Improving High Energy Efficiency and Streaming Quality in Cloud Based Mobile Systems

K. AnandBabu, J.Venkata Prasad

Abstract*—The rapidly increasing power of personal mobile devices (smartphones, tablets, etc.) is providing muchn richer contents and social interactions to users on the move. This trend however is throttled by the limited battery lifetime of mobile devices and unstable wireless connectivity, making the highest possible quality of service experienced by mobile users not feasible. The recent cloud computing technology, with its rich resources to compensate for the limitations of mobile devices and connections, can potentially provide an ideal platform to support the desired mobile services. Tough challenges arise on how to effectively exploit cloud resources to facilitate mobile services, especially those with stringent interaction delay requirements. In this paper, we propose the design of a Cloud-based, novel Mobile sOcial tV system (CloudMoV). The system effectively utilizes both PaaS (Platform-as-a-Service) and IaaS(Infrastructure-asa-Service) cloud services to offer the living-room experience of video watching to a group of disparate mobile users who can interact socially while sharing the video. To guarantee good streaming quality as experienced by the mobile users with timevarying wireless connectivity, we employ a surrogate for each user in the IaaS cloud for video\ downloading and social exchanges on behalf of the user. The surrogate performs efficient stream transcoding that matches the current connectivity quality of the mobile user. Given the battery life as a key performance bottleneck, we advocate the use of burst transmission from the surrogates to the mobile users, and carefully decide the burst size which can lead to high energy efficiency and streaming quality. Social interactions among the users, in terms of spontaneous textual exchanges, are effectively achieved by efficient designs of data storage with BigTable and dynamic handling of large volumes of concurrent messages in a typical PaaS cloud. These various designs for flexible transcoding capabilities, battery efficiency of mobile devices and spontaneous social interactivity together provide an ideal platform for mobile social TV services. We have implemented CloudMoV on Amazon EC2 and Google App Engine and verified its superior performance based on realworldexperiments.

I.INTRODUCTION

Thanks to the revolutionary “reinventing the phone” campaigns initiated by Apple Inc. in 2007, smartphones nowadays are shipped with multiple microprocessor cores and gigabyte RAMs; they possess more computation power than personal computers of a few years ago. On the other hand, the wide deployment of 3G broadband cellular infrastructures further fuels the trend. Apart from common productivity tasks like emails and web surfing, smartphones are flexing their strengths in more challenging scenarios such as realtime video streaming and online gaming, as well as serving as a main tool for social exchanges.

Although many mobile social or media applications have emerged, truely killer ones gaining mass acceptance are still impeded by the limitations of the current mobile and wireless technologies, among which battery lifetime and unstable connection bandwidth are the most difficult ones. It is natural to resort to cloud computing, the newly-emerged computing paradigm for low-cost, agile, scalable resource supply, to support power-efficient mobile data communication. With virtually infinite hardware and software resources, the cloud can offload the computation and other tasks involved in a mobile application and may significantly reduce battery consumption at the mobile devices, if a proper design is in place. The big challenge in front of us is how to effectively exploit cloud services to facilitate mobile applications. There have been a few studies on designing mobile cloud computing systems , but none of them deal in particular with stringent delay requirements for spontaneous social interactivity among mobile users.In this paper, we describe the design of a novel mobile social TV system, CloudMoV, which can effectively utilize the cloud computing paradigm to offer a living-room experience of video watching to disparate mobile users with spontaneous social interactions. In CloudMoV, mobile users can import a live or on-demand video to watch from any video streaming site, invite their friends to watch the video concurrently, and chat with their friends while enjoying the video. It therefore blends viewing experience and social awareness among friends
on the go. We design CloudMoV to seamlessly utilize agile resource support and rich functionalities offered by both an IaaS (Infrastructure-as-a-Service) cloud and a PaaS (Platform-asa- Service) cloud. Our design achieves the following goals.
Encoding flexibility. Different mobile devices have differently sized displays, customized playback hardwares, and various codecs.
Battery efficiency. A breakdown analysis conducted by
Carroll et al. [6] indicates that the network modules (both Wi-Fi and 3G) and the display contribute to a significant portion of the overall power consumption in a mobile device, dwarfing usages from other hardware modules including CPU, memory, etc.
Spontaneous social interactivity. Multiple mechanisms are included in the design of CloudMoV to enable spontaneous social, co-viewing experience. Portability. A prototype CloudMov system is implemented following the philosophy of “Write Once, Run Anywhere” (WORA): both the front-end mobile modules and the backend server modules are implemented in “100% Pure Java” with well-designed generic data models suitable for any BigTable-like data store; the only exception is the transcoding module, which is implemented using ANSI C for performance reasons and uses no platform-dependent or proprietary APIs.

