AMES-Cloud: A Framework of Adaptive Mobile Video Streaming and Efficient Social Video Sharing in the Clouds

Xiaofei Wang, *Student Member, IEEE*, Min Chen, *Senior Member, IEEE*, Ted Taekyoung Kwon, *Member, IEEE*, LaurenceT. Yang, *Senior Member, IEEE*, and Victor C. M. Leung, *Fellow, IEEE*

***Abstract*—While demands on video traffic over mobile networks have been souring, the wireless link capacity cannot keep up with the traffic demand. The gap between the traffic demand and the link capacity, along with time-varying link conditions, results in poor service quality of video streaming over mobile networks such as long buffering time and intermittent disruptions. Leveraging the cloud computing technology, we propose a new mobile video streaming framework, dubbed AMES-Cloud, which has two main parts: adaptive mobile video streaming (AMoV) and efficient social video sharing (ESoV). AMoV and ESoV construct a private agent to provide video streaming services efficiently for each mobile user. For a given user, AMoV lets her private agent adaptively adjust her streaming flow with a scalable video coding technique based on the feedback of link quality. Likewise, ESoV monitors the social network interactions among mobile users, and their private agents try to prefetch video content in advance. We implement a prototype of the AMES-Cloud framework to demonstrate its performance. It is shown that the private agents in the clouds can effectively provide the adaptive streaming, and perform video sharing (i.e., prefetching) based on the social network analysis.**

***Index Terms*—Adaptive video streaming, cloud computing, mobile networks, scalable video coding, social video sharing.**

I. Introduction

OVER the past decade, increasingly more traffic is accounted by video streaming and downloading. In particular, video streaming services over mobile networks have become prevalent over the past few years [1]. While the video streaming is not so challenging in wired networks, mobile networks have been suffering from video traffic transmissions over scarce bandwidth of wireless links. Despite network operators' desperate efforts to enhance the wireless link bandwidth (e.g., 3G and LTE), soaring video traffic demands from mobile users are rapidly overwhelming the wireless link capacity.

While receiving video streaming traffic via 3G/4G mobile networks, mobile users often suffer from long buffering time and intermittent disruptions due to the limited bandwidth and link condition fluctuation caused by multi-path fading and user mobility [2]–[4]. Thus, it is crucial to improve the service quality of mobile video streaming while using the networking and computing resources efficiently [5]–[8].

Recently there have been many studies on how to improve the service quality of mobile video streaming on two aspects:

- **Scalability:** Mobile video streaming services should support a wide spectrum of mobile devices; they have different video resolutions, different computing powers, different wireless links (like 3G and LTE) and so on. Also, the available link capacity of a mobile device may vary over time and space depending on its signal strength, other users traffic in the same cell, and link condition variation. Storing multiple versions (with different bit rates) of the same video content may incur high overhead in terms of storage and communication. To address this issue, the Scalable Video Coding (SVC) technique (Annex G extension) of the H.264 AVC video compression standard [9]–[11] defines a base layer (BL) with multiple enhance layers (ELs). These substreams can be encoded by exploiting three scalability features: (i) spatial scalability by layering image resolution (screen pixels), (ii) temporal scalability by layering the frame rate, and (iii) quality scalability by layering the image compression. By the SVC, a video can be decoded/played at the lowest quality if only the BL is delivered. However, the more ELs can be delivered, the better quality of the video stream is achieved.
- **Adaptability:** Traditional video streaming techniques designed by considering relatively stable traffic links between servers and users, perform poorly in mobile environments [2]. Thus the fluctuating wireless link status should be properly dealt with to provide 'tolerable" video streaming services. To address this issue, we have to adjust the video bit rate adapting to the currently time-varying available link bandwidth of each mobile user. Such adap-

Manuscript received March 02, 2012; revised July 30, 2012; accepted November 13, 2012. Date of publication January 11, 2013; date of current version May 13, 2013. This work was supported by the Korea Communications Commission, Korea, under the R&D program supervised by the Korea Communications Agency under Grant KCA-2012-11-911-05-002. The work of M. Chen was supported in part by Youth 1000 Talent Program and the Program for New Century Excellent Talents in University. The associate editor coordinating the review of this manuscript and approving it for publication was Yonggang Wen.

X. Wang and T. T. Kwon are with the Department of Computer Science and Engineering, Seoul National University, Seoul 151-742, Korea (e-mail: dobbymmlab@gmail.com; tkkwon@snu.ac.kr).

M. Chen is with the School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China (e-mail: minchen@ieee.org).

L. T. Yang is with the School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China, and also with the Department of Computer Science, St. Francis Xavier University, Antigonish, NS, Canada B2G 2W5 (e-mail: ltyang@gmail.com).

V. C. M. Leung is with the Department of Electrical and Computer Engineering, The University of British Columbia, Vancouver, BC, Canada V6T 1Z4 (e-mail: vleung@ece.ubc.ca).

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.2013.2239630

tive streaming techniques can effectively reduce packet losses and bandwidth waste.

Scalable video coding and adaptive streaming techniques can be jointly combined to accomplish effectively the best possible quality of video streaming services. That is, we can dynamically adjust the number of SVC layers depending on the current link status [9], [12].

However most of the proposals seeking to jointly utilize the video scalability and adaptability rely on the active control on the server side. That is, every mobile user needs to individually report the transmission status (e.g., packet loss, delay and signal quality) periodically to the server, which predicts the available bandwidth for each user. Thus the problem is that the server should take over the substantial processing overhead, as the number of users increases.

Cloud computing techniques are poised to flexibly provide scalable resources to content/service providers, and process offloading to mobile users [13]–[19]. Thus, cloud data centers can easily provision for large-scale real-time video services as investigated in [9], [20]. Several studies on mobile cloud computing technologies have proposed to generate personalized intelligent agents for servicing mobile users, e.g., Cloudlet [21] and Stratus [22]. This is because, in the cloud, multiple agent instances (or threads) can be maintained dynamically and efficiently depending on the time-varying user demands.

Recently social network services (SNSs) have been increasingly popular. There have been proposals to improve the quality of content delivery using SNSs [23], [24]. In SNSs, users may share, comment or re-post videos among friends and members in the same group, which implies a user may watch a video that her friends have recommended (e.g., [24]). Users in SNSs can also follow famous and popular users based on their interests (e.g., an official facebook or twitter account that shares the newest pop music videos), which is likely to be watched by its followers.

In this regard, we are further motivated to exploit the relationship among mobile users from their SNS activities in order to prefetch in advance the beginning part of the video or even the whole video to the members of a group who have not seen the video yet. It can be done by a background job supported by the agent (of a member) in the cloud; once the user clicks to watch the video, it can instantly start playing.

