement sequence for timeslots $t_0,...,t_e$, and $D^{t_0}_{\boldsymbol{\pi}}(t)$ can be found from the cost prediction module.
3) Apply placements $\boldsymbol{\pi}_T(t_0,...,t_e)$ in timeslots $t_0,...,t_e$.
4) If $t_e < T_{\max}$, set $t_0 = t_e + 1$ and go to step 2. Else, stop.

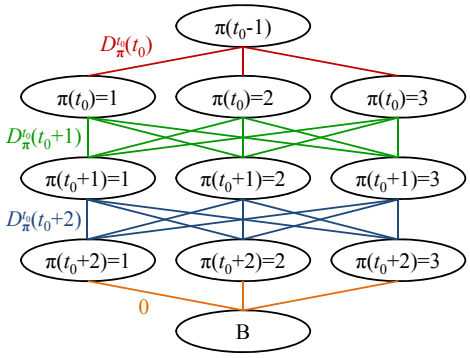


Figure 3. Shortest-path formulation with $M = 3$ and $T = 3$.

Note that in the above procedure, the vector $\boldsymbol{\pi}_T$ is found in real-time. We can get the full vector of $\boldsymbol{\pi}_T$ only after (3) has been solved for timeslots in the last window. Such a solution is sufficient in practice because we only need to know the placement in the current timeslot to make the placement decision. The value of $D^{t_0}_{\boldsymbol{\pi}(t_0,...,t_e)}(t)$ in (3) also depends on the placement in timeslot $t_0 - 1$, i.e. $\pi(t_0 - 1)$, according to (1). When $t_0 = 1$, $\pi(t_0 - 1)$ can be regarded as any dummy variable, because the migration cost $w(1,\cdot,\cdot) = 0$.

The intuitive explanation of (3) is that, at the beginning of slot t_0, it finds the optimal placement sequence that minimizes the predicted cost over the next slots (including slot t_0) up to t_e, given the location of the service in the previous slot $t_0 - 1$. The remaining part of this section focuses on solving (3), and we write $\boldsymbol{\pi}(t_0,...,t_e)$ as $\boldsymbol{\pi}$ for short.

A. Equivalence to Shortest-Path Problem

The problem (3) is equivalent to a shortest-path problem with $D^{t_0}_{\boldsymbol{\pi}}(t)$ as weights, as shown in Fig. 3. Each edge represents one possible combination of service placements in adjacent timeslots, and the weight on each edge is the predicted cost for such placement. The placement in timeslot $t_0 - 1$ is always given, and the number of possible service locations in subsequent timeslots is equal to M. Node B is a dummy node to ensure that we find a single shortest path, and the edges connecting node B have zero weights. It is obvious that the optimal solution to (3) can be found by taking the shortest (minimum-weighted) path from node $\boldsymbol{\pi}(t_0 - 1)$ to node B in Fig. 3, and the nodes that the shortest path traverses correspond to the optimal solution $\boldsymbol{\pi}_T(t_0,...,t_e)$ for (3).

B. Algorithm

We can solve the abovementioned shortest-path problem by means of dynamic programming [15]. The algorithm is shown in Algorithm 1, where we use $u_p(t,m)$ and $w_p(t,n,m)$ to respectively denote the predicted local and migration costs, when the service is placed at cloud m in the current timeslot and at cloud n in the previous timeslot. In the algorithm, Lines 5–16 iteratively finds the shortest path (minimum objective function) for each timeslot. In each iteration, the optimal solution for placing the service on each cloud m is found by solving the Bellman's equation of the problem (Line 11). Then, the final optimal placement is found in Lines 17 and

Algorithm 1 Algorithm for solving (3)