II. RELATED WORK

A number of mobile TV systems have sprung up in recent years, driven by both hardware and software advances in mobile devices. Some early systems [8][9] bring the “livingroom” experience to small screens on the move. But they focus more on barrier clearance in order to realize the convergence of the television network and the mobile

network, than exploring the demand of "social" interactions among mobile users. There is another trend in which efforts are dedicated to extending social elements to television systems. For any application targeted at mobile devices, reducing power consumption is perennially one of the major concerns and challenges. Flinn exploit collaborations between the mobile OS and the mobile applications to balance the energy conservation and application performance. Yuan etal. investigate mobile multimedia streaming, similar to most of the other work, by adjusting the CPU power for energy saving; however, according to the recent measurement work of Carroll et al. [6], the display and the wireless network card (including the cellular module) and not the CPU consume more than half of the overall power consumption in smart phones nowadays. Our work is able to achieve a significant (about 30%) power saving, by opportunistically switching the device between high-power and low-power transmission modes during streaming. Some existing work (e.g., Anastasi et al.) have provided valuable guidelines for energy saving over WiFi transmissions; our work focuses on 3G cellular transmissions which have significantly different power models; 3G is a more practical wireless connection technology for mobile TVs on the go at the present time. We instead advocate non-layered coding in such delaysensitive mobile applications, although the detailed transcoding algorithm designs are out of the scope of this work. In addition, we novelly employ a surrogate for each mobile user in the cloud rather than relying on a dedicated cluster, which can be more easily implemented in practice. Liu et al.build a mobile-based social interaction framework on top of
the Google App Engine and offer an iOS implementation. We set out to design a portable, generic, and robust framework to enable realtime streaming and social interaction concurrently, which is not bound to any specific cloud platform. Although our prototype is implemented on only two public clouds, i.e., Amazon EC2 and Google App Engine, it can be easily ported to other cloud systems as long as the targeted cloud platforms conform to the unified standard. Finally, we are aware of the lack of a richly-featured cloudbased mobile social TV system in real life. The only system coming close to ours is Live Stream [22] on the iOS platform.
This iOS-locked application only supports live video channels, and all its social functions are bound to Facebook open APIs. Conversely, the prototype we implement is browser-based and platform independent; it supports both live channels, VoD channels and even personal channels hosted by any user, with wider usage ranges and flexible extensibility. The framework
we propose can be readily applied to other cloud-assisted mobile media applications as well.

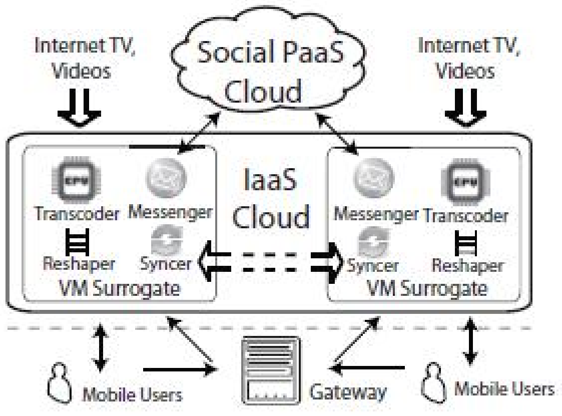

Fig. 1. The architecture of CloudMoV

III ARCHITECTURE AND DESIGN

As a novel Cloud-based Mobile sOcial tV system, CloudMoV provides two major functionalities to participating mobile users: (1) Universal streaming. A user can stream a live or on-demand video from any video sources he chooses, such as a TV program provider or an Internet video streaming site, with tailored encoding formats and rates for the device each time. (2) Co-viewing with social exchanges. A user can invite multiple friends to watch the same video, and exchange text messages while watching. The group of friends watching the same video is referred to as a session. The mobile user who initiates a session is the host of the session. We present the architecture of CloudMoV and the detailed designs of the different modules in the following.