In this paper, we design a adaptive video streaming and prefetching framework for mobile users with the above objectives in mind, dubbed AMES-Cloud. AMES-Cloud constructs a private agent for each mobile user in cloud computing environments, which is used by its two main parts: (i) **AMoV** (adaptive mobile video streaming), and **ESoV** (efficient social video sharing). The contributions of this paper can be summarized as follows:

- AMoV offers the best possible streaming experiences by adaptively controlling the streaming bit rate depending on the fluctuation of the link quality. AMoV adjusts the bit rate for each user leveraging the scalable video coding. The private agent of a user keeps track of the feedback information on the link status. Private agents of users are dynamically initiated and optimized in the cloud computing platform. Also the real-time SVC coding is done on the cloud computing side efficiently.
- AMES-Cloud supports distributing video streams efficiently by facilitating a 2-tier structure: the first tier is a content delivery network, and the second tier is a data center. With this structure, video sharing can be optimized within the cloud. Unnecessary redundant downloads of popular videos can be prevented [25], [26].
- Based on the analysis of the SNS activities of mobile users, ESoV seeks to provide a user with instant playing of video clips by prefetching the video clips in advance from her private agent to the local storage of her device. The strength of the social links between users and the history of various social activities can probabilistically determine how much and which video will be prefetched.

The rest of the paper is organized as follows. We first introduce related work in Section II, and explain the AMES-Cloud framework in Section III. The adaptive video streaming service and the efficient social video sharing will be detailed in Sections IV and V, respectively. Then the operations of AMES-Cloud is illustrated in Section VI. Finally, we evaluate the prototype implementation in Section VII, and conclude the paper in Section VIII.

II. Related Work

A. Adaptive Video Streaming Techniques

In the adaptive streaming, the video traffic rate is adjusted on the fly so that a user can experience the maximum possible video quality based on his or her link's time-varying bandwidth capacity [2]. There are mainly two types of adaptive streaming techniques, depending on whether the adaptivity is controlled by the client or the server. The Microsoft's Smooth Streaming [27] is a live adaptive streaming service which can switch among different bit rate segments encoded with configurable bit rates and video resolutions at servers, while clients dynamically request videos based on local monitoring of link quality. Adobe and Apple also developed client-side HTTP adaptive live streaming solutions operating in the similar manner. There are also some similar adaptive streaming services where servers controls the adaptive transmission of video segments, for example, the Quavlive Adaptive Streaming. However, most of these solutions maintain multiple copies of the video content with different bit rates, which brings huge burden of storage on the server.

Regarding rate adaptation controlling techniques, TCP-friendly rate control methods for streaming services over mobile networks are proposed [28], [29], where TCP throughput of a flow is predicted as a function of packet loss rate, round trip time, and packet size. Considering the estimated throughput, the bit rate of the streaming traffic can be adjusted. A rate adaptation algorithm for conversational 3G video streaming is introduced by [30]. Then, a few cross-layer adaptation techniques are discussed [31], [32], which can acquire more accurate information of link quality so that the rate adaptation can be more accurately made. However, the servers have to always control and thus suffer from large workload.

Recently the H.264 Scalable Video Coding (SVC) technique has gained a momentum [10]. An adaptive video streaming system based on SVC is deployed in [9], which studies the real-time SVC decoding and encoding at PC servers. The work in [12] proposes a quality-oriented scalable video delivery

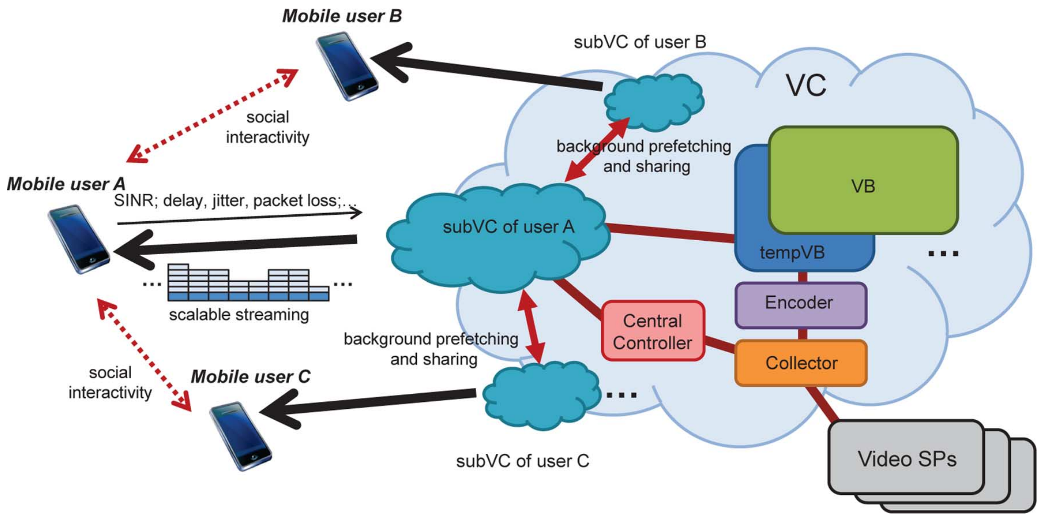


Fig. 1. Illustration of the AMES-Cloud framework with the Video Cloud (VC), subVCs for mobile users, the Video Base (VB), and the Video Service Providers (SPs).

using SVC, but it is only tested in a simulated LTE Network. Regarding the encoding performance of SVC, CloudStream mainly proposes to deliver high-quality streaming videos through a cloud-based SVC proxy [20], which discovered that the cloud computing can significantly improve the performance of SVC coding. The above studies motivate us to use SVC for video streaming on top of cloud computing.

B. Mobile Cloud Computing Techniques

The cloud computing has been well positioned to provide video streaming services, especially in the wired Internet because of its scalability and capability [13]. For example, the quality-assured bandwidth auto-scaling for VoD streaming based on the cloud computing is proposed [14], and the CALMS framework [33] is a cloud-assisted live media streaming service for globally distributed users. However, extending the cloud computing-based services to mobile environments requires more factors to consider: wireless link dynamics, user mobility, the limited capability of mobile devices [34], [35]. More recently, new designs for users on top of mobile cloud computing environments are proposed, which virtualize private agents that are in charge of satisfyinh the requirements (e.g., QoS) of individual users such as Cloudlets [21] and Stratus [22]. Thus, we are motivated to design the AMES-Cloud framework by using virtual a gents in the cloud to provide adaptive video streaming services.

III. AMES-Cloud Framework

In this section we explain the AMES-Cloud framework includes the Adaptive Mobile Video streaming (AMoV) and the Efficient Social Video sharing (ESoV).

As shown in Fig. 1, the whole video storing and streaming system in the cloud is called the Video Cloud (VC). In the VC, there is a large-scale video base (VB), which stores the most of the popular video clips for the video service providers (VSPs). A temporal video base (tempVB) is used to cache new candidates for the popular videos, while tempVB counts the access frequency of each video. The VC keeps running a collector to seek videos which are already popular in VSPs, and will re-encode the collected videos into SVC format and store into tempVB first. By this 2-tier storage, the AMES-Cloud can keep serving most of popular videos eternally. Note that management work will be handled by the controller in the VC.

