Reducing Operational Costs in Cloud Social TV: An Opportunity for Cloud Cloning

Yichao Jin, Yonggang Wen, *Senior Member, IEEE*, Han Hu, and Marie-Jose Montpetit, *Senior Member, IEEE*

***Abstract*—The emergence of social TV has transformed TV experiences, providing a unified media experience across different devices. In response to this trend, we have implemented a multi-screen social TV system, offering video teleportation as an attractive feature. The enabling technology is instantiating a cloud clone to support all media outlets of each user. As the user shifts his attention from one device to the other, the cloud clone might migrate to a better location to reduce its operational cost. This paper investigates this cloud clone migration problem, aiming to minimize the monetary cost on operating video teleportation. Specifically, we formulate it into a Markov Decision Problem, to balance the trade-off between the migration cost and the content transmission cost. Under this framework, four algorithms are proposed to solve this optimization problem. We first characterize an upper and a lower bound for the optimal cost, by considering a random fixed placement and an offline algorithm. We then present a semi-online and a more practical Q-learning approach to make online decisions. Their performances are evaluated based on both simulated and real user traces. The results show that the Q-learning method achieves up to 25% cost compared to random fixed placement in typical scenarios. The savings are affected by the delivery path length, the migration size, and the user behavior pattern. Moreover, our investigations reveal the optimal cloud clone location is either at the nearest or the furthest node to the user along the content delivery path for a single user scenario.**

***Index Terms*—Cloud clone, cost minimization, markov decision process, Q-learning, social TV.**

I. Introduction

LATELY, TV experience have been dramatically transformed, with the emergence of multi-screen social TV [1], [2]. First, a traditional "laid-back" video watching experience is combined with "lean-forward" social interactions among peer viewers, resulting in an user-centric viewing environment [1]. Second, multi-screen social TV offers ubiquitous and unified services that are available at anytime, anywhere, on any device at an affordable cost, with personalized experiences [2]. Finally, with the latest multi-screen or second-screen technology, users can transfer the ongoing sessions from one device to another, without any service interruption. Nonetheless, given its highly regarded value, large-scale deployment of social TV has been limited, if not totally absent.

In response to this trend, we have designed and implemented a multi-screen social TV system [3], [4] over a Cloud-Centric Media Network (CCMN) [5]. It offers *video teleportation* as its salient feature. In particular, one can easily migrate video session back and forward among different devices, with intuitive human-computer interactions. As a result, a seamless multi-screen social experience is achieved.

The enabling technology for *video teleportation* is to instantiate a virtual machine in the cloud as a cloud clone [4] for each user.[1] Specifically, each cloud clone represents one user, serving as his proxy in the cloud, to manage all the associated devices and session information. In addition, the cloud clone also provides video transcoding [6] and advertisement insertion function to the original video streaming, achieving a personalized multi-screen social TV experience for end-users in a scalable and flexible manner.

One critical design objective is to minimize the monetary cost on operating *video teleportation*, potentially making this service affordable to the general public. A possible deployment scenario is to rent cloud resources from a vendor-neutral provider. In this case, we need to intelligently manage the resource rental cost. In particular, the transmission cost is determined by the transmitted content size and its delivery path length. The former one depends on the video source and the way the user consumes contents (e.g., TV or smartphone). The latter one depends on the cloud clone location along the content delivery path. As the user shifts sessions from one device to another, the retargeted content size changes, and the cloud clone might dynamically migrate to a better location to save the cost. Therefore, an optimal cloud clone migration policy should be in place to balance the trade-off between the migration cost and the transmission cost savings.

In this paper, our contributions are multi-folder, including:

- We formulate the cost-minimization problem as a Markov Decision Process by adopting a Markov chain to model the user watching behavior across TV and smartphone. The objective is to minimize the monetary cost of operating the video teleportation service, by migrating the cloud clone to the best place, as the user shifts his device.
- Under this framework, we first consider a random fixed placement policy and an offline policy as an upper and a lower bound for the optimal cost. Then we follow up with a semi-online algorithm, where only the state transition

Manuscript received December 05, 2013; revised March 23, 2014 and June 01, 2014; accepted June 03, 2014. Date of publication June 05, 2014; date of current version September 15, 2014. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Sheng-Wei (Kuan-Ta) Chen.

Y. Jin and Y. Wen are with the School of Computer Engineering, Nanyang Technological University, Singapore 639798, Singapore (e-mail: yjin3@ntu.edu.sg; ygwen@ntu.edu.sg).

H. Hu is with the School of Computing, National University of Singapore, Singapore 119077, Singapore (e-mail: huh@comp.nus.edu.sg).

M.-J. Montpetit is with the Comparative Media Studies Department, Massachusetts Institute of Technology, Cambridge, MA 02139 USA (e-mail: mariejo@mit.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TMM.2014.2329370

[1]We will use virtual machine and cloud clone interchangeably in this paper.

probability is known in advance. Finally, we propose the Q-learning method, which learns the user behavior from the history and makes online decisions at each time slot. This approach is more practical in real system.

- Extensive experiments based on both simulated data and real user behaviors traces are conducted to evaluate those proposed strategies. Our numerical results suggest that, up to 25% monetary cost compared with the random fixed placement can be saved by using Q-learning method in typical scenarios. The cost saving can be affected by the length of content delivery path, the data size of VM migration, and the user behavior pattern.

These insights would offer operational guidelines to deliver cost effective multi-screen social TV services over CCMN, potentially easing its adoption.

The rest of the paper is organized as follows. Section II outlines the related works. Section III presents the system architecture and formulates the problem. Section IV proposes four alternative methods to derive the optimal policy. Section V evaluates the performances of different approaches. Finally, Section VI concludes this paper.

II. Related Works

This section surveys the existing literatures on multi-screen technologies, adaptive video streaming, and cost minimization studies in the context of cloud based media systems.

A. Multi-Screen Technologies

Recently, an increasing number of novel systems are proposed to provide multi-screen experiences with different enabling technologies. Deep shot [7] was proposed as a general framework to transfer ongoing sessions across multiple screens. *Lu et al.* [8] presented a virtualized screen solution, which used a cloud clone scheme to complete screen rendering in the cloud. In this way, the screen display can be delivered as a series of images to the thin clients. *Wu et al.* [9] described their design on a cloud based mobile social TV system. Its key module was a virtual machine based surrogate in the cloud, which provided video transcoding services and segmenting the streaming traffic for each user. *Wang et al.* [10] adopted a similar scheme to instantiate a private agent in the cloud for each user, providing adaptive video streaming and social networking services.

Our work differs from these researches in several aspects. First, we built CCMN as a novel framework to integrate a variety of media services, and elastically provide them to users. Second, our cloud clone offers the richest set of functions, including video transcoding, session synchronizing, and video teleportation. As a result, users can seamlessly transfer ongoing sessions among heterogeneous media outlets instantly. Finally, none of those works aimed to improve the system by optimizing the operational cost, while we focus on a cost minimization problem via cloud clone migration.

B. Adaptive Video Streaming Solutions

Adaptive video streaming, due to the its high practical values to efficiently distribute video contents to different end-users, has become an active research topic. *Bernaschi et al.* [11] presented a transcoding-based adaptive streaming scheme for heterogenous networks. They implemented a testbed to show their solution is practical and efficient in real scenarios. *Shen et al.* [12] proposed a transcoding-enabled caching scheme for content distribution networks, where the content adaptation is performed at the network intermediaries. In [13], the authors discussed the techniques for media and streaming strategy adaptation, and indicated placing transcoding service in proxy servers as one of the typical streaming systems.

The main difference between our work and these researches is that, we aim to find the optimal location to transcode the video streaming, while none of them considered such problem.

C. Cost Minimization Techniques

Similar cost minimization problems on cloud based systems have been addressed by several previous researches. *Armbrust et al.* [14] listed the data transfer cost as one of the top obstacles and opportunities for cloud computing. It indicated that, an appropriate virtual machine placement policy is important to reduce the operational cost. *Jaime et al.* [15] examined the energy tradeoff between video transportation and processing for view video streaming, and obtain the optimal video processing location with the lowest energy cost. In [16], the authors balanced the tradeoff between content transmission cost and content storage cost in cloud centric media network, to find the optimal content placement strategy with minimal monetary cost. *I. Zhovnirofsky et al.* [17] described a performance enhancement (PE) scheme, where each application is supported by a PE. Each PE locates at one server, and it can be turned on or off on demand. In this way, this mechanism aims to improve the quality of service and reduce the cost.

However, none of those works can be directly applied to our problem. First, the optimization metrics are different. Most of them targeted at reserving energy consumption, while we aim to minimize the monetary cost by using cloud resources. Second, those works did not take multi-screen experiences into account. Finally, they mostly aimed at an optimal fixed location for each virtual machine, whereas we need an optimal dynamic cloud clone migration scheme with online scheduling.

III. System Overview & Problem Formulation

This section first introduces the system implementations. Then we focus on three system models. Finally, we formulate the minimum-cost cloud clone migration problem as a Markov Decision Process. For clarity and ease of reference in the discussion, we summarize the important notations in Table I.

A. System Implementation

In this subsection, we first present the system architecture and its key application (i.e., video teleportation). Then we discuss the detailed design of cloud clone.