Key Modules

Fig. 1 gives an overview of the architecture of CloudMoV. A surrogate (i.e., a virtual machine (VM) instance), or a VM surrogate equivalently, is created for each online mobile user in an IaaS cloud infrastructure. The surrogate acts as a proxy between the mobile device and the video sources, providing transcoding services as well as segmenting the streaming traffic for burst transmission to the user.

Transcoder. It resides in each surrogate, and is responsible for dynamically deciding how to encode the video stream from the video source in the appropriate format, dimension, and bit rate.

Reshaper. The reshaper in each surrogate receives the encoded transport stream from the transcoder, chops it into segments, and then sends each segment in a burst to the mobile device upon its request (i.e., a burst transmission mechanism), to achieve the best power efficiency of the device.

Social Cloud. The social cloud is built on top of any general PaaS cloud services with BigTable-like data store to yield better economies of scale without being locked down to any specific proprietary platforms. Despite its implementation on Google App Engine (GAE) as a proof of concept, our prototype can be readily ported to other platforms.

Messenger. It is the client side of the social cloud, residing in each surrogate in the IaaS cloud. The Messenger periodically queries the social cloud for the social data on behalf of the mobile user and pre-processes the data into a light-weighted format (plain text files), at a much lower frequency.

Syncer. The syncer on a surrogate guarantees that viewing progress of this user is within a time window of
other users in the same session (if the user chooses to
synchronize with others).
Mobile Client. The mobile client is not required to install any specific client software in order to use CloudMoV, as long as it has an HTML5 compatible browser (e.g.,Mobile Safari, Chrome, etc.) and supports the HTTP Live Streaming protocol. Both are widely supported on most state-of-the-art smartphones. Loosely Coupled Interfaces Similar in spirit to web services, the interfaces between different modules in CloudMov, i.e., mobile users, VM surrogates, and the social cloud, are based on HTTP, a universal standard for all Internet-connected devices or platforms.
Thanks to the loose coupling between users and the infrastructure. Pipelined Video Processing Both live streaming of realtime contents and on-demand streaming of stored contents are supported in CloudMoV. Video processing in each surrogate is designed to work on the fly, i.e., the transcoder conducts realtime encoding from the video source, the encoded video is fed immediately into the reshaper for segmentation and transmission, and a mobile user can start viewing the video as soon as the first segment is received.
Burst Transmissions
1) 3G power states: Different from Wi-Fi which is more
similar to the LANed Internet access, 3G cellular services suffer from the limited radio resources, and therefore each user equipment (UE) needs to be regulated by a Radio Resource Control (RRC) state machine [25]. Different 3G carriers may customize and deploy complex states in their respective cellular networks.
Transmission mechanism: In CloudMoV, we aim at
maximum conservation of the battery capacity of the mobile device, and design a burst transmission mechanism for streaming between the surrogate and the device.
Burst size: To decide the burst size, i.e., the size of
the segment transmitted in one burst, we need to take into consideration characteristics of mobile streaming and energy consumption during state transitions.
We consider two cases: (1) transmission of a video
according to our burst transmission mechanism, with Pburst(t) being the power level at time t during the transmission; (2) continuous transmission of the video stream whenever there are transcoded contents ready, with Pcont(t) being the power level at time t during the transmission. An illustration of power consumption in both cases is given in Fig. The burst transmission operates at state CELL DCH to send a total amount of data S for a duration of SB ; then it transits to state IDLE via state CELL FACH, and remains there for duration Sb
− SB −tDCH!FACH−tFACH!IDLE (Sb is the time taken for the mobile user to play the segment of size S). Note that the power consumption level during transition periods tDCH!FACH and tFACH!IDLE, remains at PDCH and PFACH, respectively, although no data is transmitted then. The
continuous transmission always operates at the high-power state CELL DCH with power level Pcont(t) = PDCH. We calculate the overall energy saving (_E) by burst transmission of the video over the time span T (multiples of Sb), as compared to the continuous transmission, as follows:

$$\begin{aligned}
\Delta E &= \int_0^T (P_{cont}(t) - P_{burst}(t))dt \\
&= \int_0^{\frac{S}{b}} (P_{cont}(t) - P_{burst}(t))dt \times \frac{T}{S/b} \\
&= \frac{T\times b}{S} \int_0^{\frac{S}{b}-\frac{S}{B}} (P_{cont}(t) - P_{burst}(t))dt \\
&= \frac{T\times b}{S} \times ((P_{DCH} - P_{FACH}) \times t_{FACH\to IDLE} \\
&+(P_{DCH} - P_{IDLE}) \times (\tfrac{S}{b} - \tfrac{S}{B} - t_{FACH\to IDLE} \\
&-t_{DCH\to FACH}) - P_{IDLE\to FACH} - P_{FACH\to DCH}).
\end{aligned} \tag{1}$$

IV. CLOUDMOV: PROTOTYPE IMPLEMENTATION

Following the design guidelines in Sec. III, we have implemented a real-world mobile social TV system, and deployed it on the Google App Engine (GAE) and Amazon EC2 clouds, which are the two most widely used public PaaS and IaaS cloud platforms. GAE, as a PaaS cloud, provides rich services on top of Google's data centers and enables rapid deployment of Javabased and Python-based applications. Data store, a thin layer built on top of Google's famous BigTable [26], handles "big" data queries well with linear and modular scalability even for high-throughput usage scenarios. Hence, GAE is an ideal platform for implementing our social cloud, which dynamically handles large volumes of messages. On the other hand,
GAE imposes many constraints on application deployment, e.g., lack of support for multi-threading, file storage, etc., which may hinder both computation-intensive jobs and content distribution applications.

A. Client Use of CloudMov

All mobile devices installed with HTML5 compatible
browsers can use CloudMoV services, as long as the HTTP Live Streaming (HLS) [24] protocol is supported. The user first connects to the login page of CloudMoV, as illustrated in the top left corner of Fig. 3. After the user successfully logs in through the gateway, he is assigned a VM surrogate from
the VM pool (the hostnames of available VMs, e.g., ec2-50-16-xx-xx.compute-1.amazonaws.com, are maintained in an inmemory table of a MySQL database deployed in the gateway).

B. VM Surrogates

All the VM surrogates are provisioned from Amazon EC2 web services and tracked by the gateway. We create our own AMI (ami-b6f220df) based on Linux kernel 2.6.35.14, the default image Amazon provides .

C. Data Models in the Social Cloud

We use GAE mainly as the back-end data store to keep the transient states and data of CloudMoV, including users' online presence status, social messages (invitation and chat messages) in all the sessions. With Jetty as the underlying Servlet container, most Java-based applications can be easily migrated to GAE, under limited usage constraints, where no platform-specific APIs are enforced for the deployment. GAE provides both its Java Persistence API (JPA 1.0, part of JSR 220) adapter and a set of proprietary low-level APIs to map the relational data. We choose to use the former only in CloudMoV.

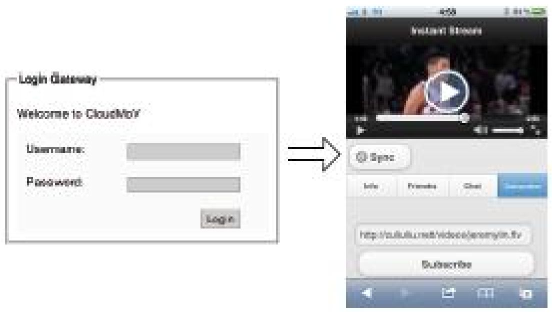

Fig. 2. Client UI of CloudMoV

Measuring the RRC States We first design measurement experiments to discover the timeout values of the critical inactivity timers employed in 3HK's 3G network, as discussed in Sec. III-D. We enable logging functions on an fully charged iPhone 4S and use the Mobile Safari (the HTML5-compatible browser on iPhone) to watch a YouTube video using CloudMoV services. The battery consumption traces on the phone are profiled by "Instruments", a powerful tool of Xcode . The playback rate of the video on the phone is about 254 Kbps.

