Emulation-Based Study of Dynamic Service Placement in Mobile Micro-Clouds

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

***Abstract*—Tactical networks are highly constrained in communication and computation resources, particularly at the edge. This limitation can be effectively addressed by the emerging technology of mobile micro-clouds (MMCs) that is aimed at providing seamless computing/data access at the edge of such networks. Deployment of MMCs can enable the delivery of critical, timely, and mission relevant situational awareness to end users in highly dynamic environments. Different from traditional clouds, an MMC is smaller and deployed closer to users, typically attached to a fixed or mobile basestation that is deployed in the field. Due to the relatively small coverage area of each basestation, a mobile user may frequently switch across areas covered by different basestations. An important issue therefore is where to place the service so that acceptable service performance can be maintained, while coping with the user and network dynamics. Existing work has considered this problem mainly from a theoretical angle. In this paper, with the aim of pushing the theoretical results one step closer to practice, we study the performance of dynamic service placement using an emulation framework, namely the Common Open Research Emulator (CORE) which embeds the Extendable Mobile Ad-hoc Network Emulator (EMANE). We first present the system architecture used in the emulation. Then, we present the message exchange and control process between different network entities, as well as methods of deciding where to place the services. Finally, we perform emulation using real-world user mobility traces of San Francisco taxis and present the results. The results show several insightful observations in a realistic network setting, such as the impact of randomness and delay on the service placement performance.**

***Index Terms*—Cloud technologies, edge computing, emulation, mobility, optimization, wireless networks**

I. Introduction

Tactical networks are required to provide efficient and effective computing capability at the tactical edge. As described in [1], there is an increasing need for enhanced information services and associated infrastructure, and mobile cloud computing can enhance mission performance by providing users on-demand data and services in dynamic network environments. Typical cloud applications consists of front-end components running on the mobile devices carried by users and back-end components that perform the majority of computation running on the cloud. With this architecture, applications can take advantage of the on-demand feature of cloud computing. However, new challenges are also introduced, such as increase in network load and latency.

These challenges can be addressed by moving computation closer to the users. It is envisioned that small cloud computing platforms will be distributed across the network, hosted by entities at the network edge (such as fixed or mobile basestations deployed on the field), as shown in Fig. 1. We refer to this distributed micro-cloud infrastructure as *Mobile Micro-Cloud (MMC)* in this paper, which is also known as Edge Computing [2], Cloudlets [3], Follow Me Cloud [4], etc. MMCs are expected develop rapidly as the data and computation requirements of users at the tactical edge increase. Furthermore, they can also improve the robustness of cloud computing systems in hostile environments [3].

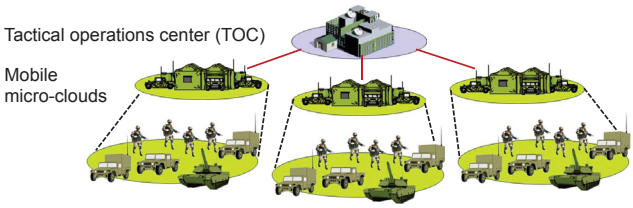

Figure 1. Application scenario of mobile micro-clouds (MMCs).

One important aspect in MMCs is to determine the location (in terms of which cloud) of running the services, where services for different users may be run at different locations and the users are usually mobile. Both MMCs and centralized core cloud(s) can host services, and the selection of service locations depends on several factors. There usually exists ongoing data transmission between each user and the cloud that is hosting its service, which incurs a *transmission cost*. There is also a *migration cost* associated with migrating a service from one cloud to another cloud. In practice, these costs can be related to communication bandwidth consumption, delay in accessing the service, interruption of service, etc. Migration is needed because the best service location may change over time due to user mobility. A trade-off exists between the transmission and migration costs, both of which are related to factors including the distance of data transmission and migration, load of the cloud and the network, etc.