1) System Architecture: In Fig. 1, we present a systematic end-to-end view of our cloud multi-screen social TV system. Specifically, the system is built upon CCMN [5], which provides on-demand media services, including content distribution, media processing and content adaption. End users with different devices are connected via residential gateways and access networks to the cloud. They can request a live or on-demand TV program from an IPTV source through the cloud via virtual

TABLE I
NOTATION TABLE

Symbol	Definition
p_t	The probability that an user uses TV in both the current and the next time slot
p_p	The probability that an user uses phone in both the current and the next time slot
L	Content delivery hops from source to user
B_o	Expected original content size
B_t	Expected retargeted content size (B_t^t for TV, B_t^p for phone)
$\bar{B}_t(x)$	Mean expected retargeted content size during a period x
c_{tr}	Per hop price to transmit per GB data
$C_{tr}(k)$	Transmission cost when cloud clone locates at k hops away to the source
V_{mig}	Migrated data volume
c_m	Additional price to complete each migration
$C_{mig}(k,k')$	Migration cost from node k to node k'
$\mathcal{S}$	System state set
$\mathcal{U}$	User behavior set
$\mathcal{A}$	Cloud clone action set
$\mathcal{P}_{a_t}(s,s')$	System state transition probability from state s to state s' by fulfilling action a_t at time t
$\mathcal{R}_{a_t}(s,s')$	Total monetary cost when state s changes into state s' by fulfilling action a_t at time t
T	Total period that the user is interacting with the multi-screen social TV system
$\pi(s_t)$	Optimal action policy at time slot t
$Q(s,a)$	Q-value of performing action a at state s
α	Learning rate in Q-learning
γ	Discount rate in Q-learning
θ	Q-value convergence threshold

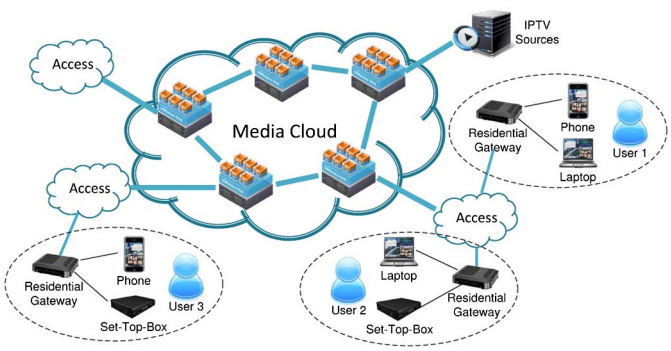


Fig. 1. Multi-screen cloud social TV architecture.

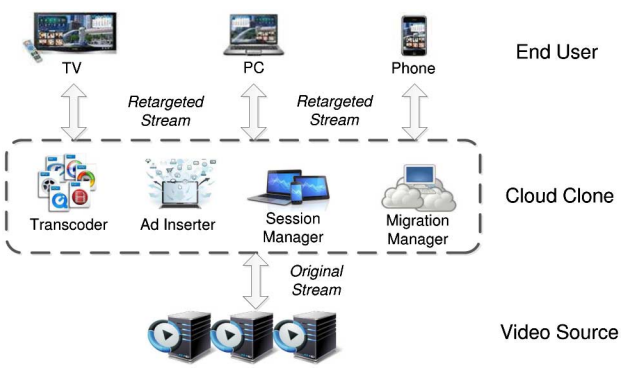


Fig. 2. System overview on cloud clone.

overlay content delivery network. Under this framework, the cloud resources can be dynamically operated in different servers (i.e., cloud nodes) on top of the overlay network, based on a pay-per-use pricing model. Such cloud based CDN paradigm has also been offered by a list of cloud service providers[2] to operate media services in a more efficient manner, than the traditional CDN solution [18].

2) Key Feature: Our system offers a highly-touted multi-screen experience via *video teleportation*. With this feature, one user can simultaneously operate multiple devices (e.g., TV and smartphone) and freely migrate video session back and forward from one device to another without interruption. As a result, the users can always stay connected to TV programs and social interactions.

3) Cloud Clone: The enabling technology for video teleportation is to deploy a *cloud clone* as shown in Fig. 2 for each user. Specifically, every user is represented by a virtual machine (VM), which serves as his proxy in the cloud to manage all the associated devices and real-time session information (e.g., ongoing programs, information about the active device, the updated viewing history, and most recent video segments, etc.). It elastically turns on once its represented user is online, and turns off once the user gets offline. In addition, the VM also dynamically transcodes the original stream from the video source into the one with appropriate format, resolution, and bitrate as its user shifts the devices. Moreover, the VM also offers other functionalities, such as ad insertion to support personalized multi-screen experience. We call such VM as *cloud clone*. And we assume each cloud clone can be operated at any cloud server along any overlay content delivery path.

The location of cloud clone plays an important role on the cost of operating video teleportation. Specifically, the retargeted stream size changes as the user moves around or shift sessions from one device to another. It leads to the changes of the transmission cost, which further depends on the cloud clone location along the delivery path from the source to the user. Thus, there is an opportunity to reduce the operational cost by migrating the cloud clone to its best place.

B. System Models

In this subsection, we present three system models to drive the problem formulation on minimizing the operational cost on support the video teleportation feature.

1) Content Processing Model: In this paper, we assume the cloud clone performs two content processing functions, including advertisement insertion and video transcoding. Through these processing procedures, a content of size B_o will be changed to B_t in the following two cases.

Case 1) $B_o \leq B_t$: When the user is consuming the content on a bigger screen (e.g., TV), B_o would be smaller than the retargeted stream size B_t. In this case, the cloud clone inserts a few personalized video advertisements, by first picking the related advertisement video based on recommendation algorithms, then combining the selected video ad overlay with the targeted quality to the original content [19]. Finally, the combined video streaming is delivered to the end user. Such method has become a key online monetization strategy to make profit [19]. At the same time, some set-top boxes may not support the latest

[2]http://aws.amazon.com/cloudfront/ and http://www.metacdn.com/

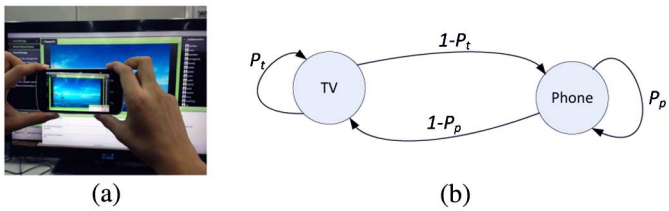


Fig. 3. User behavior on session migration by using video teleportation. (a) Real user case. (b) Device switching model as a Markov chain.

video format (e.g., H.264 high profile) of the content source. Thus, the cloud clone has to transcode the original content into a compatible one (e.g., H.264 baseline profile), and pick the video ad overlay in the same format. As a result, the combined content size could be bigger than the original one. Note, the transmission cost for the cloud clone to load advertising videos is ignored, since we assume the cloud has already pre-loaded all those data at each node.

Case 2) $B_o > B_t$: When the user is consuming the content on a smaller screen (e.g., smartphone), B_o would be larger than B_t. The main reason is that, the video resolution and bitrate required by such a device are significantly lower than those of the original one. In this case, the cloud clone transcodes the original content into an appropriate format with much smaller resolution and lower bitrate, to fit the output. As a result, regardless of the inserted advertisements with small resolution and low bitrate, the retargeted stream size is still much smaller than the original one.

2) User Behavior Model: We model user behavior across different devices as a Markovian process, which has been widely adopted to characterize a variety of user behaviors on web browsing [20], online social activities [21] and IPTV interactions [22]. Without loss of generality, the user is assumed to switch between two devices (i.e., TV and phone), thus the model has two corresponding states. This model can be easily extended if there are more devices or states (e.g., when user is simultaneously using TV and phone, it is a new state). Since the collection of devices at home is limited, the number of states could be within a reasonable size.

Fig. 3(a) present a snapshot to capture a real use case that an user shifts his attention between TV and smartphone by using video teleportation. In particular, from the user's perspective, he can transfer the ongoing programs from TV to smartphone by first scanning the TV screen using smartphone camera, then flipping the phone to trigger the transfer. He can also transfer the sessions back from his smartphone to TV, by simply performing a "throw" gesture. From the system operation aspect, in both cases, the workflow is coordinated by the cloud clone. More specific details can be found from our previous works [3], [4].

Fig. 3(b) illustrates a Markov process with two states to model such user behaviors. Specifically, the state transition matrix is completely determined by p_t (the probability in which the user uses TV in both the current and the next time slot) and p_p (the probability in which the user uses smartphone in both the current and the next time slot). Accordingly, we have $p_{tp} = 1 - p_t$, $p_{pt} = 1 - p_p$, where p_{tp} and p_{pt} are the user device switching probability between TV and smartphone.

3) Cost Model: In supporting the cloud clone migration, the media cloud would incur four cost components, including,

- *Transmission Cost* occurs when videos are transmitted from the source to the user. Note, when operating the media cloud, we only need to consider the cloud network cost, while the cost incurred by access networks (e.g., broadband and wireless network cost) will not be included in this work.
- *Migration Cost* corresponds to the bandwidth cost in which the cloud clone migrates from one node to another within the media cloud.
- *Computing Cost* refers to the consumption of computational resources (e.g., CPU/GPU), when cloud clone processes user requests, transcodes the requested contents into suitable format, and inserts advertisements into the original video. This cost component is a baseline cost, which is invariant of the cloud clone location.
- *Storage Cost* is charged for keeping advertising videos and user sessions. This cost is also a baseline cost, which is constant for different cloud clone locations.

In this paper, we only focus on the transmission and migration cost, and ignore other two baseline costs which are independent of our decision variable (i.e., cloud clone location).

In this research, we consider an operational model in which the media service provider rents cloud resources from a vendor-neutral cloud provider (e.g., AWS, Azure). Among the three price models (i.e., on-demand, reserved and spot instances), we adopt the on-demand model, because media service is often real-time. In this case, the spot instance cannot guarantee QoS and the reserved instance could be wasted.