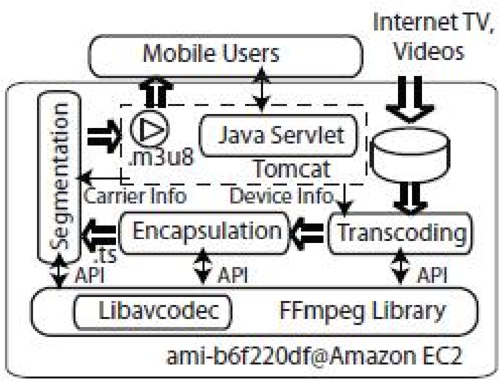

Fig. 3. Streaming architecture in each customized VM image (ami-b6f220df).

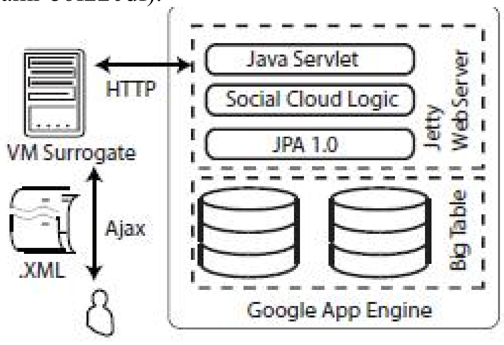

Fig. 5. Social message exchanges via Google App Engine.

Impact of Burst Size on Power Consumption The technique of video segmentation is widely employed in video streaming applications, but mostly for ease of distribution and not for battery efficiency at potential mobile users. Apple Inc., which proposed the HTTP Live Streaming protocol , suggests 10-second-playback segments, which has been followed in many streaming applications. We find this segment size is problematic and can drain the battery of a mobile device quickly. In Fig. 8, we compare the power consumption levels when burst transmission intervals of 10seconds and 60 seconds are used, respectively, for the iPhone 4S to stream a 10-minute YouTube flash video (.flv). We note
that, iOS devices can not play flash videos, but CloudMoV helps transcode the flash to the H264/AAC stream, which is compatible with our iPhone 4S.

Sign-in Latency into the System When a user signs into the CloudMoV system via the login gateway shown in Fig. 3 and gets identified, the gateway will request a virtual machine instance from the IaaS cloud to be the user's surrogate. The sign-in process finishes when
the surrogate is initialized and the user is connected to the surrogate. In this experiment, five mobile users repeatedly join the system and log off as soon as the respective surrogate is initialized. We inject JavaScript snippets into the client of CloudMov on the mobile device to record the timestamps during the sign-in process. Fig. 9 shows the average signin latencies experienced by these clients during a 4.5-hour
span. The "Front-end" latency consists of both the sign-in request/response and identification delays, while the "Backend" latency is the surrogate VM provisioning delay from the IaaS cloud (Amazon EC2). We can see that most of the latencies are caused by the latter. The delay can be significantly reduced if a VM pool is maintained wherein idlenb surrogates are initialized before hand (based on estimated user numbers), ready for immediate allocation when new users sign in.

Startup Latency of Video Playback

We evaluate the transcoding performance on the surrogates vin CloudMoV, first by measuring the playback startup latency on the surrogates, from the time when the video subscription request is received from the mobile user to the time when the first transcoded burst segment is generated. We deploy the VM surrogates (ami-b6f220df) on three types of instances provided by Amazon EC2, with the detailed configurations shown in Table. I. For fair comparison, all the instances are deployed in the zone "east-1-c", and they transcode the same flash video used in experiments of Sec. V-B. Fig. 10 shows the playback
startup latencies when different VM instances are used as theb surrogate for an iPhone 4S, and different burst transmission intervals are employed.