Specialized for each mobile user, a sub-video cloud (subVC) is created dynamically if there is any video streaming demand from the user. The sub-VC has a sub video base (subVB), which stores the recently fetched video segments. Note that the video deliveries among the subVCs and the VC in most cases are actually not "copy", but just "link" operations on the same file eternally within the cloud data center [36]. There is also encoding function in subVC (actually a smaller-scale encoder instance of the encoder in VC), and if the mobile user demands a new video, which is not in the subVB or the VB in VC, the subVC will fetch, encode and transfer the video. During video streaming, mobile users will always report link conditions to their corresponding subVCs, and then the subVCs offer adaptive video streams. Note that each mobile device also has a temporary caching storage, which is called local video base (localVB), and is used for buffering and prefetching.

Note that as the cloud service may across different places, or even continents, so in the case of a video delivery and prefetching between different data centers, an transmission will be carried out, which can be then called "copy". And because of the optimal deployment of data centers, as well as the capable links among the data centers, the "copy" of a large video file takes tiny delay [36].

IV. AMoV: Adaptive Mobile Video Streaming

A. SVC

As shown in Fig. 2, traditional video streams with fixed bit rates cannot adapt to the fluctuation of the link quality. For a particular bit rate, if the sustainable link bandwidth varies much, the video streaming can be frequently terminated due to the packet loss.

In SVC, a combination of the three lowest scalability is called the Base Layer (BL) while the enhanced combinations are called Enhancement Layers (ELs). To this regard, if BL

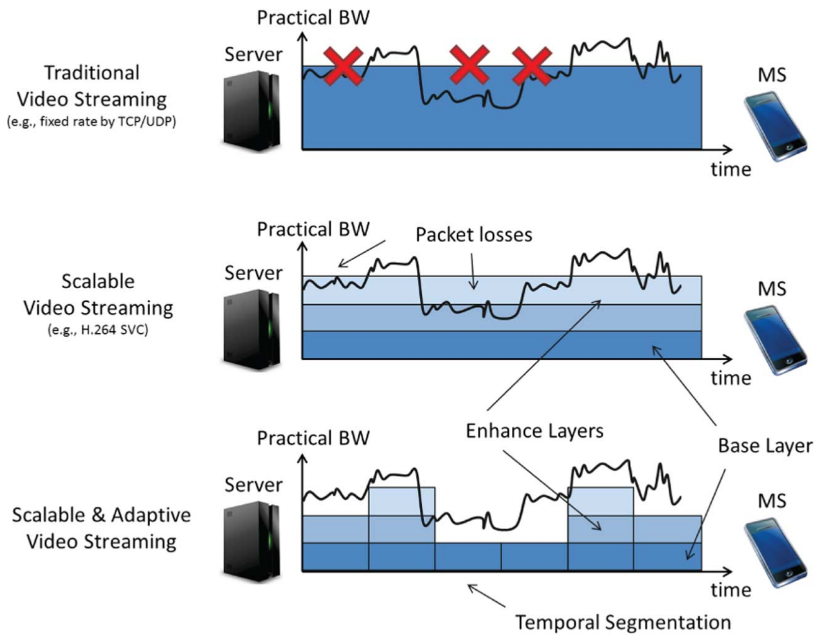


Fig. 2. A comparison of the traditional video streaming, the scalable video streaming and the streaming in the AMES-Cloud framework.

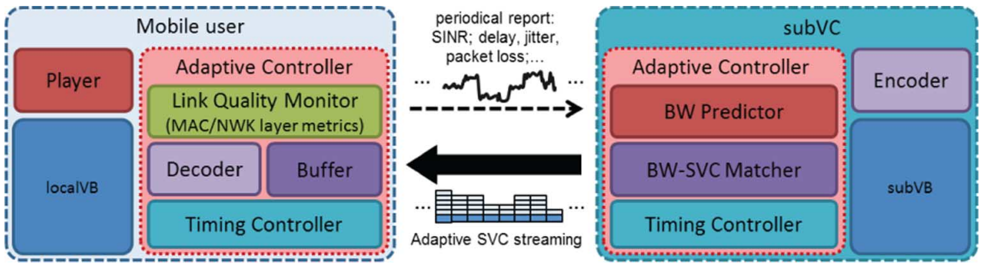


Fig. 3. Functional structure of the client and the subVC.

is guaranteed to be delivered, while more ELs can be also obtained when the link can afford, a better video quality can be expected.

By using SVC encoding techniques, the server doesn't need to concern the client side or the link quality. Even some packets are lost, the client still can decode the video and display. But this is still not bandwidth-efficient due to the unnecessary packet loss. So it is necessary to control the SVC-based video streaming at the server side with the rate adaptation method to efficiently utilize the bandwidth.

B. *Adaptability With Monitoring on Link Quality*

We design the mobile client and the subVC with the structure as shown in Fig. 3. The link quality monitor at mobile client keeps tracking on metrics including signal strength, packet round-trip-time (RTT), jitter and packet loss with a certain duty cycle. And the client will periodically report to the subVC. Hereby we define the cycle period for the reporting as the "time window", denoted by T_{win}, Note that the video is also split by temporal segmentation by interval T_{win}.

Once the subVC gets the information of the link quality, it will perform a calculation and predict the potential bandwidth in the next time window. Note that we will use "predicted bandwidth" and "predicted goodput" interchangeably in following parts.

Suppose sequence number of current time window is i, the predicted bandwidth can be estimated by:

$$BW_{i+1}^{estimate} = BW_i^{practical} \cdot [\alpha \cdot f(p_i, p_{i-1}) \\ + \beta \cdot g(RTT_i, RTT_{i-1}) + \gamma \cdot h(SINR_i, SINR_{i-1})]$$

where, $\alpha + \beta + \gamma = 1$ indicating the importance of each factor, p is for packet loss rate, RTT is for RTT, $SINR$ is for the signal to interference and noise ratio, and $f()$, $g()$, $h()$ are three functions reflecting the value change of each factor compared with that of last time window.

Actually in this paper we deploy a measurement-based prediction, that is we directly use $BW_i^{practical}$ of last time window as the $BW_{i+1}^{estimate}$ of next time window, which is proved with already high accuracy [37].