1: Define variables m and n with $m, n \in \{1, 2, ..., M\}$ to represent cloud indexes respectively in the current and previous iteration
2: Define vectors $\boldsymbol{\pi}_m$ and $\boldsymbol{\xi}_m$ for all $m \in \{1, 2, ..., M\}$, where $\boldsymbol{\pi}_m$ (correspondingly, $\boldsymbol{\xi}_m$) records the optimal service placement sequence given that the placement at the current (correspondingly, previous) timeslot of iteration is at m
3: Define variables μ_m and ν_m for all $m \in \{1, 2, ..., M\}$ to record the sum cost values from slot t_0 respectively to the current and previous slot of iteration, given that the service is placed at m in the current or previous slot
4: Initialize $\mu_m \leftarrow 0$ and $\boldsymbol{\pi}_m \leftarrow \emptyset$ for all $m \in \{1, 2, ..., M\}$
5: **for** $t = t_0, ..., t_e$ **do**
6: **for** $m = 1, ..., M$ **do**
7: $\nu_m \leftarrow \mu_m$
8: $\boldsymbol{\xi}_m \leftarrow \boldsymbol{\pi}_m$
9: **end for**
10: **for** $m = 1, ..., M$ **do**
11: $n^* \leftarrow \arg\min_n \{\nu_n + u_p(t, m) + w_p(t, n, m)\}$
12: $\boldsymbol{\pi}_m(t_0, ..., t-1) \leftarrow \boldsymbol{\xi}_{n^*}(t_0, ..., t-1)$
13: $\pi_m(t) \leftarrow m$
14: $\mu_m \leftarrow \nu_{n^*} + u_p(t, m) + w_p(t, n^*, m)$
15: **end for**
16: **end for**
17: $m^* \leftarrow \arg\min_m \mu_m$
18: $\boldsymbol{\pi}_T(t_0, ..., t_e) \leftarrow \boldsymbol{\pi}_{m^*}(t_0, ..., t_e)$
19: **return** $\boldsymbol{\pi}_T(t_0, ..., t_e)$

It is obvious that output of this algorithm satisfies the Bellman's principle of optimality, so the result is the shortest path and hence the optimal solution to (3).

When the vectors $\boldsymbol{\pi}_m$ and $\boldsymbol{\xi}_m$ are stored in linked-lists, Algorithm 1 has time-complexity $O(M^2T)$; and when $\boldsymbol{\pi}_m$ and $\boldsymbol{\xi}_m$ are stored in arrays, the time-complexity is $O((M + T)MT)$. This is because the minimization in Line 11 requires enumerating through M clouds, and if $\boldsymbol{\pi}_m$ and $\boldsymbol{\xi}_m$ are stored in arrays, then Line 12 copies at most T data elements.

IV. Optimal Look-Ahead Window Size

In this section, we study how to find the optimal window size T to look-ahead. We note that T is upper bounded by the service duration $T_{\max}$. When there are no errors in the cost prediction, setting $T = T_{\max}$ makes (3) equivalent to (2), which brings the best long-term performance. However, the problem becomes more complicated when we consider the prediction error, because the farther ahead we look into the future, the more uncertain we are about the costs. When T is large, the predicted cost value may be far away from the actual cost, which can cause the solution from the predicted cost ($\boldsymbol{\pi}_T$) deviate significantly from the true optimal solution $\boldsymbol{\pi}^*$. Conversely, when T is small, the solution may not perform well in the long-term, because the long-term effect of service placement is not considered. Therefore, we have to find the optimal value of T which minimizes both the impact of prediction error and the impact of truncating the look-ahead time-span.

Recall that in Section II, we defined the maximum difference between the predicted and actual costs when looking τ slots ahead as $\epsilon(\tau)$, we also defined the maximum migration cost as δ. Both $\epsilon(\tau)$ and δ are regarded as inputs to our problem. To help with our analysis below, we define the sum-error starting from slot t_0 up to slot $t_0 + T - 1$ as

$$F(T) = \sum_{t=t_0}^{t_0+T-1} \epsilon(t - t_0) \tag{4}$$

Because $\epsilon(t - t_0) \geq 0$ and $\epsilon(t - t_0)$ is non-decreasing with t, it is obvious that $F(T + 2) - F(T + 1) \geq F(T + 1) - F(T)$. Hence, $F(T)$ is a convex non-decreasing function for $T \geq 0$, where we define $F(0) = 0$.

A. Upper Bound of Cost Difference

In the following, we focus on the objective function given in (2), and study how worse the placement sequence $\boldsymbol{\pi}_T$ can perform, compared with the true optimal sequence $\boldsymbol{\pi}^*$.