The problem of dynamic service placement/migration aims at adaptively mapping services to clouds to minimize the total transmission and migration cost. It is a highly challenging problem particularly due to user mobility, and has attracted notable interest recently. Most existing work take a theoretical approach to this problem, using mathematical tools ranging from Markov decision processes (MDPs) [5]–[7], dynamic programming [8], and Lyapunov optimization [9]. While these theoretical results provide useful insights into the problem, the performance of these algorithms under a practical setting remains to be investigated.

In this paper, we aim to push the theoretical results one step closer to practice. We present a framework for conducting experiments in an emulated cloud environment that contains one centralized core cloud (the tactical operations center (TOC) in Fig. 1) and multiple MMCs, which jointly cover an area that contains multiple mobile users. The emulation is performed using the Common Open Research Emulator (CORE) [10], [11] which embeds the Extendable Mobile Ad-hoc Network Emulator (EMANE) [12]. Implementing algorithms in CORE enables experimentation with greater realism than simulation, thus providing insight into what behavior or performance may be expected upon actual deployment of these algorithms.

The emulation-based study we consider in this paper has the following key differences from the theoretical work in [5]–[9]:

1) In the emulation, the parameters related to the transmission and migration costs are estimated based on real-time network measurements, which can be subject to fluctuations and inaccuracies.
2) A fully distributed system is considered in the emulation. No node (i.e., core cloud, MMC, or user) in the network has full control over other nodes. They can only communicate with each other by exchanging control messages over the communication network.
3) The emulation considers realistic communication links that may experience loss and delay as a result of congestion and/or physical layer effects. Control messages may also be delayed or lost.

The remainder of this paper is organized as follows. We first give an overview to the existing theoretical results in Section II. The system architecture that we use in the emulation is presented in Section III. In Section IV, we present the process of control message exchange that is used for performance measurement and controlling the placement of services; we also introduce the methods that we use for placement decision in the emulation. The emulation is performed using real-world user mobility traces of San Francisco taxis, and results are shown in Section V. Section VI draws conclusions.

II. Overview of Theoretical Results

In this section, we summarize the existing results on the theoretical modeling and analysis of dynamic service placement/migration in MMCs. Starting with the work in [5], the problem of dynamic service migration and workload scheduling has been studied under increasingly complex settings. The overall objective is to develop a fundamental understanding of this decision problem and the associated tradeoffs.

A class of existing work aims at solving this problem using MDPs [5]–[7]. The work in [5] formulated the service migration problem for a single user with a one-dimensional (1-D) mobility and a specific cost function, and the solution is found with standard MDP solution approaches such as value and policy iteration. We note that these standard solutions can become time consuming when the number of states in the MDP is large. Therefore, simplified solution approaches and approximate solutions are proposed in [6], [7].

The work in [6] considers the problem of service migration for a single user with a 1-D asymmetric random walk mobility model, where the transmission and migration costs are assumed to be constants respectively when transmission or migration occurs, and zero otherwise. This problem is formulated as an MDP, and the optimal policy for service migration is shown to be a threshold policy, in which the service is migrated to the closest MMC to the user when the offset between the service and user locations exceeds a threshold. Further, an analytical solution of the cost is derived for any given thresholds, based on which an algorithm is proposed to find the optimal thresholds. It is shown that the proposed algorithm is more efficient than standard mechanisms for solving MDPs.

The work in [7] extends the formulation in [6] to consider more general two-dimensional (2-D) mobility. A more general cost function, which is in a constant-plus-exponential form, is also considered. To reduce the computational complexity, the state space of the MDP is approximated by the distance between the user and service locations. This approximation is shown to be exact for uniform 1-D mobility and closely approximate uniform 2-D mobility with a constant additive error term. The proposed algorithm is significantly faster than traditional MDP solution for 2-D mobility.

