IJREAT International Journal of Research in Engineering & Advanced Technology, Volume 2, Issue 2, Apr-May, 2014
ISSN: 2320 – 8791 (Impact Factor: 1.479)
www.ijreat.org

Live T.V Experience in Smart Phones with Cloud using Video Transcoding and Surrogate

Pradeep B.M[1], Venkatravanna Nayak K[2], Krishna Gudi[3]

[1, 2, 3]**GSS Institute of Technology, Kengeri, Bangalore–560064, Karnataka**

Abstract

The rapidly increasing power of personal mobile devices such as smart phones, tablets, etc. is providing much richer contents and social interactions to users on the move. This trend however is throttled by the limited battery life of mobile devices and unstable wireless connectivity, achieving 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. Critical 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, Mobile Smart TV. This system effectively utilizes both IaaS (Infrastructure-as-a-Service) and PaaS (Platform-as-a-Service) cloud delivery services to offer the living-room experience of video watching to a group of disparate mobile users who can interact socially while watching and sharing the video. To guarantee good streaming quality as experienced by the mobile users with time-varying 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 smart TV services

Keywords: Computers and information processing, Mobile computing, Communications technology, TV, Mobile TV.

1. Introduction

Thanks to the revolutionary "reinventing the phone" Campaigns initiated by Apple Inc. in 2007, Smartphones and Tablets 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 strength in more challenging scenarios such as real time video streaming and online gaming, as well as serving as a main tool for social exchanges. 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. Tough challenge in front of us is how to effectively exploit cloud services to facilitate mobile applications. In this paper, we describe the design of a novel mobile social TV system, "Smart TV", 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 Smart TV, 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 watching the video. As opposed to traditional TV watching, mobile social TV is well suited to today's Traditional life style, where family and friends may be distributed geographically but hope to share a co-viewing experience. We design Smart TV to seamlessly utilize agile resource support and rich functionalities offered by both an IaaS (Infrastructure- as-a-Service) cloud and a PaaS (Platform-as-a-Service) cloud. Our design achieves the following goals.

1.1 Encoding Flexibility

Different mobile devices have various sized displays, customized playback hardwares, and various codecs. Traditional solutions would adopt a few encoding formats ahead of the release of a video program. But even the most generous content providers would not be able to attend to all possible mobile platforms, if not only to the current hottest models. Smart TV customizes the streams for different devices at real time, by offloading the transcoding tasks to an IaaS cloud. In particular, we novelly employ a surrogate to each user, which is a virtual machine (VM) in the IaaS cloud. The surrogate downloads the video on behalf of the user and transcodes it into the desired formats, while catering to the specific configurations of the mobile device as well as the current connectivity quality.

1.2. Battery Efficiency

A breakdown analysis conducted by Carroll and G. Heiser [6] indicates that the network modules (such as 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. We target at energy saving coming from the network module of smart phones through an efficient data transmission mechanism design. We focus on 3G wireless networking as it is getting more widely used and challenging in our design than Wi-Fi based transmissions. Based on cellular network traces from real-world 3G carriers, we investigate the key 3G configuration parameters such as the power states and the inactivity timers, and design a novel burst transmission mechanism for streaming from the surrogates to them mobile devices

1.3. Spontaneous Social Interactivity

Multiple mechanisms are included in the design of Smart TV to enable spontaneous social interactivity, co-viewing experience. First, efficient synchronization mechanisms are proposed to guarantee that friends joining in a video program may watch the same portion (if they choose to), and share immediate reactions and comments. Although synchronized playback is inherently a feature of traditional TV, the current Internet video services (e.g., Web 2.0 TV) rarely offer such a service. Second, efficient message communication mechanisms are designed for social textual interactions among friends, and different types of messages are prioritized in their retrieval frequencies to avoid unnecessary interruptions of the viewing progress. For example, online friend lists can be retrieved at longer intervals at each user, while invitation and chat messages should be delivered more timely. We adopt textual chat messages rather than voice in our current design, believing that text chats are less distractive to viewers and easier to write/read and manage by any user.

1.4. Portability

A prototype Smart TV system is implemented following the philosophy of “Write Once, Run Anywhere” (WORA) “100% Pure Java” [4] platform used to implement both the front-end mobile modules and the back-end server modules, with well-designed generic data models suitable for any Big Table-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. The client module can run on any mobile devices supporting HTML5, including Android phones, IOS systems, etc. To showcase its performance, we deploy the system on Amazon EC2 and Google App Engine, and conduct thorough tests on IOS platforms.