Proposition 1. *For look-ahead window size T, the upper bound on the cost difference from placement sequences $\boldsymbol{\pi}_T$ and $\boldsymbol{\pi}^*$ is given by*

$$\frac{\sum_{t=1}^{T_{max}} A_{\boldsymbol{\pi}_T}(t)}{T_{max}} - \frac{\sum_{t=1}^{T_{max}} A_{\boldsymbol{\pi}^*}(t)}{T_{max}} \leq \frac{2F(T) + \delta}{T} \tag{5}$$

Proof: See Appendix. ∎

We define the optimal look-ahead window size as the solution to the following optimization problem:

$$\begin{aligned} \min_T \quad & \frac{2F(T) + \delta}{T} \\ \text{s.t.} \quad & 1 \leq T \leq T_{\max} \end{aligned} \tag{6}$$

Considering the original objective in (2), the problem (6) can be regarded as finding the optimal look-ahead window size such that an upper bound of the objective function in (2) is minimized. It is impractical to solve (2) directly because we cannot precisely predict the future costs. However, as discussed above, we can approximate (2) by utilizing predicted costs, and the solution to (6) is the optimal window size to look-ahead so that (in the worst case) the cost is closest to the cost from the true optimal placement sequence $\boldsymbol{\pi}^*$.

B. Characteristics of the Optimization Problem in (6)

In the following, we study the characteristics of (6). To help with the analysis, we interchangeably use variable T to represent either a discrete or a continuous variable. We define a continuous convex function $G(T)$, where T is a continuous variable within the interval of $[1, T_{\max}]$. The function $G(T)$ is defined in such a way that $G(T) = F(T)$ for all the discrete values $T \in \{1, 2, ..., T_{\max}\}$, i.e. $G(T)$ is a *continuous time extension* of $F(T)$. Such a definition is always possible by connecting the discrete points in $F(T)$. We will work with continuous values of T in some parts of our analysis below, and will discretize it when appropriate.

We define a function $\theta(T) \triangleq \frac{2G(T)+\delta}{T}$ to represent the upper bound in (5) after replacing $F(T)$ with $G(T)$, where T is regarded as a continuous variable. We take the logarithm of $\theta(T)$, yielding

$$\ln\theta = \ln\left(2G(T)+\delta\right) - \ln T \tag{7}$$

Taking the derivative of $\ln\theta$, we have

$$\frac{d\ln\theta}{dT} = \frac{2\frac{dG(T)}{dT}}{2G(T)+\delta} - \frac{1}{T} \tag{8}$$

We set (8) equal to zero, and rearrange the equation, yielding

$$\Phi(T) \triangleq 2T\frac{dG(T)}{dT} - 2G(T) - \delta = 0 \tag{9}$$

where we define $\Phi(T)$ to represent the left hand-side of (9).

Proposition 2. *Let T_0 denote a solution to (9), if the solution exists, then the optimal look-ahead window size T^* for problem (6) is either $\lfloor T_0 \rfloor$ or $\lceil T_0 \rceil$, where $\lfloor x \rfloor$ and $\lceil x \rceil$ respectively denote the floor (rounding down to integer) and ceiling (rounding up to integer) of x .*

Proof: Taking the derivative of $\Phi(T)$, we get

$$\frac{d\Phi}{dT} = 2T\frac{d^2G(T)}{dT^2} \geq 0 \tag{10}$$

where the last inequality is because $G(T)$ is convex. This implies that $\Phi(T)$ is non-decreasing with T. Hence, there is at most one consecutive interval of T (the interval may only contain one value) such that (9) is satisfied. We denote this interval by $[T_-, T_+]$, and a specific solution to (9) is $T_0 \in [T_-, T_+]$.