In all the above works, the decision problem is formulated as an MDP. The solutions are relatively straightforward to implement. However, the MDP formulation requires that the user mobility has to follow (or at least can be approximated by) a Markov chain, and that the transition probabilities of this chain is known. Moreover, the methods for simplifying and approximating the MDP only work for a specific class of cost functions.

To overcome some of these drawbacks, an alternative approach is taken in [8], where a mechanism for dynamic service placement utilizing future costs that can be predicted with known accuracy is proposed. In contrast to slot-based decisions in the MDP approaches, the solution in [8] makes decisions on the basis of a look-ahead window, whose size is optimized as a function of the prediction accuracy. Within the look-ahead window, placement decisions are made using dynamic programming based on the predicted costs. This method can work with more general cost functions, as long as future costs are predictable with known accuracy.

Finally, [9] considers a more general problem involving both service migration and workload scheduling. This joint problem is still formulated as a constrained MDP for which traditional solution methods (such as dynamic programming) require extensive statistical knowledge and are computationally prohibitive. Instead, a novel alternative methodology is developed. First, an interesting decoupling property of the MDP is established, which reduces it to two independent MDPs on disjoint state spaces. Then, using the technique of Lyapunov optimization over renewals, an online control algorithm is designed for the decoupled problem that is provably cost-optimal. This algorithm does not require any statistical knowledge of the user mobility or demand model and can be implemented efficiently. However, the user requests usually need to be queued before they can be served by the MMCs.

The ultimate goal of the emulation study we consider in

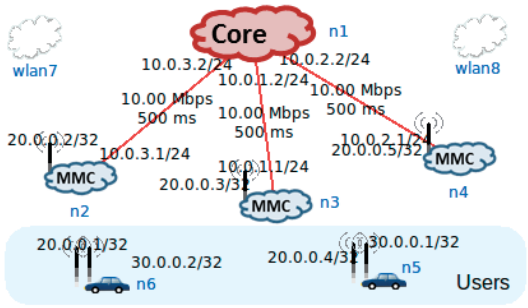

Figure 2. System architecture for CORE emulation.

this paper is to evaluate the above theoretical findings in realistic settings where some assumptions made for theoretical tractability may not hold. We propose an initial emulation framework in this paper, and perform the emulation with some simple algorithms. This sets the foundation of emulating more sophisticated algorithms in the future.

III. System Architecture

In the CORE emulation, we abstract the application scenario depicted in Fig. 1 into the system architecture shown in Fig. 2. In Fig. 2, node `n1` is the core cloud, which is located in the TOC in Fig. 1; nodes `n2`, `n3`, and `n4` are MMCs; and nodes `n5` and `n6` are users. Note that this is only an example and the number of MMCs and users can be arbitrary in the emulation.

A. Network Connection and User Mobility

The core cloud is connected with each MMC via a satellite link that is emulated by a persistent link configured with a relative low bandwidth and a relatively large delay.

Each MMC and each user has a wireless interface, which has an IP address of `20.0.0.x` and is configured by the configuration node `wlan7` in Fig. 2. Throughout the emulation, we use a fixed range radio propagation model. A pair of nodes within the specified range can communicate with each other, and they cannot communicate if they are outside the specified range, where a node can be either a user or an MMC. More realistic propagation models can be considered in the future using functionalities of EMANE. The OSPF-MDR routing protocol [13] is used for routing among the wireless nodes. A pair of nodes may communicate either in single-hop or multi-hop depending on their distance.

We consider in the emulation the case where MMCs are static and users are mobile. This is only to ease the emulation setup, and the same design principle can be applied to scenarios where MMCs are also mobile. We employ the EMANE event service to use existing traces to govern the mobility of users. The mobility is described by an Emulation Event Log (EEL) file. To receive messages from EMANE, each user has an additional wireless interface with an IP address of `30.0.0.x`, which is configured by the configuration node `wlan8`. This interface is used only for communicating with EMANE (which is executed outside the emulator in the emulator's host machine) so that the user locations remain updated in real time according to the trace file. It does not participate in any actual communication in the emulated network.