For a chosen content, we assume a content delivery path from the source to the end user. The path length is L, and there are overall $L + 1$ nodes including the source node, the end user node and $L - 1$ intermediate nodes. Each node along the path is indexed as k according to its hop distance to the source. Note the cloud clone can neither locate at the user side nor the content source. We assume the monetary cost incurred by transmission is proportional to the traversed hop distance, and the content size. When the cloud clone is at k, $k \in \{1, \ldots, L-1\}$, the transmission cost in a time slot is

$$C_{tr}(k) = c_{tr} B_o k + c_{tr} B_t (L - k), \tag{1}$$

where c_{tr} is the per hop price to transmit per GB data, B_o is the expected original video size, and B_t is the expected size, after the cloud clone transcodes the content into a smaller one for small screen, or inserts personalized advertisements to convert it into a bigger one for big screen experience.

The monetary cost for cloud clone migration, includes both VM processing cost to initialize and complete the migration, and the transmission cost to send this VM image, which stores online user sessions (e.g., meta data of ongoing program, active device information, updated viewing and behavior history, most recent streaming segments for continuous watching experience, meta data of personalized ads, etc.) to its new destination. Therefore,

the monetary charged by migrating cloud clone from one node k to another node k' is

$$C_{mig}(k,k') = c_{tr}V_{mig}|k-k'| + c_m, k \neq k', \quad (2)$$

where $k' \in \{1,\ldots,L-1\}$ denotes the hop distance from the cloud clone to media source after migration, V_{mig} is the migrated data volume, and c_m is the additional price to initialize and complete each migration.

C. Markov Decision Process Formulation

Using the system models, we introduce Markov decision process (MDP) to formulate the cloud clone migration problem. Specifically, the MDP formulation is a 4-tuple including the system state set, the scheduling action set, the state transition matrix and the cost function.

System States: We define a system state at time slot t as $s_t = (l_t, u_t)$ by jointly considering the location of cloud clone and active user device. $l_t \in \{1,2,\ldots,L-1\}$ denotes the location of cloud clone at time slot t. $u_t \in \{TV, phone\}$ denotes the active device is TV or smartphone at time slot t. As a result, this system state set $\mathcal{S} = \{s_0,\ldots,s_T\}$ presents both the changes initiated by the user, and the corresponding cloud clone migrations made by the system, where T denotes the period that the user is interacting with our system.

Cloud Clone Migration Actions: The scheduler action set $\mathcal{A} = \{a_0,\ldots,a_{T-1}\}$ defines the destination of the cloud clone migration at each time slot. Specifically, we model the action a_t as the migration decision at time slot t, where $a_t = k$, $k \in \{1,\ldots,L-1\}$, denotes the cloud clone migrates from its current location l_t to a new place $l_{t+1} = k$. We treat $l_t = l_{t+1}$ as the case that no migration is taken at time slot t.

State Transition: The transition from state $s_t = s$ to $s_{t+1} = s'$ is determined by both the user behavior model and the cloud clone migration decision. Since the user decision is independent of the cloud clone migration, we have the transition probability $\mathcal{P}_{a_t}(s,s')$, that action a in state $s = (l,u)$ at time t will lead to state $s' = (l',u')$ at time slot $t+1$ as

$$\begin{aligned}\mathcal{P}_{a_t}(s,s') &= Pr_{(s_{t+1}=(l',u')|s_t=(l,u),l'=a_t)} \\ &= Pr_{(l_{t+1}=l'|l_t=l,l'=a_t)} Pr_{(u_{t+1}=u'|u_t=u)}, \end{aligned} \quad (3)$$

where $Pr_{(u_{t+1}=u'|u_t=u)}$ can be obtained from the transition matrix of user behavior model, and $Pr_{(l_{t+1}=l'|l_t=l,l'=a)}$ is determined by the action policy $\pi(s_t) = a_t$.

Cost Function: We define the cost function $\mathcal{R}_{a_t}(s,s')$ as the total monetary cost consumed during the period from time t to $t+1$. This cost includes both media transmission cost and cloud clone migration cost. As a result, we have

$$\mathcal{R}_{a_t}(s,s') = C_{tr}(k) + (1-\delta(k-k'))C_{mig}(k,k'), \quad (4)$$

where $\delta(x)$ is the indicator function that

$$\delta(x) = \begin{cases} 1, & x = 0 \\ 0, & otherwise. \end{cases} \quad (5)$$

As a result, $1-\delta(k-k')$ indicates the decision on whether the migration should be taken at time t.

Optimization Objective: The goal is to find the optimal migration policy $\pi(s_t) = a_t$ to minimize the total cost. It can be formulated as an unconstrained optimization problem over a finite time horizon

$$\min_{a_t} \sum_{t=0}^{T-1} \mathcal{R}_{a_t}(s_t, s_{t+1}). \quad (6)$$

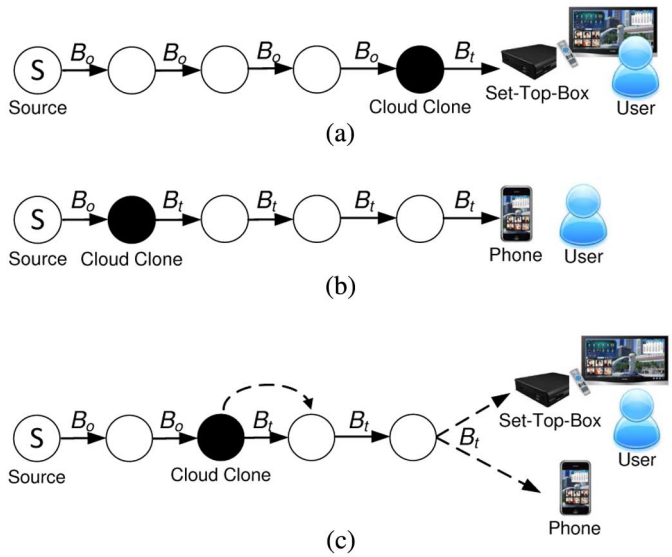


Fig. 4. Case studies on cloud clone migration. (a) Optimal cloud clone placement when $B_o < B_t$. (b) Optimal cloud clone placement when $B_o \geq B_t$. (c) Possible cloud clone scheduling when B_t changes.

IV. Migration Strategies for Cloud Clone

In this section, we start with three simple case studies, to illustrate the problem and the fundamental trade-off between transmission cost and migration cost. Following that, we propose four alternative methods to solve this problem.

A. Case Studies for Cloud Clone Placement

Fig. 4 illustrates three cases for optimal cloud clone placement. In these cases, we consider the content delivery path as a line topology. Node S serves as the media source, and the black node serves the cloud clone. In particular, we consider the advertisement insertion and the video transcoding function at the cloud clone, through which the original content size of B_o is converted to B_t.

1) Case A: In this case, as shown in Fig. 4(a), the end user is always consuming the content on one device with a larger screen (i.e. TV) and the retargeted content size B_t^t is larger than that of the original content (i.e., $B_o < B_t^t$). It can be shown that the optimal location for the cloud clone would be the node nearest to the user along the delivery path, to minimize the transmission cost.

2) Case B: In this case, as shown in Fig. 4(b), the end user is always consuming the content on one device with a smaller screen (i.e., smartphone) and the retargeted size B_t^p is smaller than that of the original content (i.e., $B_o > B_t^p$). It can be shown that the optimal location for the cloud clone would be the node nearest to the content source along the delivery path, to minimize the transmission cost.

3) Case C: In this case, as shown in Fig. 4(c), the user switches between two devices when consuming the same

content. In the two phases, the optimal location for the cloud clone could change in the process to minimize the total cost.

Throughout these three cases, we observe a clear trade-off between the transmission cost and the migration cost. In particular, on the one hand, once the user switches the device, the cloud clone should immediately migrate to its best location (i.e., Case A and B), to minimize the transmission cost. On the other hand, if the switching behavior is frequent, such policy will generate significant migration cost, which may overwhelm the saving on transmission cost. As a result, we need to strategically schedule the cloud clone migration to balance this trade-off.

In this paper, we study the cost minimization problem by examining the fundamental trade-off, according to the aforementioned Markovian user pattern. Four alternative methods will be pursued, as explained in next four subsections.

B. Random Fixed Placement & Cost Upper Bound

In this subsection, we propose a random fixed placement strategy for the cloud clone by assuming it does not migrate during the whole content consumption period T. This is the simplest strategy to this problem, which involves no dynamics. In this case, the total cost C_{tot} in every time slot is completely decided by the content transmission cost as

$$C_{tot}(k) = c_{tr}B_o k + c_{tr}\bar{B}_t(T)(L-k), \tag{7}$$

where $k \in \{1, \ldots, L-1\}$, and $\bar{B}_t(T)$ is the mean content size per time slot after processed by the cloud clone during T

$$\bar{B}_t(T) = (t_p B_t^p + (T-t_p)B_t^t)/T, \tag{8}$$

where t_p is the number of time slot that the user spends on phone during the period T.

Algorithm 1 describes the details of this method. The purpose of proposing such an algorithm is to get a cost upper bound, and set it as a reference to evaluate other algorithms. Because the system can not know the exact user behavior trace in advance, the scheduler can not decide which location is the optimal one at the beginning. Most probably, the system just randomly picks a location in real implementation. As a result, we aim to obtain the expected total cost of the random fixed placement strategy. We first calculate the cost for all $L-1$ locations, then dividing the summation by $L-1$. The results can represent a numerical upper bound for our problem.