We note that $\frac{d\ln\theta}{dT}$ and $\Phi(T)$ have the same sign, because $\frac{d\ln\theta}{dT} \lessgtr 0$ yields $\Phi(T) \lessgtr 0$ and vice versa, which can be seen from (8) and (9). When $T < T_-$, we have $\Phi(T) < 0$ and hence $\frac{d\ln\theta}{dT} < 0$; when $T > T_+$, we have $\Phi(T) > 0$ and hence $\frac{d\ln\theta}{dT} > 0$. This implies that $\ln\theta$, thus $\theta(T)$, keeps decreasing with T until the optimal solution is reached, and afterwards it keeps increasing with T. It follows that the minimum value of $\theta(T)$ is attained at $T \in [T_-, T_+]$. Because $T_0 \in [T_-, T_+]$ and T^* is a discrete variable, we complete the proof. ∎

Note that we do not assume the continuity of the derivatives of $G(T)$ in the above analysis, which means that $\frac{dG(T)}{dT}$ may be non-continuous and $\frac{d^2G(T)}{dT^2}$ may have $+\infty$ values. However, these do not affect our analysis above. Also note that we do not consider the convexity of $\theta(T)$. From the proof of Proposition 2, we can also conclude the following corollary.

Corollary 1. *For window sizes T and $T+1$, if $\theta(T) < \theta(T+1)$, then the optimal size $T^* \leq T$; if $\theta(T) > \theta(T+1)$, then $T^* \geq T+1$; if $\theta(T) = \theta(T+1)$, then $T^* = T$.*

C. Finding the Optimal Solution

According to Proposition 2, we can solve (9) to find the optimal look-ahead window size. When $G(T)$ (and $F(T)$) can be expressed in some specific analytical forms, the solution to (9) can be found analytically. For example, consider $G(T) = F(T) = \beta T^\alpha$, where $\beta > 0$ and $\alpha > 1$. In this case, $T_0 = \left(\frac{\delta}{2\beta(\alpha-1)}\right)^{\frac{1}{\alpha}}$, and $T^* = \arg\min_{T\in\{\lfloor T_0\rfloor, \lceil T_0\rceil\}} \theta(T)$. One can also use such specific forms as an upper bound for a general function.

When $G(T)$ (and $F(T)$) have more general forms, we can perform a search on the optimal window size according to the properties discussed in Section IV-B. Because we do not know the convexity of $\theta(T)$ or $\Phi(T)$, standard numerical methods for solving (6) or (9) may not be efficient. However, from Corollary 1, we know that the local minimum point of $\theta(T)$ is the global minimum point, so we can develop algorithms that use this property.

Because the optimal window size T^* takes discrete values, we can perform a discrete search on $T \in \{1, 2, ..., T_{\max}\}$, and compare $\theta(T)$ with $\theta(T+1)$ and determine the optimal solution according to Corollary 1. One possible approach is to use binary search, as shown in Algorithm 2, which has time-complexity of $O(\log T_{\max})$.

Algorithm 2 Binary search for finding optimal window size

```
Initialize variables T_- ← 1 and T_+ ← T_max
repeat
    T ← ⌊(T_- + T_+)/2⌋
    if θ(T) < θ(T+1) then
        T_+ ← T
    else if θ(T) > θ(T+1) then
        T_- ← T+1
    else if θ(T) = θ(T+1) then
        return T //Optimum found
    end if
until T_- = T_+
return T_-
```

V. Simulation Results

To evaluate the performance of the proposed approach, we perform simulations using real-world San Francisco taxi traces on May 31, 2008, obtained from [16], [17]. We consider the existence of a cellular system that is deployed in the area, where a hexagon cellular structure is assumed and the distance between adjacent base-stations is 1000 m. There are 91 base-stations in total, and each user connects to its closest base-station for network access. We assume that there exist a centralized cloud and multiple MMCs, each base-station is connected to an MMC. The service can be placed on either one of the MMCs or on the centralized cloud. There are 536 users (taxis) in total and not all the users are active at a given time. We assume that each active user generates some load to the network. Among the users, 50 of them require service from the cloud when they are active. We consider two sets of simulations where the time-lengths of each timeslot are respectively set to 60 s and 300 s.