With the above setting, the core cloud, MMCs, and users can communicate with each other. The wireless connections are also updated in real-time based on locations of users. Users do not have a direct connection with the core cloud, but they can connect to the core via an MMC.

B. Service Model

We consider a relatively simple service model as presented next, while noting that more sophisticated cases can be built based on the emulation framework presented in this paper.

We assume that each user is running one service instance in the cloud, and the terms "service" and "service instance" are exchangeably used in this paper. We consider a delay-sensitive situational awareness application, where each user regularly sends data (e.g., images) describing its current surrounding to the cloud that is hosting its service. The cloud analyzes user data to extract user situation (e.g., detecting and tagging objects and faces in images) and sends the results back to the user, also in a regular manner. We abstract this process as UDP packet transfer between the user and the cloud, and we name these packets as *service packets* to distinguish them from control packets introduced later. The service packet transmission is initiated by the cloud. The cloud sends a `MMC_SERVICE_PACKET` to each user (for which it is hosting a service) at a pre-specified interval. After receiving this packet, the user responds with its own `USER_SERVICE_PACKET` that contains newly measured data to the cloud.

The performance metric of the service is the round-trip delay of transmitting service packets from the cloud to the user and then back to the cloud. This mimics situational awareness applications that need to collect user data as quickly as possible so that rapid decisions can be made. This round-trip delay can be seen as the transmission cost between the user and the cloud.

For simplicity, we assume that service migration can be completed instantaneously. We record the number of migrations in the emulation results in Section V, which is an indicator of the migration cost.

We assume that services can only be run on MMCs and no service is run on the core cloud. This can be justified by noting that it can be inefficient to run delay sensitive applications on the core cloud due to the large communication delay. It also simplifies the control procedure of service placement. The role of the core cloud is to act as a controller for the service placement, which is a relatively robust setup for a centralized control approach because the system can still work even if some MMCs fail to function. The centralized control approach is also used in the theoretical work in [5]–[9].

IV. Packet Exchange and Placement Control

A. Control Messages

Control messages need to be exchanged for performance measurement and controlling the placement of services. All messages are sent in UDP packets. The different types of messages are described as follows.

Algorithm 1 Procedure at the core cloud

```
loop
    if timer t(CORE_BEACON_MMC) expired then
        Update beacon information, such as the placement of services for
        all users (obtained based on the recorded delay measurements)
        Send CORE_BEACON_MMC to each MMC
        Reset timer t(CORE_BEACON_MMC)
    if timer t(CORE_BEACON_USER) expired then
        Update beacon information
        Send CORE_BEACON_USER to each user
        Reset timer t(CORE_BEACON_USER)
    if received MMC_USER_DELAY_MEASURED then
        Update the recorded delay statistics
    if received MMC_CONNECT or USER_CONNECT then
        Update the MMC or user record
```

1) Beacon Messages from Core Cloud: At a pre-specified interval, the core cloud sends out a beacon message CORE_BEACON_MMC to each MMC, and it also sends out a beacon message CORE_BEACON_USER to each user. This is to notify MMCs and users of some status information, including the set of users each MMC should currently serve, the set of users each MMC should probe for delay measurement, etc. Before sending these beacon messages, the core cloud makes decisions such as where to place the service for each user.

2) Connection Request: In order for the core cloud to know which MMCs and users are currently present in the system, each MMC sends MMC_CONNECT and each user sends USER_CONNECT at a pre-specified interval.

3) Delay Probing: Probing for delay measurement is initiated by each MMC at a pre-specified interval. Every MMC probes a set of users as instructed by the core cloud. This set of users should be within a specific proximity of the MMC, so that the MMC is potentially suitable of running services for these users. An MMC first sends MMC_DELAY_PROBE to all the users it intends to probe. Upon receiving this message, each user immediately replies with a USER_DELAY_PROBE to the MMC that sent this message. The MMC calculates the round-trip delay for the particular user and sends the result to the core cloud with an MMC_USER_DELAY_MEASURED message. After the core cloud has received this delay information, it stores it in a list so that it can be used for making service placement decisions before sending beacon messages.