C. *Matching Between Bandwidth Prediction and SVC Segments*

After obtaining the predicted bandwidth, or say goodput, of next time window, subVC will match and decide how many video segments of BL and ELs can be transmitted approximately. We hereby define the term "resolution" to indicate the level of temporal segmentation and the number of ELs. If T_{win} is small and there are more ELs, we say the SVC-based video source is with a higher resolution. We illustrate two cases of

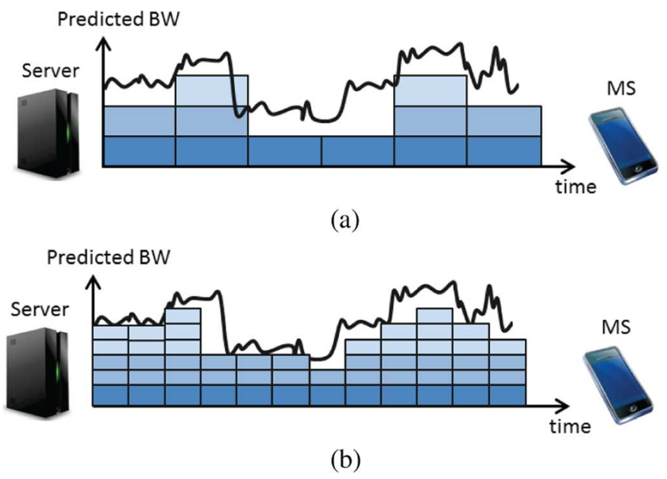


Fig. 4. Matching between predicted bandwidth and SVC-segments with different resolutions. (a) Fine-grained (high resolution). (b) Coarse-grained (low resolution).

coarse-grained (low resolution) and a relatively fine-grained (high resolution) for matching between the SVC segments and the predicted goodput in Fig. 4. The resolution with two ELs and a larger T_{win} can hardly fit to the signal fluctuation, and thus there are some bandwidth wasted or packets lost. In contrast a higher resolution with more ELs and a smaller T_{win} can always fit the fluctuation of the bandwidth. However a higher resolution also induces more encoding workload to the servers.

Suppose there are totally j ELs, and the bit rate of the jth EL is denoted as R_{EL^j} while the bit rate of the BL is R_{BL}). We let BL_i indicate the SVC segment of BL with temporal sequence i, and let EL_i^j indicate the SVC segment of the jth EL with temporal sequence i. So the algorithm of matching between predicted bandwidth and SVC segments is shown in Algorithm 1 as following:

Algorithm 1 Matching Algorithm between BW and Segments

```
$i = 0$
$BW_0 = R_{BL}$
Transmit $BL_0$
Monitor $BW_0^{practical}$
**repeat**
    Sleep for $T_{win}$
    Obtain $p_i$, $RTT_i$, $SINR_i$ etc., from client's report
    Predict $BW_{i+1}^{estimate}$ (or $BW_{i+1}^{estimate} = BW_i^{practical}$)
    $k = 0$
    $BW_{EL} = 0$
    **repeat**
        $k++$
        if $k >= j$ break
        $BW_{EL} = BW_{EL} + R_{EL^k}$
    **until** $BW_{EL} >= BW_{i+1}^{estimate} - R_{BL}$
    Transmit $BL_{i+1}$ and $EL_{i+1}^1, EL_{i+1}^2, \ldots, EL_{i+1}^{k-1}$
    Monitor $BW_{i+1}^{practical}$
    $i++$
**until** All video segments are transmitted
```

V. ESOV: EFFICIENT SOCIAL VIDEO SHARING

A. *Social Content Sharing*

In SNSs, users subscribe to known friends, famous people, and particular interested content publishers as well; also there are various types of social activities among users in SNSs, such as direct message and public posting. For spreading videos in SNSs, one can post a video in the public, and his/her subscribers can quickly see it; one can also directly recommend a video to specified friend(s); furthermore one can periodically get noticed by subscribed content publisher for new or popular videos.

Similar to studies in [23], [24], we define different strength levels for those social activities to indicate the probability that the video shared by one user may be watched by the receivers of the one's sharing activities, which is called a "hitting probability", so that subVCs can carry out effective background prefetching at subVB and even localVB. Because after a video sharing activity, there may be a certain delay that the recipient gets to know the sharing, and initiates to watch [38]. Therefore the prefetching in prior will not impact the users at most cases. Instead, a user can click to see without any buffering delay as the beginning part or even the whole video is already prefetched at the localVB. The amount of prefetched segments is mainly determined by the strength of the social activities. And the prefetching from VC to subVC only refers to the "linking" action, so there is only file locating and linking operations with tiny delays; the prefetching from subVC to localVB also depends on the strength of the social activities, but will also consider the wireless link status.

We classify the social activities in current popular SNSs into three kinds, regarding the impact of the activities and the potential reacting priority from the point of view of the recipient:

- **Subscription:** Like the popular RSS services, an user can subscribe to a particular video publisher or a special video collection service based on his/her interests. This interest-driven connectivity between the subscriber and the video publisher is considered as "median", because the subscriber may not always watch all subscribed videos.
- **Direct recommendation:** In SNSs, an user directly recommend a video to particular friend(s) with a short message. The recipients of the message may watch it with very high probability. This is considered as "strong".
- **Public sharing:**Each user in SNSs has a timeline-based of activity stream, which shows his/her recent activities. The activity of a user watching or sharing a video can be seen by his/her friends (or followers). We consider this public sharing with the "weak" connectivity among users, because not many people may watch the video that one has seen without direct recommendation.

TABLE I
SOCIAL ACTIVITIES AND BACKGROUND PUSHING STRATEGIES

	Direct recommendation	Subscription	Public sharing
VB→subVB	All	Parts	Little
subVB→locVB (via Wi-Fi)	All	Parts	Little
subVB→locVB (via 3G/4G)	Parts	Little	None

B. Prefetching Levels

Different strengths of the social activities indicate different levels of probability that a video will be soon watched by the recipient. Correspondingly we also define three prefetching levels regarding the social activities of mobile users:

- **"Parts":** Because the videos that published by **subscriptions** may be watched by the subscribers with a not high probability, we propose to only push a part of BL and ELs segments, for example, the first 10% segments.
- **"All":** The video shared by the **direct recommendations** will be watched with a high probability, so we propose to prefetch the BL and all ELs, in order to let the recipient(s) directly watch the video with a good quality, without any buffering.
- **"Little":** The **public sharing** has a weak connectivity among users, so the probability that a user's friends (followers) watch the video that the user has watched or shared is low. We propose to only prefetch the BL segment of the first time window in the beginning to those who have seen his/her activity in the stream.

The prefetching happens among subVBs and the VB, also more importantly, will be performed from the subVB to localVB of the mobile device depending on the link quality. If a mobile user is covered by Wi-Fi access, due to Wi-Fi's capable link and low price (or mostly for free), subVC can push as much as possible in most cases. However if it is with a 3G/4G connection, which charges a lot and suffers limited bandwidth, we propose to downgrade the prefetching level to save energy and cost as listed in Table I, but users can still benefit from the prefetching effectively. Note that some energy prediction methods can be deployed in order to actively decide whether current battery status is suitable for "parts" or "little" [39]. If a user, A, gets the direct recommendation of a video from another user, B, A's subVC will immediately prefetch the video either from B's subVB, or from the VB (or tempVB) at the level of "all", if A is with Wi-Fi access. However if user A is connected to 3G/4G link, we will selectively prefetch a part of the video segment to A's local storage at the level of "parts". Note that the subscribed videos will be not prefetched when user A is at 3G/4G connection, as it is downgraded from "little" to none.