The local cost of running the service at MMC m in timeslot t is defined as

$$u(t,m) = \frac{1}{1-\frac{s(t,m)}{s_{\max}+1}} + kr_1(t,m) \tag{11}$$

where $s(t,m)$ denotes the number of users that are associated with the base-station connected to MMC m in slot t, the constant $s_{\max}$ is the maximum number of users at any base-station in any timeslot, $r_1(t,m)$ is the distance (expressed as the number of hops) between the base-station that the user is currently connected to and the MMC location running its service. The first term in (11) captures the load at MMCs. It can be explained as related to the queuing delay of processing/transmission requests according to queuing theory, and it is a widely used objective (such as in [8]) which pushes the system towards a load-balanced state. The second term in (11) captures the cost increase with the user-service distance. The constant k is a trade-off factor, which equals to 0.2 in the simulations. Similarly, we define the migration cost from MMC n to MMC m as

$$w(t,n,m) = \frac{1}{1-\frac{s(t,n)+s(t,m)}{2(s_{\max}+1)}} + kr_2(t,n,m) \qquad (12)$$

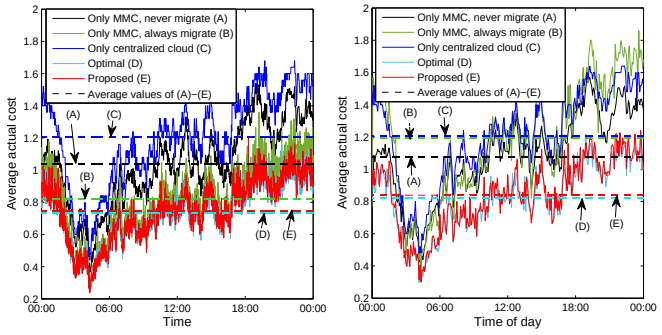
where $r_2(t,n,m)$ is the distance between MMCs n and m. The local and migration costs of running the service at the centralized cloud or migrating to/from the centralized cloud are set as a constant 2. When the user is inactive, its cost is zero.

We use the error upper bound in the form of $F(T) = \beta T^{\alpha}$, with $\alpha = 1.1$ and the value of β varies in the simulations. The prediction error for each user in each timeslot is generated randomly, while ensuring that the upper bound is satisfied.

The actual costs at different time of the day is shown in Fig. 4, where we set $\beta = 0.2$. The costs are averaged over the 50 users which potentially require cloud services, and the fluctuation is because of different number of users that are active during the day (thus different network load). We can see that the result from the proposed approach (E) performs close to the true optimal result (D), where the optimal result is obtained by assuming that all the future costs are precisely known. The proposed method also outperforms alternative placement methods including never migrating the service (A), always following the user when the user hands-over to a different base-station (B), and always placing the service on the centralized cloud (C). We can also see that the costs from the always-migrate mechanism is closer to the optimum in Fig. 4(a), compared to Fig. 4(b). This is because the above cost definition does not consider the timeslot length[1], and the percentage of timeslots that the user hands-over to a different base-station is smaller when the timeslot length is small.

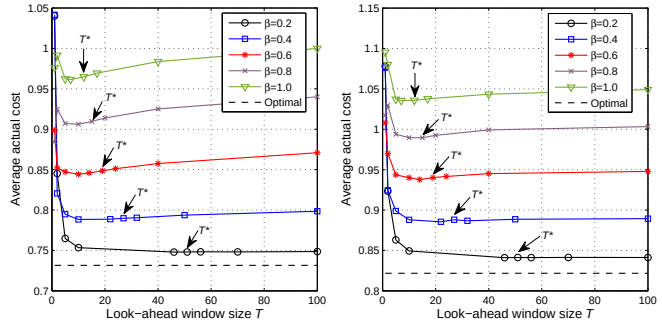
To further investigate the impact of prediction errors and look-ahead window size, we consider the cost when using different window sizes and β values, as shown in Fig. 5, where we run the simulation with 8 different random seeds and plot the average value. It can be seen that the optimal window size (T^*) found from the proposed method is close to the window size that brings the lowest cost, which implies that the proposed method is reasonably accurate. The slight

[1]When the cost is averaged out by the timeslot length, then the results for different timeslot lengths become similar, thus we do not impose averaging in the cost definition so that we can study more diverse cases.



(a) (b)

Figure 4. Average actual costs at different time of a day with $\beta = 0.2$. The time is shown in 24-hour clock. The timeslot lengths are (a) 60 s, (b) 300 s.