We note that one may think of using service packets for delay measurement. However, service packets are only transmitted between a user and the particular MMC that is hosting its service, and the delay of transmitting packets between the user and other MMCs cannot be obtained from service packets. Therefore, we introduce additional packets for delay probing, which is usually substantially shorter than service packets.

B. Packet Exchange and Control Procedure

The detailed procedures that are executed at the core cloud, MMC, and user are respectively shown in Algorithms 1, 2, and 3, where we use the notation $t(\texttt{A})$ to denote the timer for packet A, the packet A is sent when $t(\texttt{A})$ expires.

C. Service Placement Decisions

In the emulation, we consider three different policies for deciding the service placement. These are described as follows.

Algorithm 2 Procedure at each MMC

```
loop
    if timer t(MMC_CONNECT) expired then
        Send MMC_CONNECT to the core cloud
        Reset timer t(MMC_CONNECT)
    if timer t(MMC_DELAY_PROBE) expired then
        Send MMC_DELAY_PROBE to each user it intends to probe
        Reset timer t(MMC_DELAY_PROBE)
    if timer t(MMC_SERVICE_PACKET) expired then
        Send MMC_SERVICE_PACKET to each user it is serving
        Reset timer t(MMC_SERVICE_PACKET)
    if received USER_DELAY_PROBE then
        Calculate the round-trip delay and send the result via
        MMC_USER_DELAY_MEASURED to the core cloud
```

Algorithm 3 Procedure at each user

```
loop
    if timer t(USER_CONNECT) expired then
        Send USER_CONNECT to the core cloud
        Reset timer t(USER_CONNECT)
    if received t(MMC_DELAY_PROBE) then
        Send USER_DELAY_PROBE to the originating MMC
    if received t(MMC_SERVICE_PACKET) then
        Send USER_SERVICE_PACKET to the originating MMC
```

1) Always Migrate (AM): In the AM policy, the core cloud always looks at its most recently received delay measurements, and places the service for each user to the MMC that has the lowest delay as measured by the latest probe. This policy puts strong emphasis on minimizing the transmission cost (round-trip delay), and does not consider the migration cost (number of migrations).

2) Infrequently Migrate (IM): The IM policy is similar to the AM policy, except that migration can only occur at an interval denoted by τ. The value of τ is normally large compared to the delay probing interval, so that migration happens infrequently. The initial placement of services is not restricted by τ, which means that if a user is not served by any cloud previously, its service can be placed immediately after the core cloud (controller) recognizes that it has established connection with an MMC. The IM policy attempts to bound the migration cost, while the transmission cost may be large because services may not be migrated soon enough after the transmission delay has changed.

3) Moving Average + Hysteresis (MAH): The MAH policy does not use the instantaneous delay measurement for service placement and migration decisions. Instead, it uses an exponentially-weighted moving averaged delay for decision making. For each MMC i and user j, the moving average is performed according to

$$D_{ij}(k) = \alpha D_{ij}(k-1) + (1-\alpha) d_{ij}(k) \tag{1}$$

where $d_{ij}(k)$ is the kth round-trip delay for MMC-user pair (i, j) that has been received by the core cloud, $D_{ij}(k)$ is the delay used for decision making after receiving the kth and before receiving the $(k+1)$th measurement, $0 \leq \alpha \leq 1$ is a controllable parameter for moving average computation.

The MAH policy also has a hysteresis delay value $\epsilon \geq 0$. If the service for user j is previously running at MMC i, a migration to MMC i' only occurs when $D_{i'j}(k) < D_{ij}(k) - \epsilon$. This prevents frequent migration between different MMCs.