Algorithm 1 Random Fixed Placement Algorithm

Input: a complete user behavior trace set $\mathcal{U}$ during T

Output: the expected cost of random fixed placement C_{tot}

1: initialize $C_{tot} = 0$

2: calculate $\bar{B}_t(T)$ by Eq. (8)

3: **for** $k = 1$ **to** $L-1$ **do**

4: $\quad C_{tot} = C_{tot} + B_o k + c_{tr}\bar{B}_t(T)(L-k)$

5: **end for**

6: return $C_{tot} = C_{tot}/(L-1)$ as the expected cost

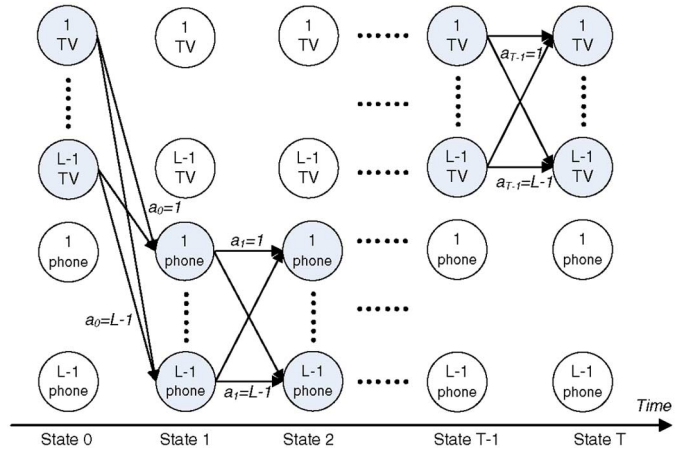


Fig. 5. An example of the state transition of offline algorithm.

We also derive an expected upper bound of the generated monetary cost. According to the Markovian user behavior model, we have the mean state sojourn time that an user spends on TV is

$$\bar{T}_t = \sum_{x=1}^{\infty} x P_t^{x-1} = \frac{1}{1-p_t}. \tag{9}$$

Similarly, we also have the mean state sojourn time the user spends on smartphone is

$$\bar{T}_p = \sum_{x=1}^{\infty} x P_p^{x-1} = \frac{1}{1-p_p}. \tag{10}$$

Thus, we have $\bar{B}_t$ as

$$\bar{B}_t(T) = \frac{\bar{T}_t}{\bar{T}_t + \bar{T}_p} B_t^t + \frac{\bar{T}_p}{\bar{T}_t + \bar{T}_p} B_t^p. \tag{11}$$

As a result, we have the expected total monetary cost C_{fix} of the random fixed placement algorithm as

$$\mathbb{E}[C_{fix}] = \frac{LT}{2} c_{tr}(B_o + \bar{B}_t(T)). \tag{12}$$

The complexity of this random fixed algorithm to take an action is $O(1)$. It just randomly picks one place among all the possible L locations, then keeps it unchanged all the time.

C. Offline Algorithm & Cost Lower Bound

In this subsection, we investigate an offline algorithm, based on dynamic programming approach, to solve the optimal migration problem. Specifically, we assume that the scheduler already has the knowledge of all user behaviors $\mathcal{U} = \{u_0, \ldots, u_T\}$ in advance.

Fig. 5 presents an example of the state transition diagram of this offline algorithm. In this case, all the previous and future user behaviors are deterministic (e.g., in Fig. 5, $u_0 = TV, u_1 = phone, \ldots, u_T = TV$), and all other states are not reachable. Thus, the transition of system states is completely determined by the decision variable a_t at each time slot as

$$\hat{\mathcal{P}}_{a_t}(s, s') = \begin{cases} 1, & u = u_t, u' = u_{t+1}, l' = a_t; \\ 0, & otherwise. \end{cases} \tag{13}$$

We denote $v(s_t)$ as the minimal aggregated cost from s_t to s_T. Using value iteration in the backward induction, we have

$$v(s_t) = \min_{a_t} \left\{ \sum_{s_{t+1}} \hat{\mathcal{P}}_{a_t}(s_t, s_{t+1})(\mathcal{R}_{a_t}(s_t, s_{t+1}) + v(s_{t+1})) \right\},$$

where $v(s_T) = 0$ gives an initial value. By using this iterative equation, we can find the optimal policy, given by

$$\pi(s_t) = \arg\min_{a_t} \left\{ \sum_{s_{t+1}} \hat{\mathcal{P}}_{a_t}(s_t, s_{t+1})(\mathcal{R}_{a_t}(s_t, s_{t+1}) + v(s_{t+1})) \right\}.$$

Algorithm 2 presents the details of this method. Specifically, it adopts the dynamic programming approach to iteratively calculate the minimal aggregated cost and the corresponding decision at each time slot. Thus, the complexity of the offline algorithm to take an action is $O(L)$.

Algorithm 2 Offline Algorithm

Input: a complete user behavior trace set $\mathcal{U}$ during T

Output: optimal policy $\pi(s_t)$ at each time slot

1:initialize $v(s_T) = 0$

2:calculate $\hat{\mathcal{P}}_{a_t}(s, s')$ from $\mathcal{U}$ by Eq. (13)

3:**for** $t = T - 1$ **to** 1 **do**

4: **for** $a_t = 1$ **to** $L - 1$ **do**

5: $v'(s_t) = \sum_{s_{t+1}} \hat{\mathcal{P}}_{a_t}(s_t, s_{t+1})(\mathcal{R}_{a_t}(s_t, s_{t+1}) + v(s_{t+1}))$

6: **end for**

7: update $v(s_t) = \min_{a_t} v'(s_t)$, $\pi(s_t) = \arg\min_{a_t} v'(s_t)$

8: **end for**

9: return $\Pi = \{\pi(s_1), \ldots, \pi(s_{T-1})\}$ as the optimal policy

Since the scheduler has perfect information from the user side, it follows that the monetary cost resulted from this offline algorithm provides a lower bound for the original cloud clone migration problem. We will verify this result with numerical simulations in Section V.

D. Semi-Online Algorithm with Known Transition Model

This subsection proposes an semi-online algorithm with known transition model. In this case, the scheduler only knows the user behavior transition matrix instead of the exact traces in advance, and can observe the user pattern as time evolves.

Fig. 6 presents a system illustration of the state transition diagram of this non-learning semi-online algorithm. Specifically, it becomes a stochastic environment that, by performing action a_t at time t, the new system state can be either $s' = (a_t, TV)$ or $s' = (a_t, phone)$. This transition probability is defined by the user behavior model as

$$\mathcal{P}_{a_t}(s, s') = \begin{cases} p_t, & u = TV, u' = TV, l' = a_t; \\ p_p, & u = phone, u' = phone, l' = a_t; \\ 1 - p_t, & u = TV, u' = phone, l' = a_t; \\ 1 - p_p, & u = phone, u' = TV, l' = a_t; \end{cases}$$

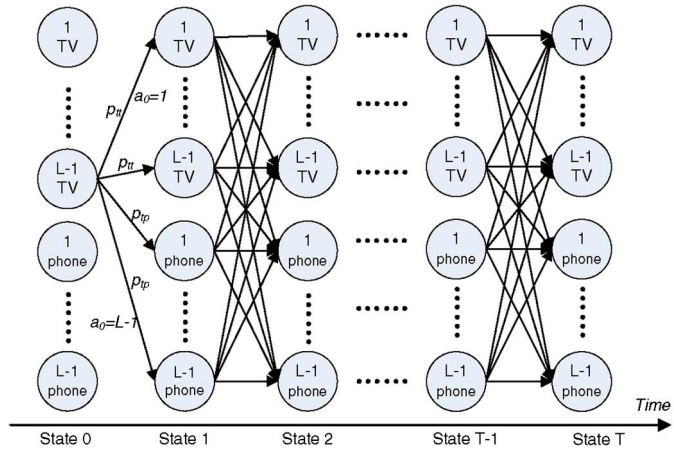


Fig. 6. Systematic illustration of the state transition of semi-online algorithm with known transition probability.

Similar to the offline method, we define $v(s_t)$ as the minimal expected aggregated cost from s_t to s_T as

$$v(s_t) = \min_{a_t} \left\{ \sum_{s_{t+1}} \mathcal{P}_{a_t}(s_t, s_{t+1})(\mathcal{R}_{a_t}(s_t, s_{t+1}) + v(s_{t+1})) \right\}.$$

By still using a backward Bellman equation, we can derive the optimal policy as

$$\pi(s_t) = \arg\min_{a_t} \left\{ \sum_{s_{t+1}} \mathcal{P}_{a_t}(s_t, s_{t+1})(\mathcal{R}_{a_t}(s_t, s_{t+1}) + v(s_{t+1})) \right\}.$$

Algorithm 3 presents the implementation details. Comparing to the offline algorithm, there are only two differences. First, the input changes from a complete user behavior trace set to the transition probability. Second, we replace the deterministic transition model $\hat{\mathcal{P}}$ with the stochastic state transit model $\mathcal{P}$. Therefore, the complexity of the semi-online algorithm to take an action is $O(L)$, which is the same as the offline algorithm.