E. Dynamic Bit Rate Switch

We next evaluate whether CloudMoV can effectively
transcode a video stream to different bit rates when the
long movie produced by Pixar, in the original bit rate of 1017 Kbps and .avi format with the XviD codec. The movie is stored on an Apache web server isolated from the CloudMoV system. The format of the movie can not be directly played on an iPhone. CloudMoV dynamically transcodes this movie into two H264/AAC streams of different bit rates: a high-quality stream with a bit rate up to 515 Kbps and a low-quality stream at 261 Kbps. When the phone's wireless connection bandwidth is lower than 900 Kbps, CloudMoV directs the low-quality stream to it; otherwise, it transmits the high-quality stream. Jitters By "Jitters", we mean the discontinuous video playback experienced by mobile users who have to wait for segmentsnto be buffered, due to the dynamically varying download bandwidths. Following the same experiment settings as in Sec. V-E, we emulate a highly unstable 3G cellular network

and measure the occurrence and stall duration of jitters when a mobile client is viewing the movie. We examine the download completion time for each segment: if this time is later than the playback deadline of the segment, a jitter is captured and the stall duration is estimated as the difference between the two. Fig. compares the results of CloudMoV and the case where the movie is directly streamed to the mobile user without dynamic transcoding nor burst transmission mechanisms, i.e., the case of "Normal Streaming".

Social Interaction Latencies The service latency of Google App Engine is critical to the overall performance of CloudMoV. In this set of experiments,we launch a VM surrogate in each of four different regions (corresponding to four mobile users), i.e., "east-1-a", "east- 1-b", "east-1-c" and "east-1-d", all of which join the same session. Each surrogate keeps posting a short chat message every second and retrieves its own message immediately.

We evaluate two critical latencies: one is the post latency to the GAE, i.e., the time from when a message is sent out from a surrogate to the time when it receives confirmation from GAE that the message is successfully recorded in the social cloud; the other is the query latency, i.e., the time from when a query is sent out from a surrogate to the time when the queried message is received at the surrogate.

Scalability,To evaluate the scalability of CloudMoV, we investigate the workload at the host of a session. As compared to a regular participant in a session, the surrogate of a session host is additionally responsible for maintaining the session group and carrying out synchronization for "co-viewing" experiences, besides its own transcoding tasks, which may potentially become a performance bottleneck in the system when the number of participants in the session is large.

TABLE I
CONFIGURATIONS OF VM INSTANCES

Type	CPU	Memory	Zone
Micro	Up to 2 ECUs	613MB	east-1-c
Small	1 ECU	1.7 GB	east-1-c
Medium	5 ECUs (2 cores)	1.7 GB	east-1-c

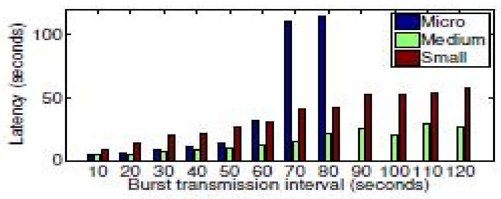

Fig. 5. Startup latency at different burst sizes.

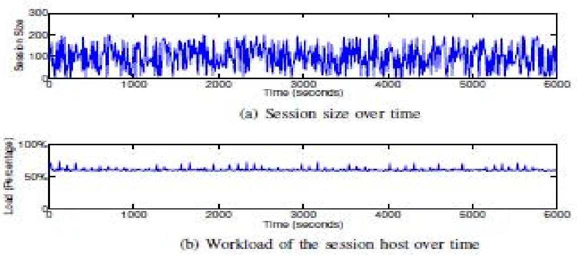

Fig. 6. Session size and the workload on the surrogate of the session host over time.