We can see that the MAH policy can be configured to

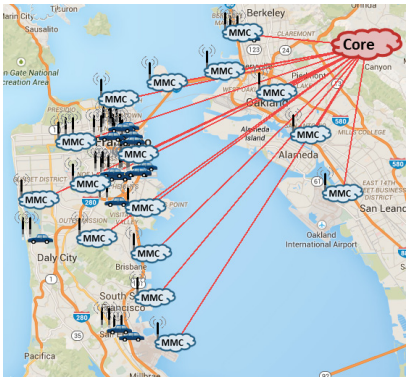

Figure 3. Emulation scenario (source of map: https://maps.google.com/).

Table I
EMULATION SETUP

Parameter name	Value
Wireless communication bandwidth	10 Mbps
One-hop wireless communication range	6,000 m
One-hop wireless communication delay	20 ms
Bandwidth of link connecting MMC and core cloud	10 Mbps
Delay of link connecting MMC and core cloud	500 ms
Size of `MMC_SERVICE_PACKET`	1,000 Bytes
Size of `USER_SERVICE_PACKET`	50,000 Bytes
Interval of sending `MMC_SERVICE_PACKET`	2 s
Interval of sending `MMC_CONNECT`	10 s
Interval of sending `CORE_BEACON_MMC`, `CORE_BEACON_USER`, and `MMC_DELAY_PROBE`	T (variable)
Parameter τ in IM policy	100 s
Parameter α in MAH policy	0.5
Parameter ϵ in MAH policy	10 ms

achieve various trade-offs between the transmission and migration costs by tuning the values of α and ϵ. It is used here as an alternative representative of the theoretical results in [5]–[9], which jointly consider the transmission and migration costs. The MAH policy is simpler than the algorithms in [5]–[9], but also less robust due to the existence of parameters α and ϵ that is subject to tuning. We also note that the AM policy is a special case of the MAH policy under $\alpha = \epsilon = 0$.

V. Emulation Scenario and Results

We created an emulation scenario that consists of 20 users and 16 MMCs distributed in the San Francisco area, as shown in Fig. 3. We assume that the MMCs are static. The user mobility is generated using the taxi cabspotting dataset [14], which is a dataset of 536 taxis with time (with minute resolution), latitude, longitude, altitude, and occupancy data for each taxi over approximately a 30 day period. To reduce the required emulation time and see the impact of user mobility, we compressed the timescale by a factor of 6, so that 6 seconds in the original trace is one second in the emulation.

We ran the emulation for 15,000 seconds, in which we applied either the AM, IM, or MAH policy for service placement. The interval of sending `CORE_BEACON_MMC`, `CORE_BEACON_USER`, and `MMC_DELAY_PROBE` is specified by parameter T which take different values in the emulation. The value of T specifies the interval of delay probing and service placement update. Other parameter settings in the emulation are summarized in Table I.

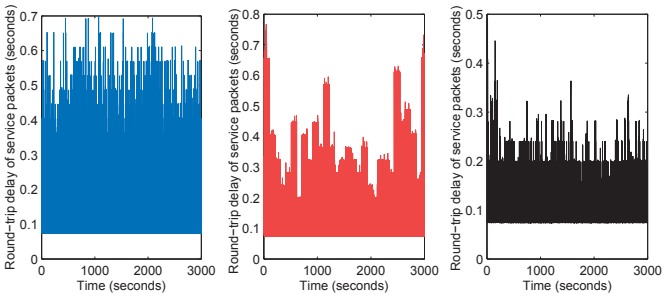

Figure 4. Instantaneous round-trip delays of service packets for the first 3,000 s of emulation with $T = 2$ s: (a) AM policy, (b) IM policy, (c) MAH policy.