A better extension of the prefetching strategy by social activities can be designed by an self-updating mechanism from the user's hitting history in an evolutionary manner. This learning-based prefetching is out of the scope of this paper, and will be explored as our future work.

VI. VIDEO STORAGE AND STREAMING FLOW BY AMOV AND EMOS

The two parts, AMoV and EMoS, in AMES-Cloud framework have tight connections and will together service the video streaming and sharing: they both rely on the cloud computing platform and are carried out by the private agencies of users; while prefetching in EMoS, the AMoV will still monitor and improve the transmission considering the link status; with a certain amount of prefetched segments by EMoS, AMoV can offer better video quality.

With the efforts of AMoV and EMoS, we illustrate the flow chart of how a video will be streamed in Fig. 5. Note that in order to exchange the videos among the localVBs, subVBs, tempVB and the VB, a video map (VMap) is used to indicate the required segments.

Once a mobile user starts to watch a video by a link, the localVB will first be checked whether there is any prefetched segments of the video so that it can directly start. If there is none or just some parts, the client will report a corresponding VMap to its subVC. if the subVC has prefetched parts in subVB, the subVC will initiate the segment transmission. But if there is also none in the subVB, the tempVB and VB in the center VC will be checked. For a non-existing video in AMES-Cloud, the collector in VC will immediately fetch it from external video providers via the link; after re-encoding the video into SVC format, taking a bit longer delay, the subVC will transfer to the mobile user.

Also in AMES-Cloud, if a video is shared among the subVCs at a certain frequency threshold (e.g., 10 times per day), it will be uploaded to the tempVB of the VC; and if it is further shared at a much higher frequency (e.g., 100 times per day), it will be stored with a longer lifetime in the VB. In such a manner, which is quite similar to the leveled CPU cache, the subVB and VB can always store fresh and popular videos in order to increase the probability of re-usage.

VII. IMPLEMENTATION AND EVALUATION

We evaluate the performance of the AMES-Cloud framework by a prototype implementation. We choose the U-cloud server (premium) in the cloud computing service offered by Korean Telecom, and utilize the virtual server with 6 virtual CPU cores (2.66 GHz) and 32 GB memory, which is fast enough for encoding 480P (480 by 720) video with H.264 SVC format in 30 fps at real time [9]. In the cloud, we deploy our server application based on Java, including one main program handling all tasks of the whole VC, while the program dynamically initializes, maintains and terminates instances of another small Java application as private agents for all active users. We implement the mobile client at a mobile phone, Samsung Galaxy II, with android system version 4.0. The mobile data service is offered by LG $U+$ LTE network, while in some uncovered area the 3G network is used. Note that we still use "3G" to indicate the general cellular network. We test in the downtown area, so the practical bandwidth of the mobile link is not as high as we expected, but this won't impact our experiment results.

The test video is the Tomb Raider 2012 Trailer in H.264 format with 480P resolution downloaded from YouTube. Its size

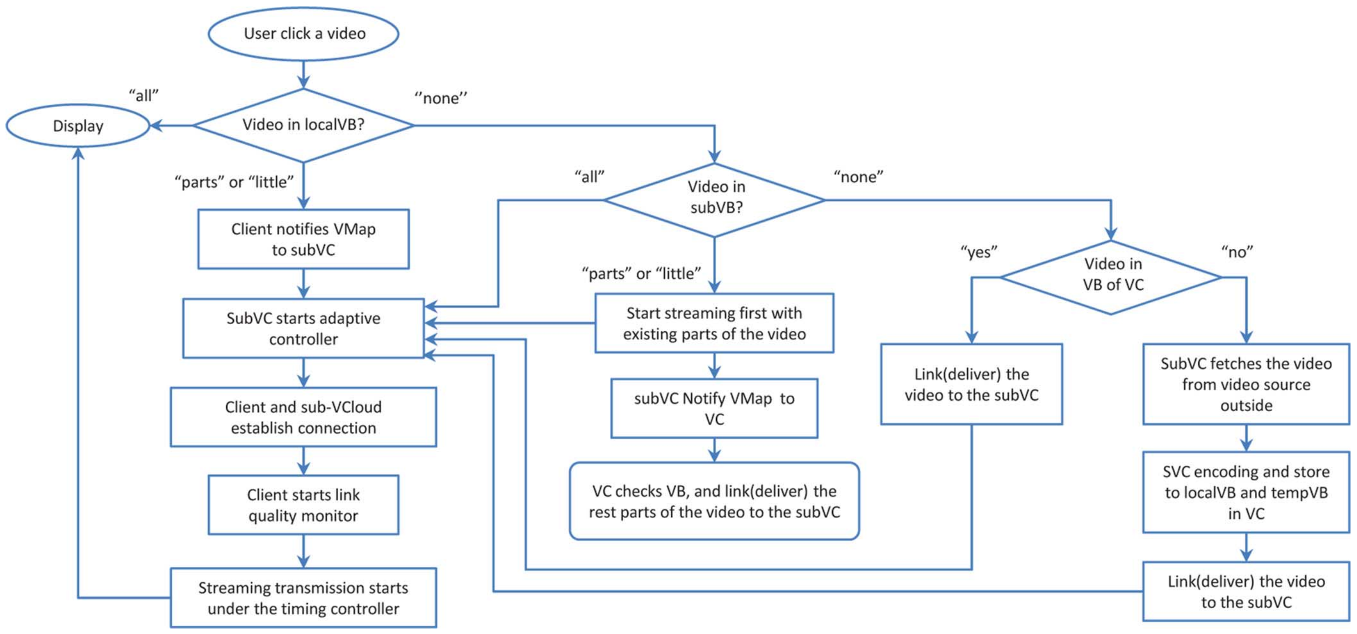

User click a video
Video in localVB?
"all"
Display
"none"
"parts" or "little"
Client notifies VMap to subVC
SubVC starts adaptive controller
Client and sub-VCloud establish connection
Client starts link quality monitor
Streaming transmission starts under the timing controller
"all"
Video in subVB?
"none"
"parts" or "little"
Start streaming first with existing parts of the video
subVC Notify VMap to VC
VC checks VB, and link(deliver) the rest parts of the video to the subVC
"yes"
Video in VB of VC
"no"
Link(deliver) the video to the subVC
SubVC fetches the video from video source outside
SVC encoding and store to localVB and tempVB in VC
Link(deliver) the video to the subVC

Fig. 5. Working flow of video streaming in the subVC and VC of AMES-Cloud framework.

is 13.849 Mbytes and with a duration of 180 seconds. We first decode it by the x264 decoder into the YUV format, and re-encode it by the H.264 SVC encoder, the Joint Scalable Video Model (JSVM) software of version 9.1 [40]. We just use default settings for the decoding and encoding, and do the H.264 SVC encoding at the virtual server in the cloud.