V.CONCLUDING AND FUTURE WORK

This paper presents our view of what might become a trend for mobile TV, i.e., mobile social TV based on agile resource supports and rich functionalities of cloud computing services. We introduce a generic and portable mobile socialn TV framework, CloudMoV, that makes use of both an IaaS cloud and a PaaS cloud. The framework provides efficient transcoding services for most platforms under various network conditions and supports for co-viewing experiences through timely chat exchanges among the viewing users. By employing one surrogate VM for each mobile user, we achieve ultimate scalability of the system. Through an in-depth investigation of the power states in commercial 3G cellular networks, we then propose an energy-efficient burst transmission mechanism that can effectively increase the battery lifetime of user devices. We have implemented a realistic prototype of CloudMoV, deployed on Amazon EC2 and Google App Engine, where EC2 instances serve as the mobile users' surrogates and GAE as the social cloud to handle the large volumes of social message exchanges. We conducted carefully designed experiments on iPhone 4S platforms. The experimental results prove the superior performance of CloudMoV, in terms of transcoding efficiency, power saving, timely social interaction, and scalability. The experiments also highlight the drawbacks of the current HTTP Live Streaming protocol implementation on mobile devices as compared to our proposed burst transmission mechanism which achieves a 29.1 % increase of battery lifetime. conditions and supports for co-viewing experiences throughtimely chat exchanges among the viewing users. By employing one surrogate VM for each mobile user, we achieve ultimate scalability of the system. Through an in-depth investigation of the power states in commercial 3G cellular networks, we thenpropose an energy-efficient burst transmission mechanism that can effectively increase the battery lifetime of user devices.We have implemented a realistic prototype of CloudMoV, deployed on Amazon EC2 and Google App Engine, where
EC2 instances serve as the mobile users' surrogates and GAE as the social cloud to handle the large volumes of social message exchanges. We conducted carefully designed experiments on iPhone 4S platforms. The experimental results prove the superior performance of CloudMoV, in terms of transcoding efficiency, power saving, timely social interaction and scalability.

REFERENCES

[1] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case forvm-based cloudlets in mobile computing," IEEE Pervasive Computing, vol. 8, pp. 14–23, 2009.

[2] S. Kosta, A. Aucinas, P. Hui, R. Mortier, and X. Zhang, "Thinkair: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading," in Proc. of IEEE INFOCOM, 2012.

[3] Z. Huang, C. Mei, L. E. Li, and T. Woo, "Cloudstream: Delivering high-quality streaming videos through a cloud-based svc proxy," in INFOCOM'11, 2011, pp. 201–205.

[4] T. Coppens, L. Trappeniners, and M. Godon, "AmigoTV: towards asocial TV experience," in Proc. of EuroITV, 2004.

[5] N. Ducheneaut, R. J. Moore, L. Oehlberg, J. D. Thornton, and E. Nickell, "Social TV: Designing for Distributed, Sociable Television Viewing," International Journal of

Human-Computer Interaction, vol. 24, no. 2,pp. 136–154, 2008.

[6] A. Carroll and G. Heiser, "An analysis of power consumption in as smartphone," in Proc. of USENIXATC, 2010.

[7] What is 100% Pure Java, http://www.javacoffeebreak.com/faq/faq0006.html.

[8] J. Santos, D. Gomes, S. Sargento, R. L. Aguiar, N. Baker, M. Zafar, and A. Ikram, "Multicast/broadcast network convergence in next generation mobile networks," Comput. Netw., vol. 52, pp. 228–247, January 2008.

[9] DVB-H, http://www.dvb-h.org/.

[10] K. Chorianopoulos and G. Lekakos, "Introduction to social tv: Enhancing the shared experience with interactive tv," International Journal of Human- Computer Interaction, vol. 24, no. 2, pp. 113–120, 2008.

[11] M. Chuah, "Reality instant messaging: injecting a dose of reality into online chat," in CHI '03 extended abstracts on Human factors in computing systems, ser. CHI EA '03, 2003, pp. 926–927.

[12] R. Schatz, S. Wagner, S. Egger, and N. Jordan, "Mobile TV becomes Social - Integrating Content with Communications," in Proc. of ITI, 2007.

K. AnandBabu did B.Tech CSE and PersuingM.Tech CS at Narayana Engineering College (NECG), Gudur, AP, and doing project on Improving High Energy Efficiency and Streaming Quality in Cloud Based Mobile Systems.

J.Venkata Prasad M.Tech Associate professor Department of Computer Science and Engineering Narayana Engineering College Gudur. jvprasad2003@gmail.com .