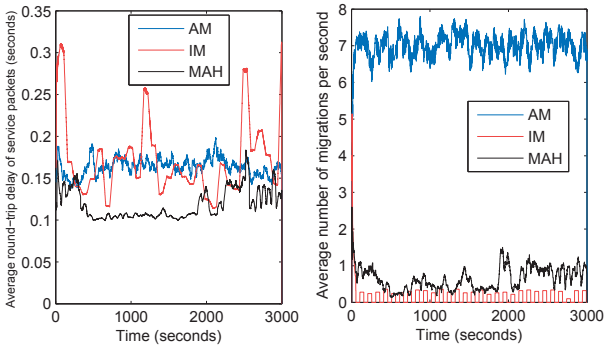

Figure 5. Moving average results for the first 3,000 s of emulation with $T = 2$ s: (a) round-trip delay of service packets, (b) number of migrations.

We first set $T = 2$ s and focus on the instantaneous results in the first 3,000 seconds of emulation. The instantaneous round-trip delay of service packets under different policies is shown in Fig. 4, where the results for service packets corresponding to different MMC-user pairs are merged into the same timeline and plotted in one graph. We can see that there are many spikes in the figure, which indicates that the delay exhibits large variation. Reasons for this include variation in network traffic, diversity of user and service locations, as well as the change of user and service locations over time. The last reason is obvious from Fig. 4(b), where the delay has a block pattern because the IM policy only migrates (and thereby changing service locations) once in 100 seconds.

To obtain a more comprehensible set of results, we perform cumulative moving average with a window size of 50 s on the instantaneously measured data. At each time instant (in the resolution of one second), we look back to the past 50 s (or up to the emulation start time, whichever is later) and plot the average result within this window. The moving averaged delay is shown in Fig. 5(a), and Fig. 5(b) shows the average number of service migrations within the 50 s window size. We can see that compared to the AM and MAH policies, the IM policy may cause blocks of large delays, because services may be placed at non-optimal locations for a long time.

The overall performance with different values of T is shown in Fig. 6, where the results are collected starting from 1,000 s emulation time to remove the impact of variation in the initialization time under different T. We see that as expected,

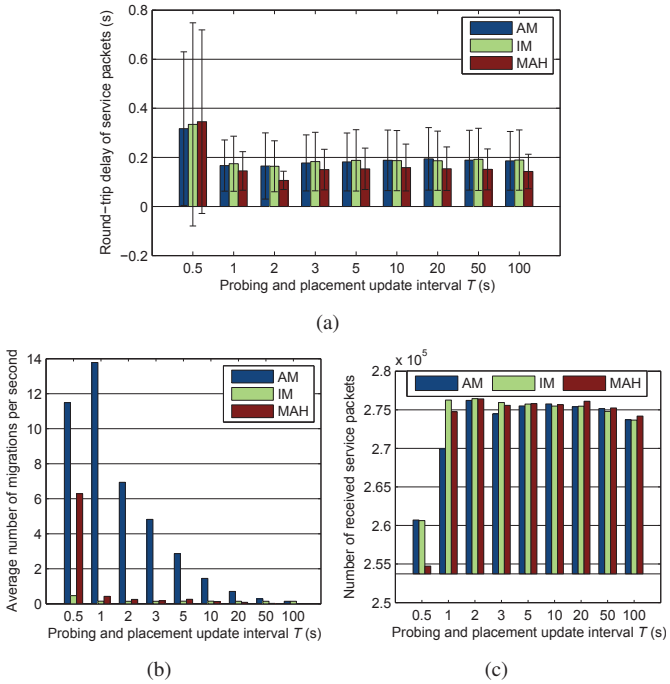

Figure 6. Overall results: (a) average round-trip delay of service packets (error bars denote the standard deviation), (b) average number of migrations per second, (c) total number of received service packets.

the AM policy always has the largest number of migrations. It is interesting that round-trip delay of the AM policy is also generally larger than that of the MAH policy, mainly because the AM policy uses instantaneous delay measurements for service placement decisions, which may fluctuate substantially over time and cause the placement decision deviate away from the optimum. Instead, the MAH policy is more stable because it has the moving average and hysteresis building blocks. The delay performance of the MAH policy is also generally better than the IM policy, because services are migrated to good locations more frequently.