Algorithm 3 Semi-Online Algorithm

Input: transition probability $\mathcal{P}_{a_t}(s, s')$ of user states

Output: optimal policy $\pi(s_t)$ at each time slot

1:initialize $v(s_T) = 0$

2:**for** $t = T - 1$ **to** 1 **do**

3: **for** $a_t = 1$ **to** $L - 1$ **do**

4: $v'(s_t) = \sum_{s_{t+1}} \mathcal{P}_{a_t}(s_t, s_{t+1})(\mathcal{R}_{a_t}(s_t, s_{t+1}) + v(s_{t+1}))$

5: **end for**

6: update $v(s_t) = \min_{a_t} v'(s_t)$, $\pi(s_t) = \arg\min_{a_t} v'(s_t)$

7:**end for**

8:return $\Pi = \{\pi(s_1), \ldots, \pi(s_{T-1})\}$ as the optimal policy

It can be shown that the minimum cost resulted from this semi-online algorithm falls between the lower bound and upper

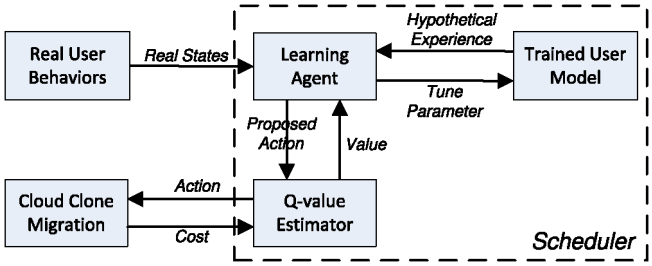


Fig. 7. Systematic model of online algorithm with reinforcement learning.

bound derived in previous two subsection, as verified by performance evaluation in Section V.

E. Online Algorithm With Reinforcement Learning

This subsection proposes an online algorithm with reinforcement learning. In this case, the scheduler knows neither the exact user behavior traces nor the states transition matrix. Instead, we set an agent to continuously learn the user behaviors and tune the transition parameters from observing the outcomes of its previous actions and past experiences.

Fig. 7 shows the systematic model on how this interacts with the real world. Specifically, there are two inputs from the real world, including the real user behaviors and the costs incurred by applying cloud clone migration. In each discrete time slot, the learning agent receives the real states from the user side. It then tunes the trained user model and decides the next action based on current and past experiences by estimating the cost of each possible action and selecting the one which generates the minimal cost.

In real implementation, this approach consists of an experience replay phase and an online scheduling/learning phase.

1) Experience Replay: The objective of this experience replay phase is to obtain the optimal policy for each state by implicitly training the user behavior model based on history data. In particular, we set a warm-up stage for each new user. During this stage, we monitor this user's behaviors, but apply a fixed placement algorithm to avoid the expensive cost incurred by the state exploration. By replaying those experiences, we adopt the Temporal Difference (TD) Q-learning [23] to calculate the Q-value and update the knowledge of the internal trained models. Specifically, the Q-value represents the cost of each state-action pair combination, denoted as $Q(s,a)$. We first initialize every Q-value with an arbitrary value. Then an action a is simulated to each observed state s, and a corresponding cost $\mathcal{R}_a(s,s')$ can be obtained. Thus, the Q-value can be iteratively updated by

$$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha(\gamma \min_a Q(s',a) - \mathcal{R}_a(s,s')), \tag{14}$$

where $\alpha(0 < \alpha \leq 1)$ is the learning rate to determine how much the newly acquired information will override the old information, $\gamma(0 \leq \gamma \leq 1)$ is the discount factor to trade off the importance of sooner and later cost.

We set T_w as the initial value of the period for the warm-up phase. The warm-up stage will terminate if it converges within this specified initial duration. Otherwise, we extend this phase until all Q-value pairs get converged. For the latter case, it has been shown that, any finite MDP problems, Q-learning will eventually converge for sure [23].

We also use an ϵ-greedy policy to explore every reachable state as much as possible, to ensure the optimality of the generated solution. Specifically, at each iteration of Q-learning phase, instead of always picking the greedily action with the learned optimal Q-value, we also select a random action with a small probability $\epsilon(0 \leq \epsilon < 1)$ as

$$\pi(s_t) = \begin{cases} \arg\min_a Q(s,a), & w.p.(1-\epsilon) \\ random\ action\ from \mathcal{A}, & w.p.\epsilon \end{cases} \tag{15}$$

where $w.p.$ means 'with probability', and ϵ will reduce from its initial value by a reduction rate ϵ' per time slot, until eventually $\epsilon = 0$ at the end of the exploration period.

Algorithm 4 presents the details of this experience replay phase. It implicitly studies the user behaviors by iteratively updating the Q-value. A classic ϵ-greedy exploration function is adopted to select actions at each time slot. The gathered user behavior trace will be replayed for $L-1$ times, because there are $L-1$ possible actions at each state. After the reply phase, we check whether all the Q-values are converged. If so, this phase ends. Otherwise, we need to collect more sample data. In our problem, the number of states (i.e., $2L-2$) and actions (i.e., $L-1$) is limited. Therefore, the convergence rate of Q-value should be within an acceptable range. If the problem space becomes larger with more states and actions, we can combine a function approximator [24] into our algorithm.

Algorithm 4 Reinforcement Learning with Experience Replay

Input: user behavior history $\mathcal{U} = \{u_1, \ldots, u_{T_w}\}$

Q-value convergence threshold θ

Output: $Q(s,a)$ for each state-action pair

1:initialize all $Q(s,a)$ arbitrarily (e.g., identically 10)

2:**for** $i = 1$ **to** $L-1$ **do**

3: **for** $t = 1$ **to** $T_w - 1$ **do**

4: simulate $a_t = \pi(s_t)$ according to Eq. (15)

5: observe the new state s_{t+1}, and the cost $\mathcal{R}_a(s_t, s_{t+1})$

6: update $Q(s_t, a_t)$ by Eq. (14)

7: update $\epsilon = \epsilon - \epsilon'$

8: **end for**

9:**end for**

10:**if** all $|Q'(s,a) - Q(s,a)| < \theta$ **then** return $Q(s,a)$

11:**else** continue to collect user behavior data

12:**end if**

2) Online Scheduling and Learning: This phase uses the trained $Q(s,a)$ generated from the experience replay phase to make optimal decisions. Algorithm 5 presents the details of

this algorithm. Specifically, at each time slot, the scheduler will choose the action with the minimal Q-value from all state-action pairs, as the optimal policy based on current system state. Therefore, the complexity of the Q-learning based algorithm is $O(L)$. Besides, each updated Q-value will continue to make contribution to learn the latest user behavior pattern. As a result, it forms a lifetime learning process.

Algorithm 5 Online Algorithm with Q-Learning

Input: current system state s_t at time slot t all learned $Q(s,a)$ from Experience Replay phase

Output: optimal policy $\pi(s_t)$ at each time slot

1:**for** $t = 1$ **to** $T-1$ **do**

2: apply $\pi(s_t) = \arg\min_a Q(s,a)$

3: observe the new state s_{t+1}, and the cost $\mathcal{R}_a(s_t, s_{t+1})$

4: update $Q(s_t, a_t)$ by Eq. (14)

5:**end for**

6:return $\Pi = \{\pi(s_1), \ldots, \pi(s_{T-1})\}$ as the optimal policy

This online algorithm with reinforcement learning is the most practical one comparing with others. First, in real system, only the history of user behavior is accessible while either the exact user traces in future or just the precise behavior model for each individual user is difficulty or even impossible to obtain in advance. More importantly, this method adopts a model-free reinforcement learning technique. It indicates that even the transition probability may not be necessarily deterministic. This approach can still work when the transition probability is a random variable that its probability follows any distribution model. Second, the complexity of the algorithms keeps at a very low level for each cloud clone to take the optimal action, because the number of possible locations along the shortest content delivery path is usually small (e.g., less than 20). This makes it capable to serve thousands of users simultaneously. We will verify its performance in comparison with the ones of other approaches in Section V.

V. Performance Evaluation

This section verifies the performance of all those algorithms based on real application scenarios.

A. Experimental Settings

1) Experimental Approaches: In this work, both simulated user behaviors and real traces are used to drive the evaluations.

For simulated user data, we generate 1000 periods of user behaviors according to the Markov model in Section III-B. Each period contains the user behaviors for $T = 2hrs$. We set each time slot as $t = 10$ seconds, since the video teleportation operation could take around 10 seconds [3]. In this way, there are 720 records in each period, where each record shows the active device at one time slot. And there are in total 720,000 records for one simulated trace. All the algorithms run for 1000 rounds based on each trace. When we run the Q-learning algorithm, only the average cost from the online scheduling period will be reported. When we run other algorithms, we report the average cost based on all those 1000 periods.

TABLE II
Experimental Parameter Settings

Parameter Definition	Settings
Length of one time slot	$t = 10$ seconds
Per-hop price to transmit per GB data	$c_{tr} = 0.12$ USD/GB
Additional price to complete migration	$c_m = 0.02$ USD
Expected original content size	$B_o = 1.875$ MB/slot
Expected content size to smartphone	$B_t^p = 0.5$ MB/slot
Expected content size to TV	$B_t^t = 2.5$ MB/slot
Q-Learning learning rate	$\alpha = 0.5$
Q-learning discount factor	$\gamma = 0.99$
Q-value convergence threshold	$\theta = 0.001$

For real traces, we collected the user behaviors by releasing this multi-screen social TV system to over 200 students at Nanyang Technological University for an internal trial. The users can enjoy the services on a bigger screen via a web-based application portal, and on their smartphone via a iOS/Android app provided by us. There are around 20 active users among all registered users. The total interaction time from each of them is over 50 hours. We collect behavior traces from all those active users over the last one month (i.e., 08-Feb-2014 to 07-Mar-2014), and use them as the input of our algorithm. Similarly, we only report the average costs of serving these users.

2) Parameter Settings: We adopt the price information from Amazon [25]. Specifically, the per-hop price to transmit one Gigabyte data is $c_{tr} = 0.12$ USD/GB, the price to migrate cloud clone image is $c_m = 0.02$ USD, if we use the extra small VM instance to implement cloud clone.