(a) (b)

Figure 5. Average actual costs with different look-ahead window sizes and β values. The timeslot lengths are (a) 60 s, (b) 300 s.

deviation from the true optimal window size is due to the randomness of prediction errors, and is also because the value of T^* is found using the upper bounds of prediction errors and cost difference, as discussed in Section IV. We can also see from Fig. 5 that a large prediction error (indicated by a large β) generally causes large deviation from the true optimal cost, which is intuitive. An exception is when T is very small ($T = 1$ or 2). We think that this is because we do not consider the long-term effect with small T, and some random prediction errors may trigger it actually operate closer to the optimum.

VI. Conclusions

In this paper, we have studied the dynamic service placement problem for MMCs. Noting that the actual future costs cannot be obtained precisely, we have proposed a method that utilizes predicted future costs with a known upper bound on the prediction error to make placement decisions that are close to the true optimum. The method includes finding the optimal window size to look ahead as well as finding the optimal placement within each look-ahead window. The service duration $T_{\max}$ in this paper can also be regarded as the maximum size of the look-ahead window, which may be set smaller than the actual service duration in practice, especially for services that run for a very long time. If the service duration is unknown in practice, $T_{\max}$ can also be set as the anticipated service duration, which can be derived from historical data of service usage. As mentioned in Section II, although the problem formulation focuses on a single user,

it can be extended to multi-user scenarios (such as in the simulations in Section V). We also envision that the approach proposed in this paper can be applied to a broader range of problems that share similarities with the service placement problem.

APPENDIX

Proof of Proposition 1: We note that there are $\left\lfloor \frac{T_{\max}}{T} \right\rfloor$ *full* look-ahead windows of size T within timeslots from 1 to $T_{\max}$ (see Fig. 2), where $\lfloor x \rfloor$ denotes the integral part of x. In the last window, there are $T_{\max} - T \cdot \left\lfloor \frac{T_{\max}}{T} \right\rfloor$ slots. We have

$$F\left(T_{\max} - T \cdot \left\lfloor \frac{T_{\max}}{T} \right\rfloor\right) \leq \frac{T_{\max} - T \cdot \left\lfloor \frac{T_{\max}}{T} \right\rfloor}{T} F(T) \quad (13)$$

because $F(T)$ is convex non-decreasing and $F(0) = 0$.

For the true optimal placement sequence $\boldsymbol{\pi}^*$, according to the definitions of $\epsilon(\tau)$ and $F(T)$, the difference in the predicted and actual sum-costs satisfies

$$\begin{aligned} &\sum_{t=1}^{T_{\max}} D_{\boldsymbol{\pi}^*}(t) - \sum_{t=1}^{T_{\max}} A_{\boldsymbol{\pi}^*}(t) \\ &\leq \left\lfloor \frac{T_{\max}}{T} \right\rfloor F(T) + F\left(T_{\max} - T \cdot \left\lfloor \frac{T_{\max}}{T} \right\rfloor\right) \\ &\leq \frac{T_{\max}}{T} F(T) \end{aligned} \quad (14)$$

where the last inequality follows from (13). Similarly, for the placement sequence $\boldsymbol{\pi}_T$ obtained from predicted costs, we have

$$\sum_{t=1}^{T_{\max}} A_{\boldsymbol{\pi}_T}(t) - \sum_{t=1}^{T_{\max}} D_{\boldsymbol{\pi}_T}(t) \leq \frac{T_{\max}}{T} F(T) \quad (15)$$

In the following, we establish the relationship between $\boldsymbol{\pi}^*$ and $\boldsymbol{\pi}_T$. Assume that, in (3), we neglect the migration cost at the beginning of each look-ahead window, i.e. we consider each window independently and there is no migration cost in the first timeslot of each window, then we have

$$\sum_{t=1}^{T_{\max}} D_{\boldsymbol{\pi}_T}(t) \leq \sum_{t=1}^{T_{\max}} D_{\boldsymbol{\pi}^*}(t)$$

This holds because there is no connection between different windows, thus the optimal sequences (considering predicted costs) obtained from (3) constitute the optimal sequence for all timeslots $[1, T_{\max}]$. Now we relax the assumption and consider the existence of migration cost in the first slot of each window. Note that we cannot have more than $\left\lfloor \frac{T_{\max}}{T} \right\rfloor + 1$ windows and the first timeslot $t = 1$ does not have migration cost. Thus,

$$\sum_{t=1}^{T_{\max}} D_{\boldsymbol{\pi}_T}(t) \leq \sum_{t=1}^{T_{\max}} D_{\boldsymbol{\pi}^*}(t) + \frac{T_{\max}}{T} \delta \quad (16)$$

The bound holds because no matter where the service is placed in slot $t_0 - 1$, the migration cost in slot t_0 cannot exceed δ.