We split the video into segments by 1 second to 5 seconds, that is to vary T_{win} with values 1s, 2s, 3s, 4s and 5s. By JSVM, besides the base layer, we further make five temporal layers (1.875, 3.75, 7.5, 15, and 15 fps), two spatial layers (240 by 360 and 120 by 180) and two more quality layer (low and high), referring to [12] and [40]. Thus we define the best resolution configuration as "$1+5+2+2$". And we also test different resolution configurations, including "$1+1+1+1$", "$1+2+2+2$", "$1+3+2+2$" and "$1+4+2+2$".

A. Adaptive Video Streaming Based on SVC

Firstly we examine whether there is a deep relationship between the measured bandwidth of last time window and the practical bandwidth of next time window (goodput by Kbps). We test the video streaming service via cellular link, and move the device around in the building to try to change the signal quality. Note that all tests are ran five times. The collected the relative errors for the predicted bandwidth to the practical bandwidth for every time window, calculated by $\left|BW^{estimate} - BW^{practical}\right|/\{BW_{practical}\}$, are shown in Fig. 6, where the bar indicates the 25% and 75% quartiles, and the whiskers indicate the 5% and 95% percentiles. When T_{win} is 1 second or 2 seconds, the predicted bandwidth is very near to the practical one with around 10% relative error, but large values of T_{win} have relatively poor prediction accuracy, which reflects the similar results [37]. So we suggest a short T_{win} of 2 or 3 seconds for accurate prediction in practical designs.

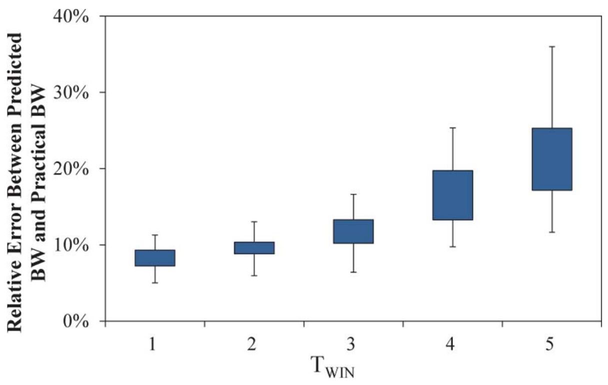
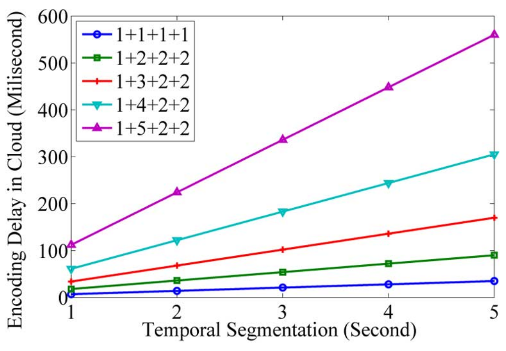


Fig. 6. Relative errors between predicted bandwidth and practical bandwidth (percentage).

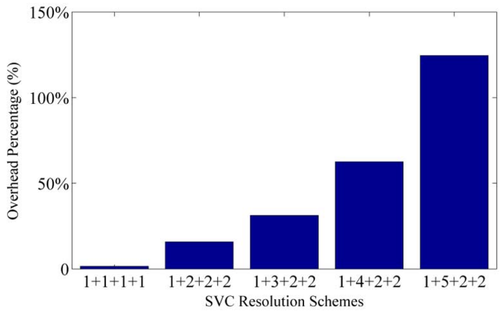
B. Video Streaming in SubVC and VC

We evaluate how H.264 SVC works in AMES-Cloud framework regarding the above mentioned SVC resolution configurations. As shown in Fig. 7(a), because of the strong computational capacity by the cloud computing, the encoding speed is fast. The best resolution configuration "$1+5+2+2$" with 5 second temporal segmentation scheme requires about 560 ms for encoding. For shorter intervals of T_{win}, the encoding delay is very small under 50 ms.

Because more ELs induce higher overhead due to the duplicated I-frames, we test the overhead, which is calculated by the ratio of the total size of the video segments after SVC encoding to the size of only the BL. As shown in Fig. 7(b), the resolution scheme of "$1+1+1+1$" has a low overhead around below 10%, and "$1+2+2+2$" with two ELs for each scalability feature has about 17% overhead, which is acceptable. However higher resolution like "$1+4+2+2$" has 61% overhead, and "$1+5+2+2$" has even 120% overhead, which is not efficient. Overall, an SVC stream should not contain too many en-



(a)



(b)

Fig. 7. Evaluation of SVC resolution schemes. (a) Delay of difference SVC resolution schemes in the Cloud (b) Overhead of different SVC resolutions schemes in the Cloud.

TABLE II
DELAYS OF PREFETCHING SHARING FOR VARIOUS LEVELS

	Little	Parts	All
subVBs↔VB	0.011 s	0.023 s	0.098 s
subVB→locVB via Wi-Fi	2.421 s	4.359 s	23.221 s
subVB→locVB via 3G	N&A	18.430 s (little)	37.308 s (parts)

hance layers for extremely high scalability, which may practically bring too much overhead.

C. Prefetching Delays

In ESoV, video segments can be prefetched among VB, tempVB, and localVBs of the mobile users, based on their activities in SNSs. we evaluate the required delays for different levels of prefetching as shown in Table. 2. We here use the normal resolution configuration of "$1 + 2 + 2 + 2$" with 2 second temporal segmentation by default (the same in following tests). We also set the sharing length of "little" as only the first 5 seconds of the BL and ELs, that of "parts" as the first 15 seconds of the BL and ELs, and that of "all" as all BL and ELs segments.

We can see that prefetching supported by the cloud computing is significantly fast. When prefetching via wireless links, it takes several seconds. However it is obvious that in most cases [26], [38] a recipient of the video sharing may not watch immediately after the original sharing behavior, that is normal users have significant access delay gaps, so this prefetching transmission delay won't impact user's experience at all, but will bring "non-buffering" experience in fact when the user clicks to watch at a later time.

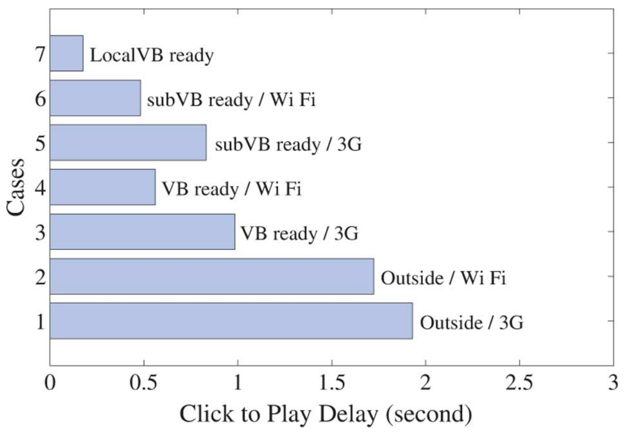


Fig. 8. Average click-to-play delay for various cases.