We use Cisco's Media Experience Engine [26] as a reference to obtain the bitrate information. Specifically, we define $b_o = 1500$ kbps as the bitrate of live streaming sources with 640x480 resolution delivered by Veoh Network (a popular Internet television provider). We set $b_t^t = 2000$ kbps as the bitrate, after the cloud clone inserts video advertisements into the original video, and transcodes it into AVC (Advanced Video Coding) SD (Standard Definition) output with 640x480 resolution for TV viewing. We set $b_t^p = 400$ kbps as the bitrate when the source is transcoded into iPhone compliant MPEG-4 output with 480×320 resolution for iPhone player. Because a time slot lasts for 10 seconds, we have $B_o = 1.875$ MB, $B_t^t = 2.5$ MB, and $B_t^p = 0.5$ MB in each time slot.

For Q-learning settings, we set the learning rate at $\alpha = 0.5$ to give equal weight to new and old knowledge, the discount factor at $\gamma = 0.99$ to take more future costs into account, and the Q-value coverage threshold at $\theta = 0.001$. We initially set $\epsilon = 0.5$, and the ϵ reduction rate is $\epsilon' = 0.0005$ per time slot. We set the initial value of the warm-up period as the first 4000 records for the simulated user behavior input, and the first 10 hours (i.e., 3600 time slots) for real traces of each user. By tracking the convergence performance of Q-learning algorithm over all evaluation sets, we find the average convergence period is around 5080 transitions. Table II summarizes these experimental settings in detail.

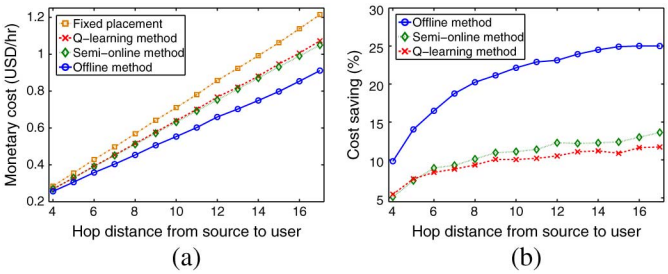


Fig. 8. Minimal monetary cost vs. delivery path (simulated user behaviors). (a) Minimal cost per hou.r (b) Cost saving.

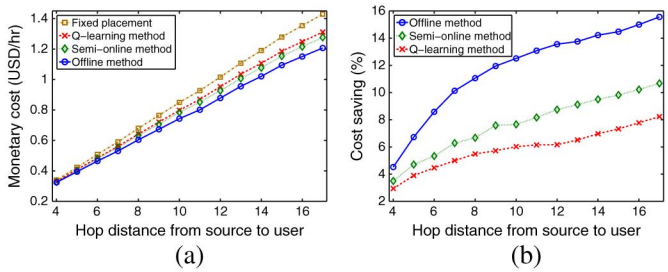


Fig. 9. Minimal monetary cost vs. delivery path (real traces). (a) Minimal cost per hour. (b) Cost saving.

B. Monetary Cost

This subsection evaluates the monetary cost resulted from different algorithms with various parameters. Our focus is to understand the impact of those parameters, ultimately obtaining operational guidelines for the system implementation.

1) Monetary Cost vs. Delivery Path Length: We then investigate the monetary cost as a function of the content delivery path length L. Fig. 8 shows the monetary cost and the cost savings compared to the random fixed placement, based on simulated data, where $p_t = p_p = 0.99$. And Fig. 9 presents the same metrics based on real user traces. In this experiment, we set the VM migration size as $V_{mig} = 8$ MB.

We have a few observations from these two figures. First, the two figures present almost the same trend. It implicitly verifies that our Markov chain based user behavior model can represent the real user behaviors. This can also observed in Fig. 10 and Fig. 11. Second, the monetary cost resulted from the semi-online method and the Q-learning method always fall between the ones resulted from the random fixed placement policy and the offline algorithm. It fits well with our analysis in Section IV. This observation can be also applied to Fig. 10, Fig. 11 and Fig. 12. Third, the monetary cost resulted from all the four algorithms increases almost linearly as the delivery path length L increases. This can be traced by the linear cost model as in Eq. (1) and (2). Finally, compared to the random fixed placement method, our algorithm provides significant cost saving, which increases as the length of the content delivery path increases. For example, the cost saving of Q-learning algorithm is less than 5% when $L = 4$. And it reaches more than 12% when $L = 17$.

2) Monetary Cost vs. Migration Size: We first investigate the relationship between the overall monetary cost and the VM migration size V_{mig}. Fig. 10 shows the monetary cost and cost savings compared to the random fixed placement, based on simulated data, where $p_t = p_p = 0.99$. And Fig. 11 illustrates the

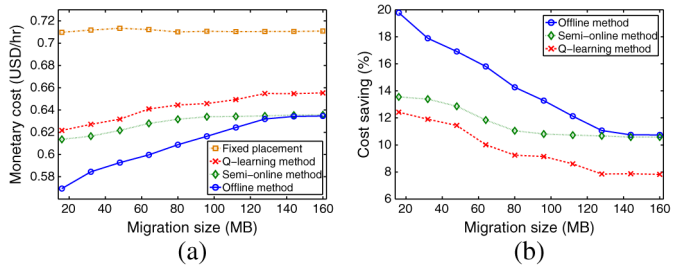


Fig. 10. Minimal monetary cost vs. migration size (simulated user behaviors). (a) Minimal cost per hour. (b) Cost saving.

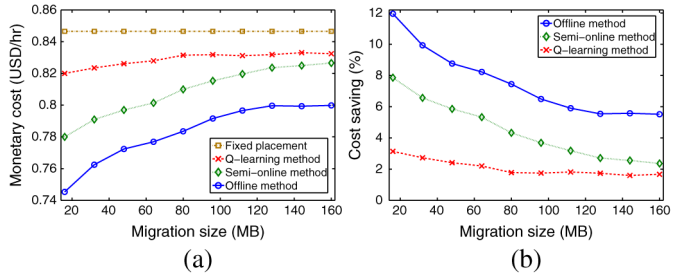


Fig. 11. Minimal monetary cost vs. migration size (real traces). (a) Minimal cost per hour. (b) Cost saving.

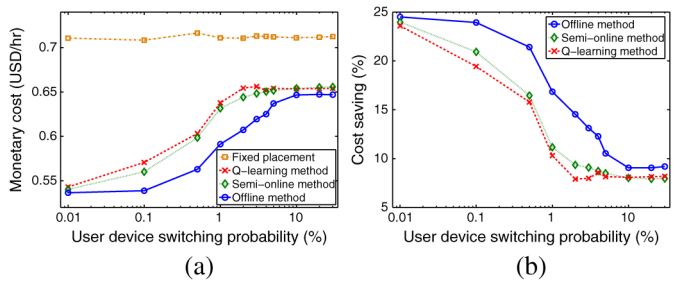


Fig. 12. Minimal monetary cost vs. simulated user behavior patterns. (a) Minimal cost per hour. (b) Cost saving.

same metrics based on real user traces. Here, we set the content delivery path length $L = 10$.

This experiment reveals a few insights. First, the cost resulted from the random fixed method remains the same when V_{mig} changes, because there is no migration based on this policy. This observation can be also applied to Fig. 12 for the same reason. Second, a threshold effect is observed, where the monetary cost remains constant when the migration size is beyond the threshold (e.g., $V_{mig} \approx 120$ MB in the simulated based experiment). On one hand, when $V_{mig} < 120$ MB, as V_{mig} increases, the migration cost grows. On the other hand, when $V_{mig} > 120$ MB, the monetary cost remains almost constant, and the cost savings of the semi-online method and the Q-learning method are getting closed to the one of the offline approach (i.e., the cost lower bound). This effect can be understood as follows. When the migration size is beyond a threshold, the migration cost would be higher than the minimum transmission cost when the cloud clone locates in the best location. As a result, it would be better if the cloud clone stays at its current location without any migration. And the optimal migration policy will eventually reduce to the case of the optimal fixed placement, as the migration size continues to increase.

3) Monetary Cost vs. User Behavior: We study the impact of user behavior pattern on the monetary cost by changing the user device switching probability. Fig. 12 presents the monetary cost and the cost savings compared to the random fixed placement method, as a function of user device switching probability $p_s = p_{tp} = p_{pt}$. In this set of experiment, we set the content delivery path length as $L = 10$, the migration size as $V_{mig} = 8$ MB. And we assume the probability of transiting from phone to phone, is the same with the one of transiting from TV to TV (i.e., $p_p = p_t$).

This experiment also reveals a threshold effect, in which the system behaves differently on the two sides of the threshold (e.g., $p_s \approx 2\%$ in this case). On one hand, when $p_s > 2\%$, the monetary cost and the according cost savings remain almost the same as the user device switching probability changes. Because in this case, the user switches his device too often. Thus, the high migration cost would prevent the cloud clone moving such often. In this way, similar to the case of large migration size in Section V-B2, the migration policy reduces to the optimal fixed placement policy. On the other hand, when $p_s < 2\%$, the cost of all migration methods grow, and their cost savings decrease accordingly, as the device switching probability p_s increases. This can be understood as follows. When the device switching probability is low, the state sojourn time on each device is long as shown in Eq. (9) and (10). As a result, the cloud clone can migrate as soon as the user switches his device. Because the transmission cost saving would easily overwhelm the incurred migration cost. As p_s increases, the state sojourn time decreases. In this way, the device switching frequency becomes higher, and the cloud clone would migrate less frequently than that for device switching. Thus, the chance in which the cloud clone is not placed in the optimal location increases, resulting in a higher transmission cost.