By summing up (14) and (16), we get

$$\sum_{t=1}^{T_{\max}} D_{\boldsymbol{\pi}_T}(t) - \sum_{t=1}^{T_{\max}} A_{\boldsymbol{\pi}^*}(t) \leq \frac{T_{\max}}{T} \left(F(T) + \delta\right) \quad (17)$$

Summing up (15) with (17) and dividing both sides by $T_{\max}$ yields Proposition 1. □

ACKNOWLEDGMENT

This research was partly sponsored 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, Z. Chen, K. Ha, W. Hu, W. Richter, and P. Pillai, "Cloudlets: at the leading edge of mobile-cloud convergence," in *Proc. of MobiCASE 2014*, Nov. 2014.

[2] K. Ha, Z. Chen, W. Hu, W. Richter, P. Pillai, and M. Satyanarayanan, "Towards wearable cognitive assistance," in *Proc. of ACM MobiSys*, 2014.

[3] S. Wang, L. Le, N. Zahariev, and K. K. Leung, "Centralized rate control mechanism for cellular-based vehicular networks," in *Proc. of IEEE GLOBECOM 2013*, Dec. 2013.

[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] 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.

[6] Z. Becvar, J. Plachy, and P. Mach, "Path selection using handover in mobile networks with cloud-enabled small cells," in *Proc. of IEEE PIMRC 2014*, Sept. 2014.

[7] A. Fischer, J. Botero, M. Beck, H. De Meer, and X. Hesselbach, "Virtual network embedding: A survey," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 4, pp. 1888–1906, 2013.

[8] M. Chowdhury, M. Rahman, and R. Boutaba, "Vineyard: Virtual network embedding algorithms with coordinated node and link mapping," *IEEE/ACM Transactions on Networking*, vol. 20, no. 1, pp. 206–219, 2012.

[9] A. Ksentini, T. Taleb, and M. Chen, "A Markov decision process-based service migration procedure for follow me cloud," in *Proc. of IEEE ICC 2014*, Jun. 2014.

[10] S. Wang, R. Urgaonkar, T. He, M. Zafer, K. Chan, and K. K. Leung, "Mobility-induced service migration in mobile micro-clouds," in *Proc. of IEEE MILCOM 2014*, Oct. 2014.

[11] M. C. Gonzalez, C. A. Hidalgo, and A.-L. Barabasi, "Understanding individual human mobility patterns," *Nature*, vol. 453, no. 7196, pp. 779–782, 2008.

[12] G. Aceto, A. Botta, W. de Donato, and A. Pescape, "Cloud monitoring: A survey," *Computer Networks*, vol. 57, no. 9, pp. 2093 – 2115, 2013.

[13] E. Cho, S. A. Myers, and J. Leskovec, "Friendship and mobility: User movement in location-based social networks," in *Proc. of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '11, 2011, pp. 1082–1090.

[14] K. LaCurts, J. Mogul, H. Balakrishnan, and Y. Turner, "Cicada: Introducing predictive guarantees for cloud networks," Jun. 2014.

[15] W. B. Powell, *Approximate Dynamic Programming: Solving the curses of dimensionality*. John Wiley & Sons, 2007.

[16] M. Piorkowski, N. Sarafijanovoc-Djukic, and M. Grossglauser, "A parsimonious model of mobile partitioned networks with clustering," in *Proc. of COMSNETS*, Jan. 2009.

[17] M. Piorkowski, N. Sarafijanovic-Djukic, and M. Grossglauser, "CRAWDAD data set epfl/mobility (v. 2009-02-24)," Downloaded from http://crawdad.org/epfl/mobility/, Feb. 2009.