D. Watching Delay

We test how long one user has to wait from the moment that one clicks the video in the mobile device to the moment that the first streaming segment arrives, which is called as "click-to-play" delay. As shown in Fig. 8, if the video has been cached in localVB, the video can be displayed nearly immediately with ignorable delay. When we watch video which is fetched from the subVC or the VC, it generally takes no more than 1 second to start. However if the user accesses to AMES-Cloud service via the cellular link, he will still suffer a bit longer delay (around 1s) due to the larger RTT of transmission via the cellular link.

For the cases to fetch videos which are not in the AMES-Cloud (but in our server at lab), the delay is a bit higher. This is mainly due to the fetching delay via the link from our server at lab to the cloud data center, as well as the encoding delay. In practical, there are be optimized links in the Internet backbone among video providers and cloud providers, and even recent video providers are just using cloud storage and computing service. Therefore this delay can be significantly reduced in practice. Also this won't happen frequently, since most of the popular videos will be already prepared in the AMES-Cloud.

VIII. CONCLUSION

In this paper, we discussed our proposal of an adaptive mobile video streaming and sharing framework, called AMES-Cloud, which efficiently stores videos in the clouds (VC), and utilizes cloud computing to construct private agent (subVC) for each mobile user to try to offer "non-terminating" video streaming adapting to the fluctuation of link quality based on the Scalable Video Coding technique. Also AMES-Cloud can further seek to provide "non-buffering" experience of video streaming by background pushing functions among the VB, subVBs and localVB of mobile users. We evaluated the AMES-Cloud by prototype implementation and shows that the cloud computing technique brings significant improvement on the adaptivity of the mobile streaming.

The focus of this paper is to verify how cloud computing can improve the transmission adaptability and prefetching for mobile users. We ignored the cost of encoding workload in the cloud while implementing the prototype. As one important future work, we will carry out large-scale implementation and with serious consideration on energy and price cost. In the future, we will also try to improve the SNS-based prefetching, and security issues in the AMES-Cloud.

REFERENCES

[1] "Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2011–2016," CISCO, 2012.
[2] Y. Li, Y. Zhang, and R. Yuan, "Measurement and analysis of a large scale commercial mobile Internet TV system," in *Proc. ACM Internet Meas. conf.*, 2011, pp. 209–224.
[3] T. Taleb and K. Hashimoto, "MS2: A novel multi-source mobile-streaming architecture," *IEEE Trans. Broadcasting*, vol. 57, no. 3, pp. 662–673, Sep. 2011.
[4] X. Wang, S. Kim, T. Kwon, H. Kim, and Y. Choi, "Unveiling the bittorrent performance in mobile WiMAX networks," in *Proc. Passive Active Meas. Conf.*, 2011, pp. 184–193.
[5] A. Nafaa, T. Taleb, and L. Murphy, "Forward error correction adaptation strategies for media streaming over wireless networks," *IEEE Commun. Mag.*, vol. 46, no. 1, pp. 72–79, Jan. 2008.
[6] J. Fernandez, T. Taleb, M. Guizani, and N. Kato, "Bandwidth aggregation-aware dynamic QoS negotiation for real-time video applications in next-generation wireless networks," *IEEE Trans. Multimedia*, vol. 11, no. 6, pp. 1082–1093, Oct. 2009.
[7] T. Taleb, K. Kashibuchi, A. Leonardi, S. Palazzo, K. Hashimoto, N. Kato, and Y. Nemoto, "A cross-layer approach for an efficient delivery of TCP/RTP-based multimedia applications in heterogeneous wireless networks," *IEEE Trans. Veh. Technol.*, vol. 57, no. 6, pp. 3801–3814, Jun. 2008.
[8] K. Zhang, J. Kong, M. Qiu, and G. L. Song, "Multimedia layout adaptation through grammatical specifications," *ACM/Springer Multimedia Syst.*, vol. 10, no. 3, pp. 245–260, 2005.
[9] M. Wien, R. Cazoulat, A. Graffunder, A. Hutter, and P. Amon, "Real-time system for adaptive video streaming based on SVC," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 17, no. 9, pp. 1227–1237, Sep. 2007.
[10] H. Schwarz, D. Marpe, and T. Wiegand, "Overview of the scalable video coding extension of the H.264/AVC standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 17, no. 9, pp. 1103–1120, Sep. 2007.
[11] H. Schwarz and M. Wien, "The scalable video coding extension of the H. 264/AVC standard," *IEEE Signal Process. Mag.*, vol. 25, no. 2, pp. 135–141, Feb. 2008.
[12] P. McDonagh, C. Vallati, A. Pande, and P. Mohapatra, "Quality-oriented scalable video delivery using H. 264 SVC on an LTE network," in *Proc. WPMC*, 2011.
[13] Q. Zhang, L. Cheng, and R. Boutaba, "Cloud computing: State-of-the-art and research challenges," *J. Internet Services Applic.*, vol. 1, no. 1, pp. 7–18, Apr. 2010.
[14] D. Niu, H. Xu, B. Li, and S. Zhao, "Quality-assured cloud bandwidth auto-scaling for video-on-Demand applications," in *Proc. IEEE INFOCOM*, 2012, pp. 460–468.
[15] Y. G. Wen, W. W. Zhang, K. Guan, D. Kilper, and H. Y. Luo, "Energy-optimal execution policy for a Cloud-assisted mobile application platform," Sep. 2011.
[16] W. W. Zhang, Y. Wen, and D. P. Wu, "Energy-efficient scheduling policy for collaborative execution in mobile cloud computing," in *Proc. INFOCOM Mini-conf.*, 2013.
[17] W. W. Zhang, Y. G. Wen, Z. Z. Chen, and A. Khisti, "QoE-driven cache management for HTTP adaptive bit rate streaming over wireless networks," *IEEE Trans. Multimedia*, Nov. 2012, accepted for publication.
[18] J. Li, M. Qiu, Z. Ming, G. Quan, X. Qin, and Z. Gu, "Online optimization for scheduling preemptable tasks on IaaS cloud systems," *J. Parallel Distrib. Computing*, vol. 72, no. 5, pp. 666–677, 2012.
[19] P. Calyam, M. Sridharan, Y. Xu, K. Zhu, A. Berryman, R. Patali, and A. Venkataraman, "Enabling performance intelligence for application adaptation in the future Internet," *J. Commun. Networks*, vol. 13, no. 6, pp. 591–601, 2011.
[20] Z. Huang, C. Mei, L. E. Li, and T. Woo, "CloudStream: Delivering high-quality streaming videos through a cloud-based SVC proxy," in *Proc. IEEE INFOCOM Mini-conf.*, 2011, pp. 201–205.
[21] N. Davies, "The case for VM-Based Cloudlets in mobile computing," *IEEE Pervasive Computing*, vol. 8, no. 4, pp. 14–23, 2009.
[22] B. Aggarwal, N. Spring, and A. Schulman, "Stratus: Energy-efficient mobile communication using cloud support," in *Proc. ACM SIGCOMM* , 2010, pp. 477–478.
[23] Y. Zhang, W. Gao, G. Cao, T. L. Porta, B. Krishnamachari, and A. Iyengar, "Social-aware data diffusion in delay tolerant MANET," in *Handbook of Optimization in Complex Networks: Communication and Social Networks*. Berlin, Germany: Springer, 2010.
[24] Z. Wang, L. Sun, C. Wu, and S. Yang, "Guiding Internet-scale video service deployment using microblog-based prediction," in *Proc. IEEE INFOCOM Mini-conf.*, 2012, pp. 2901–2905.
[25] Y. Chen, L. Qiu, W. Chen, L. Nguyen, and R. Katz, "Clustering web content for efficient replication," in *Proc. IEEE ICNP*, 2002, pp. 165–174.
[26] M. Cha, H. Kwak, P. Rodriguez, Y. Y. Ahn, and S. Moon, "I tube, you tube, everybody tubes: Analyzing the world's largest user generated content video system," in *Proc. ACM Internet Meas. Conf.*, 2007, pp. 1–14.
[27] A. Zambelli, "IIS smooth streaming technical overview," Microsoft Corp., 2009.
[28] Y. Fu, R. Hu, G. Tian, and Z. Wang, "TCP-friendly rate control for streaming service over 3G network," in *Proc. WiCOM*, 2006, pp. 1–4.
[29] K. Tappayuthpijarn, G. Liebl, T. Stockhammer, and E. Steinbach, "Adaptive video streaming over a mobile network with TCP-friendly rate control," in *Proc. IWCMC*, 2009, pp. 1325–1329.
[30] V. Singh and I. D. D. Curcio, "Rate adaptation for conversational 3G video," in *Proc. IEEE INFOCOM Workshop*, 2009, pp. 205–211.
[31] S. Akhshabi, A. C. Begen, and C. Dovrolis, "An experimental evaluation of rate-adaptation algorithms in adaptive streaming over HTTP," in *Proc. ACM MMSys*, 2011, pp. 157–168.
[32] E. Piri, M. Uitto, J. Vehkaper, and T. Sutinen, "Dynamic cross-layer adaptation of scalable video in wireless networking," in *Proc. IEEE GLOBECOM*, 2010, pp. 1–5.
[33] F. Wang, J. Liu, and M. Chen, "CALMS: Cloud-assisted live media streaming for globalized demands with time/region diversities," in *Proc. IEEE INFOCOM*, 2012, pp. 199–207.
[34] H. T. Dinh, C. Lee, D. Niyato, and P. Wang, "A survey of mobile cloud computing: Architecture, applications, and approaches," *Wiley J. Wireless Commun. Mobile Computing*, Oct. 2011.
[35] S. Chetan, G. Kumar, K. Dinesh, K. Mathew, and M. A. Abhimanyu, "Cloud Computing for Mobile World," 2010.
[36] G. Wang and T. E. Ng, "The impact of virtualization on network performance of amazon EC2 data center," in *Proc. IEEE INFOCOM*, 2010, pp. 1163–1171.
[37] A. Balasubramanian, R. Mahajan, and A. Venkataramani, "Augmenting mobile 3G using WiFi," in *Proc. ACM MobiSys*, 2010, pp. 209–222.
[38] F. Benevenuto, T. Redrigues, V. Almeida, and J. Almeida, "Video interactions in online social networks," *ACM Trans. Multimedia Cmputing, Commun. Applic.*, vol. 5, no. 4, pp. 30–44, 2009.
[39] J. M. Kang, S. S. Seo, and J. W. Hong, "Personalized battery lifetime prediction for mobile devices based on usage patterns," *J. Computing Sci. Eng.*, vol. 5, no. 4, pp. 338–345, 2011.
[40] JSVM. [Online]. Available: http://github.com/kierank/jsvm