The performance is related to the value of the interval T. It is clear that $T = 0.5\,\mathrm{s}$ brings the worst performance, in which case the control messages overload the network. For the MAH policy, its lowest delay and highest number of received service packets (which represents the packet success rate) is attained at $T = 2\,\mathrm{s}$, while giving an intermediate number of migrations. We also note that the average delay of the MAH policy is the lowest among all the policies when $T = 2\,\mathrm{s}$, also with lower standard deviation than all other cases. This implies that the MAH policy can be beneficial for delay-sensitive applications. When T is large, the delay performance does not vary significantly with different T, but the number of received service packets decreases with T. The network in this case is not overloaded, thus the delay is not very large. However, due to obsolete delay measurement and prolonged service placement update, the service location may be far away from user location, so that the connection between the MMC and the user has multiple hops, causing higher packet loss. The slight fluctuation in the performance with different values of T is due to randomness in the emulation.

VI. Conclusions

In this paper, we have taken an initial step to an emulation-based study of service placement/migration in an MMC environment containing mobile users. We have proposed a simple emulation framework that can be used as a foundation for more sophisticated emulations in the future. The emulation results show several insightful observations, such as the impact of randomness and delay on the service placement performance in a realistic network setting. It is worthwhile to carry out an in-depth future study on these impacts and their solutions from both a theoretical and practical point of view.

Acknowledgment

The authors thank Rommie Hardy for technical assistance in the CORE/EMANE software.

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] T. Takai, "Cloud computing strategy," U.S. Department of Defense, Chief Information Officer, Tech. Rep., 2012.

[2] 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.

[3] M. Satyanarayanan, G. Lewis, E. Morris, S. Simanta, J. Boleng, and K. Ha, "The role of cloudlets in hostile environments," *IEEE Pervasive Computing*, vol. 12, no. 4, pp. 40–49, Oct. 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] 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.

[6] 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.

[7] S. Wang, R. Urgaonkar, M. Zafer, T. He, K. Chan, and K. K. Leung, "Dynamic service migration in mobile edge-clouds," in *Proc. of IFIP Networking 2015*, May 2015.

[8] S. Wang, R. Urgaonkar, K. Chan, T. He, M. Zafer, and K. K. Leung, "Dynamic service placement for mobile micro-clouds with predicted future cost," in *Proc. of IEEE ICC 2015*, Jun. 2015.

[9] R. Urgaonkar, S. Wang, T. He, M. Zafer, K. Chan, and K. K. Leung, "Dynamic service migration and workload scheduling in edge-clouds," *Performance Evaluation*, vol. 91, pp. 205–228, Sept. 2015.

[10] J. Ahrenholz. (2010) CORE download. [Online]. Available: http://downloads.pf.itd.nrl.navy.mil/core/

[11] ——, "Comparison of core network emulation platforms," in *Proc. of IEEE MILCOM 2010*, Oct.-Nov. 2010, pp. 166–171.

[12] N. Ivanic, B. Rivera, and B. Adamson, "Mobile ad hoc network emulation environment," in *Proc. of IEEE MILCOM 2009*, Oct. 2009.

[13] R. Ogier and P. Spagnolo. (2009, Aug.) RFC 5614, Mobile ad hoc network (MANET) extension of OSPF using connected dominating set (CDS) flooding. [Online]. Available: http://www.ietf.org/rfc/rfc5614.txt

[14] M. Piorkowski, N. Sarafijanovoc-Djukic, and M. Grossglauser, "A Parsimonious Model of Mobile Partitioned Networks with Clustering," in *Proc. of COMSNETS*, Jan. 2009.