We also notice that, when the transition probability is extremely low (e.g., $p_s < 0.1\%$ in this case), the cost and its saving of the offline method remains constant. This flat curve can be understood by examining a saving function $\Delta(k, k')$ by migrating cloud clone from location k to k', defined as,

$$\Delta(k, k') = t_m(C_{tr}(k') - C_{tr}(k)) - C_{mig}(k, k'), \qquad (16)$$

where t_m is the time period from now to the next migration (or the end of period T if there is no further migration). By substituting Eq. (1) and (2) into Eq. (16), we obtain

$$\Delta(k,k') = \begin{cases} c_{tr}(k'-k)(t_m(\bar{B}_t(t_m)-B_o)-V_{mig})-c_m, & k' \geq k \\ c_{tr}(k'-k)(t_m(\bar{B}_t(t_m)-B_o)+V_{mig})-c_m, & k' < k \end{cases}$$

where $\bar{B}_t(t_m)$ is the mean retargeted content size during t_m

$$\bar{B}_t(t_m) = (t_p B_t^p + (t_m - t_p)B_t^t)/t_m, \qquad (17)$$

and t_p is the time the user spends on phone during t_m. We can see that, in this case, the cost saving is determined by $(\bar{B}_t(t_m) - B_o)$, which is in turn determined by t_p. Since t_p can not vary in a large scale when $p_s < 0.1\%$, the monetary cost and the cost saving are almost the same, when p_s changes.

Finally, we find the cost saving of Q-learning based algorithm can reach nearly 25% when $p_s < 0.01\%$, where its performance gets closed to the ones of offline and semi-online algorithm. In this case, if the service would be offered to 100 users for one year, the monetary cost saving can be as much as $\$250,000$ per year. The reason is that, when the switching probability is small, the time spent on one device is more likely to dominate the other. In addition, the sojourn time on each device becomes longer. Therefore, such pattern is easier to be learnt, and future user behaviors following this pattern is easier to be predicted.

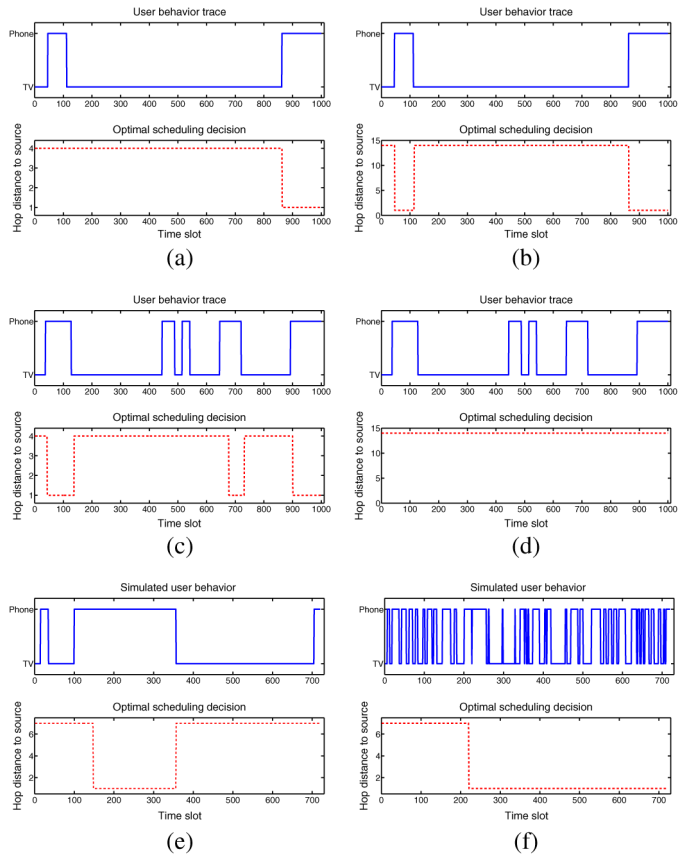


Fig. 13. Examples of the migration scheduling based on Q-learning method. (a) User A ($V_{mig} = 16$ MB, $L = 5$). (b) User A ($V_{mig} = 16$ MB, $L = 15$). (c) User B ($V_{mig} = 16$ MB, $L = 8$). (d) User B ($V_{mig} = 144$ MB, $L = 8$). (e) Simulated ($p_p = p_t = 0.99$, $V_{mig} = 16$ MB, $L = 8$). (f) Simulated $p_p = p_t = 0.9$, $V_{mig} = 32$ MB, $L = 8$.

C. Optimal Migration Polices

In this subsection, we investigate the optimal migration scheduling generated by Q-learning algorithm through a few examples as shown in Fig. 13.

First, Fig. 13(a) and Fig. 13(b), investigate the impact of content delivery path length, based on the real trace over latest 1000 time slots from User A, who switches his device the least frequently among all active users. We find that, when the delivery path length is short (e.g., Fig. 13(a)), the cloud clone only migrates for once. While in Fig. 13(b), the cloud clone migrates for three times. The migration is easier to be triggered when the delivery path is longer, because the cost savings generated by migration scheduling are much higher in this case. It verifies our discussions in Section V-B1.

Second, Fig. 13(c) and Fig. 13(d) focus on the impact of VM migration size, based on real traces over latest 1000 time slots from User B, who switches his device most frequently among

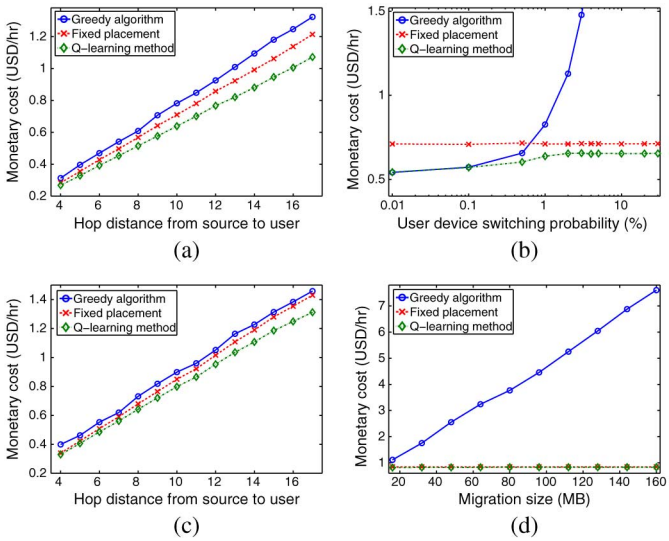


Fig. 14. Comparisons between Q-learning algorithm and existing methods. (a) Cost vs. path length (simulated). (b) Cost vs. switch prob. (simulated). (c) Cost vs. path length (real trace). (d) Cost vs. migration size (real trace).

all active users. We find that, when migration size is small (e.g., Fig. 13(c)), the cloud clone migrates more frequently, because low migration cost makes the transmission cost savings much easier to cover the incurred migration cost. As the migration size increases, the migration frequency decreases. When $V_{mig} = 144$ MB (i.e., Fig. 13(d)), there is no migration at all. This verifies our discussions in Section V-B2.

Third, Fig. 13(e) and Fig. 13(f) present the impact of user device switching probability based on simulated data. We find if the user switches the active device too frequently (i.e., 13(f)), there will be much less migrations compared to Fig. 13(e). This verifies our discussions in Section V-B3.

Moreover, we notice an interesting observation in all examples, that the optimal cloud clone location is either at the nearest or the furthest node to user, regardless of delivery path length, migration size and user behaviors. This can be understood from the cost saving function $\Delta(k, k')$ as Eq. (17). Specifically, the scheduler checks whether $\max \Delta(k, k') > 0$ to trigger a migration. If a migration is required, the place which leads to the maximal cost saving, is selected as the destination. It can be seen that, $\Delta(k, k')$ is a linear function of $(k' - k)$. Therefore, if $\Delta(k, k') > 0$, it is better to migrate the cloud clone to the furthest node to maximize $(k' - k)$, thus the cost saving can be maximized. Otherwise, the cloud clone would stay at its current location with no migration.

D. Comparisons With Existing Methods

For comparison purpose, we regard the random fixed algorithm as the standard method adopted by existing social TV systems [27], [10]. In addition, we also implement another greedy algorithm inspired by [28], where the cloud clone always migrates to its best place, once the user switches the active device.

Fig. 14 compares the performance of our Q-learning based algorithm with those two existing algorithms. First, we find the proposed Q-learning method outperforms the existing ones in all the cases. This verifies the effectiveness of our approach. Second, the greedy algorithm generates extremely high costs when the switching probability is high, or the migration size is large. In this case, the migration cost becomes very high, resulting in tremendous total operational cost. Finally, when the switching probability is very low, the greedy algorithm performs very closed to our scheme, which is better than the fixed one. This implies that when user seldom switches the active device, the greedy scheduling stands for the optimal solution. Otherwise, we need to migrate the cloud clone very carefully to reduce the cost.

VI. Conclusion and Future Works

This paper investigated the problem on minimizing monetary cost via cloud clone migration in multi-screen cloud social TV system. We formulated it as a Markov Decision Process, to balance a trade-off between the transmission and migration cost. Under this framework, we first considered a random fixed placement and an offline algorithm to obtain an upper and lower bound for the optimal cost. We then proposed a semi-online algorithm and a more practical Q-learning method. We use both simulated data and real user traces to evaluate all the four algorithms. The results indicated, up to 25% monetary cost compared with the random fixed placement can be saved in typical use scenarios, by optimally migrating the cloud clone. The cost savings can be affected by the delivery path length, the VM migration size and the user behavior pattern. Moreover, we also found the optimal cloud clone location is either at the nearest or the furthest node to the user. Those insights would offer operational guidelines to deliver cost effective multi-screen social TV services over CCMN, potentially easing its adoption.