Xiaofei Wang is a Ph.D. candidate in the Multimedia and Mobile Communication Laboratory, School of Computer Science and Engineering (CSE), Seoul National University (SNU), Korea. He received the B.S. degree in the Department of Computer Science and Technology of Huazhong University of Science and Technology (HUST) in 2005 and M.S. degree from the School of CSE at SNU in 2008. His current research interests are in the areas of mobile traffic evaluation, and content sharing in mobile content-centric networks.

Min Chen is a professor at Huazhong University of Science and Technology, he was an assistant professor in School of Computer Science and Engineering at SNU. He received the Best Paper Award from IEEE ICC 2012 and Best Paper Runner-up Award from QShine 2008. He has published more than 160 papers. He serves as editor or associate editor for Information Sciences, Wireless Communications and Mobile Computing, IET Communications, etc. He is managing editor for IJAACS and IJART. He is a guest editor for IEEE Network, IEEE Wireless Communications Magazine, etc. He is Co-Chair of IEEE ICC 2012 Communications Theory Symposium, and Co-Chair of IEEE ICC 2013 Wireless Networks Symposium. He is general Co-Chair for the 12th IEEE International Conference on Computer and Information Technology (IEEE CIT 2012). His research focuses on Internet of Things, mobile cloud computing, cloud-assisted mobile computing, ubiquitous network and services, mobile agent, and multimedia transmission over wireless networks, etc.

Ted Taekyoung Kwon has been an associate professor in the School of Computer Science and Engineering, Seoul National University, since 2008. Before joining SNU, he was a postdoctoral research associate at the University of California at Los Anfeles (UCLA) and at City University of New York (CUNY). He obtained B.S., M.S., and Ph.D. degrees from the Department of Computer Engineering, SNU, in 1993, 1995, and 2000, respectively. His research interest lies in sensor networks, wireless networks, IP mobility, and ubiquitous computing.

Laurence T. Yang graduated from Tsinghua University, China and received his Ph.D. in Computer Science from University of Victoria, Canada. He is a professor in School of Computer Science and Technology, Huazhong University of Science and Technology, China and Department of Computer Science, St. Francis Xavier University, Canada. His current research interests include parallel and distributed computing, embedded and ubiquitous/pervasive computing. His research is supported by National Sciences and Engineering Research Council, Canada and Canada Foundation for Innovation.

Victor C. M. Leung received the B.A.Sc. (Hons.) and Ph.D. degrees, both in electrical engineering, from the University of British Columbia, where he holds the positions of Professor and TELUS Mobility Research Chair. Dr. Leung is a registered Professional Engineer in the Province of British Columbia, Canada. He is a Fellow of IEEE, the Engineering Institute of Canada, and the Canadian Academy of Engineering. He has co-authored more than 600 technical papers in the broad areas of wireless networks and mobile systems.