Our multi-screen social TV system has been implemented on top of a private cloud at Nanyang Technological University. It has been exposed to over 200 students for an internal trial. The next step is to deploy it to a vendor-neutral cloud provider (e.g., Amazon EC2) to achieve the large-scale deployment. Second, we plan to explore the possibility of implementing other cloud application frameworks (e.g., one cloud clone for multiple users). Finally, we will try to solve more complicated problems (e.g., MDP with constraints) in related scenarios.

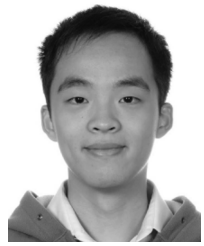
Acknowledgment

The authors would like to thank Prof. W. Zhu from Tsinghua University, Prof. C. W. Chen from the University at Buffalo, Prof. D. O. Wu from the University of Florida, Prof. H.-H. Chen from National Cheng Kung University, and Prof. X. Shen from the University of Waterloo for their technical discussions.

References

[1] M. Montpetit and M. Médard, "Social television: Enabling technologies and architectures," in *Proc. IEEE*, 2012, vol. 100, pp. 1395–1399.
[2] Z. Yu, X. Zhou, L. Zhou, and K. Du, "A hybrid similarity measure of contents for TV personalization," *Multimedia Syst.*, vol. 16, pp. 231–241, 2010.
[3] Y. Jin, X. Liu, Y. Wen, and J. Cai, "Inter-screen interaction for session recognition and transfer based on cloud centric media network," in *Proc. 2013 IEEE Int. Symp. Circuits Syst.*, 2013, pp. 877–880.
[4] Y. Jin, T. Xie, Y. Wen, and H. Xie, "Multi-screen cloud social TV: Transforming TV experience into 21st century," in *Proc. ACM MM*, 2013, pp. 435–436.
[5] Y. Jin, Y. Wen, G. Shi, G. Wang, and A. Vasilakos, "CoDaaS: An experimental cloud-centric content delivery platform for user-generated contents," in *Proc. IEEE 2012 Int. Conf. Comput., Netw., Commun.*, 2012, pp. 934–938.

[6] Y. Zhang, C. Yan, F. Dai, and Y. Ma, "Efficient parallel framework for h. 264/avc deblocking filter on many-core platform," *IEEE Trans. Multimedia*, vol. 14, no. 3, pp. 510–524, Jun. 2012.
[7] T. Chang and Y. Li, "Deep shot: A framework for migrating tasks across devices using mobile phone cameras," in *Proc. ACM CHI*, 2011, pp. 2163–2172.
[8] Y. Lu, S. Li, and H. Shen, "Virtualized screen: A third element for cloud mobile convergence," *IEEE Multimedia* , vol. 18, no. 2, pp. 4–11, Feb. 2011.
[9] Y. Wu, Z. Zhang, C. Wu, Z. Li, and F. C. Lau, "Cloudmov: Cloud-based mobile social TV," *IEEE Trans. Multimedia*, vol. 15, no. 4, pp. 821–832, Jun. 2013.
[10] X. Wang, M. Chen, T. Kwon, L. Yang, and V. Leung, "Ames-cloud: A framework of adaptive mobile video streaming and efficient social video sharing in the clouds," *IEEE Trans. Multimedia*, vol. 15, no. 4, pp. 811–820, Jun. 2013.
[11] M. Bernaschi, F. Cacace, R. Clementelli, and L. Vollero, "Adaptive streaming on heterogeneous networks," in *Proc. ACM Workshop Wireless Multimedia Netw. Perform. Model.*, 2005, pp. 16–23.
[12] B. Shen, S.-J. Lee, and S. Basu, "Caching strategies in transcoding-enabled proxy systems for streaming media distribution networks," *IEEE Trans. Multimedia*, vol. 6, no. 2, pp. 375–386, Apr. 2004.
[13] J. Chakareski and P. Frossard, "Adaptive systems for improved media streaming experience," *IEEE Commun. Mag.*, vol. 45, no. 1, pp. 77–83, Jan. 2007.
[14] M. Armbrust and A. Fox *et al.*, "A view of cloud computing," *Commun. ACM*, vol. 53, no. 4, pp. 50–58, 2010.
[15] J. Llorca, K. Guan, G. Atkinson, and D. Kilper, "Energy efficient delivery of immersive video centric services," in *Proc. IEEE INFOCOM*, 2012, pp. 1656–1664.
[16] Y. Jin, Y. Wen, K. Guan, D. Kilper, and H. Xie, "Towards monetary cost effective content placement in cloud centric media network," in *Proc. IEEE ICME*, 2013, pp. 1–6.
[17] I. Zhovnirofsky and M. Montpetit, "Internet discovery of enhancement and acceleration services (IDEAS)," in *Proc. Internet-Drafts*, 2013.
[18] F. Chen, K. Guo, J. Lin, and T. La Porta, "Intra-cloud lightning: Building cdns in the cloud," in *Proc. IEEE INFOCOM*, 2012, pp. 433–441.
[19] T. Mei, J. Guo, X.-S. Hua, and F. Liu, "AdOn: Toward contextual overlay in-video advertising," *Multimedia Syst.*, vol. 16, no. 4–5, pp. 335–344, Aug. 2010.
[20] Ş. Gündüz and M. T. Özsu, "A web page prediction model based on click-stream tree representation of user behavior," in *Proc. ACM SIGKDD*, 2003, pp. 535–540.
[21] V. S. Tseng and K. W. Lin, "Efficient mining and prediction of user behavior patterns in mobile web systems," *Inf. Softw. Technol.*, vol. 48, no. 6, pp. 357–369, 2006.
[22] V. Gopalakrishnan and R. Jana *et al.*, "Understanding couch potatoes: Measurement and modeling of interactive usage of IPTV at large scale," in *Proc. ACM IMC*, 2011, pp. 225–242.
[23] C. J. Watkins and P. Dayan, "Q-learning," *Mach. Learn.*, vol. 8, no. 3-4, pp. 279–292, 1992.
[24] G. Tesauro, "Temporal difference learning and td-gammon," *Commun. ACM*, vol. 38, no. 3, pp. 58–68, 1995.
[25] "Amazon pricing," [Online]. Available: http://aws.amazon.com/pricing/
[26] *"User Guide for Cisco Media Experience Engine 3000,"* Cisco Systems, Inc., San Jose, CA, USA, Oct. 10, 2008 [Online]. Available: http://www.cisco.com/en/US/docs/video/mxe/2_x/2.0/user/guide/mxe_20_ug.pdf
[27] Y. Wu, Z. Zhang, C. Wu, Z. Li, and F. Lau, "CloudMoV: Cloud-based mobile social TV," *IEEE Trans. Multimedia*, vol. 15, no. 4, pp. 821–832, Jun. 2013.
[28] C. Peng, M. Kim, Z. Zhang, and H. Lei, "VDN: Virtual machine image distribution network for cloud data centers," in *Proc. IEEE INFOCOM*, 2012, pp. 181–189.



Yichao Jin received the B.E and M.E degrees from Nanjing University of Posts and Telecommunications (NUPT), Nanjing, China, in 2008 and 2011, respectively. He is currently pursuing the Ph.D degree from the School of Computer Engineering, Nanyang Technological University (NTU), Singapore. His research interests are cloud computing and content delivery networks.

Yonggang Wen (S'00-M'08-SM'14) received the Ph.D. degree in electrical engineering and computer science, with a minor in western literature, from the Massachusetts Institute of Technology (MIT), Cambridge, MA, USA. He previously worked with Cisco, San Jose, CA, USA, where he lead product development in content delivery networks, which had a revenue impact of three billion US dollars globally. He is currently an assistant professor with the School of Computer Engineering at Nanyang Technological University, Singapore. He has published over 90 papers in top journals and prestigious conferences. His research interests include cloud computing, green data center, big data analytics, multimedia networks, and mobile computing. His latest work in multi-screen cloud social TV has been recognized with the ASEAN ICT Award 2013 (Gold Medal) and the IEEE Globecom 2013 Best Paper Award. He serves on the editorial boards for IEEE TRANS. MULTIMEDIA, IEEE ACCESS, and *Elsevier Ad Hoc Networks*.

Han Hu received the B.E. and Ph.D. degrees from the Department of Automation, University of Science and Technology of China, Hefei, China, in 2007 and 2012, respectively. He is currently a research fellow with the School of Computing, National University of Singapore, Singapore. His research interests include green data center, cloud assisted media network, and big data analytics.

Marie-Jose Montpetit received the Ph.D degree in EECS from the École Polytechnique, Montreal, Quebec, Canada. In 201,4 she was an invited lecturer on Social Television at the Université du Québec, Montreal, Quebec, Canada. She is currently an Invited Scientist and Lecturer with the Massachusetts Institute of Technology (MIT) Comparative Media Studies Department, MIT, Cambridge, MA, USA, and a collaborator with the MIT Communications Futures Programs, MIT, Cambridge, USA. She is also an advisor to Boston area startups in video and social networking, a technology reviewer at the European Union, Brussels, Belgium, and a participant of many standardization bodies. Her main research is on the convergence of networking and video for seamless mobility and social viewing. Her pioneering work on social, wireless, and multi-screen television is recognized worldwide and she was awarded a Motorola Technology Prize in 2007, an MIT Technology Review TR10 in 2010, and a Montreal-Video Award and TEDx presentations in 2012 and 2014.