2. RELATED WORK

A huge amount of mobile TV systems have sprung up in recent years, driven by both hardware and software advances in mobile devices. Some early systems bring the “living-room” experience to small screens on the move. But they focus more on barrier clearance in order to realize the convergence of the mobile network and the television network, rather than exploring the demand of “social interactions” among mobile users. Coppens et al. [4] try to add rich social interactions to TV but their design is limited to traditional broadcast program channels. Though inspiring, these designs are not that suitable for being applied directly in a mobile environment. Chuah et al. [6] extend the social experiences of viewing traditional broadcast programs to mobile devices, but have yet to deliver a well integrated framework. Schatz et al. [7], [8] have designed a mobile social TV system, which is customized for DVB-H networks and Symbian devices as opposed to a wider audience. Compared to these prior work and systems, we target at a design for a generic, portable mobile social TV framework, featuring “co-viewing experiences” among friends over geographical separations through mobile devices. Our framework is open to all Internet-based video programs, either live or on-demand, and supports a wide range of devices with HTML5 compatible browsers installed, without any other mandatory component on the devices. For any application targeted at mobile devices, consumes less power is perennially one of the major concerns and challenges. Our work is able to achieve a significant (about 30%) power saving, by opportunistically switching the device between low-power and high-power transmission modes during streaming. Some existing work (e.g., Anastasi et al. [9]) 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. Cloud computing had its debut with much fanfare and is now deemed a most powerful hosting platform in many areas including mobile computing. Satyanarayan an et al. [1] suggest offloading mobile devices computation workload to a nearby resource-rich infrastructure (i.e., Cloudlets) by dynamic VM synthesis. Kosta et al. [2] propose a virtualization framework for mobile code offloading to the cloud. Zhang et al. [10] introduce an elastic mobile application model by offloading part of the applications (weblets) to an IaaS cloud. Recently, attentions have been drawn to enabling media applications using the cloud, for both media storage [11] and processing [12] Conversely, the prototype we implement is browser-based and platform independent; it supports both live channels, VoD channels and personal channels hosted by any user, with wider usage ranges and flexible extensibility.

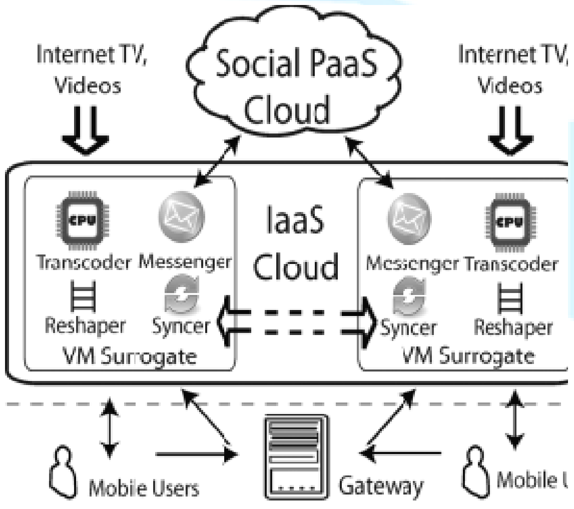

Fig. 1. The architecture of Smart T.V.

3. Smart T.V: Architecture and Design

As a novel Cloud-based Mobile Smart TV system, Smart T.V 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 Smart T.V and the detailed designs of the different modules in the following.

3.1. Key Modules

Fig. 1 gives an overview of the architecture of Smart T.V. 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. The surrogates exchange social messages via a back-end PaaS cloud, which adds scalability and robustness to the system. There is a gateway server in Smart T.V that keeps track of participating users and their VM surrogates, which can be implemented by a standalone server or VMs in the IaaS cloud. The design of Smart T.V can be divided into the following major functional modules.

• Transcoder: It resides in each surrogate, and is responsible for dynamically deciding how to encode the video stream from the video source in the recommended format, dimension, and bit rate. Before deliver to the user, the video stream is further encapsulated into a proper transport stream. In our implementation, each video is exported as MPEG-2 transport streams, which is the de-facto standard nowadays to deliver digital video and audio streams over lossy medium.

• 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. The burst size, i.e., the amount of data in each burst, is carefully decided according to the 3G technologies implemented by the corresponding carrier.

• 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 Function prototype can be readily ported to other platforms. It stores all the social data in the system, including the online statuses of all users, records of the existing sessions, and messages (invitations and chat histories) in each session. The social data are categorized into different kinds and split into different entities (in analogy to tables and rows in traditional relational database, respectively). The social cloud is queried from time to time by the VM surrogates.

• Messenger: The messenger is the client part 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. The plain text files (in XML formats) are asynchronously delivered from the surrogate to the user in a traffic-friendly manner, i.e., little traffic is incurred. In the reverse direction, the messenger disseminates this user's messages (invitations and chat messages) to other users via the data store of the social cloud.

• 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). To achieve this, the syncer periodically retrieves the current playback progress of the session host and instructs its mobile user to adjust its playback position. In this way, friends can enjoy the "sitting together" viewing experience. Different from the design of communication among messengers.

• Mobile Client: The mobile client is not necessary to install any specific client software in order to use Smart T.V, as long as it has an HTML5 compatible browser (e.g., Mobile Safari, Chrome, etc.) and supports the HTTP Live Streaming protocol [13]. Both are widely supported on most state-of-the-art smart phones.

• Gateway: The gateway provides authentication services for users to log in to the Smart T.V system, and stores users credentials in a permanent table of a MySQL database it has installed. It also stores information of the pool of currently available VMs in the IaaS cloud in another in-memory table. After a user successfully logs in to the system, a VM surrogate will be assigned from the pool to the user. The in-memory table is used to guarantee small query latencies, since the VM pool is updated frequently as the gateway reserves and destroys VM instances according to the current workload. We describe the key designs in Smart T.V in the following.

3.2. Loosely Coupled Interfaces

Similar in spirit to web services, the interfaces between different modules in Smart T.V, 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, almost any mobile device is ready to gain access to the Smart T.V services, as long as it is installed with an HTTP5 browser. The VM surrogates provisioned in the IaaS cloud co-operate with the social cloud implemented on a PaaS cloud service via HTTP as well, with no knowledge of the inner components and underlying technologies of each other, which contributes significantly to the portability and easy maintenance of the system. For social message exchanges among friends, Smart T.V employs asynchronous communication.

3.3. Pipelined Video Processing

Both live streaming of real time contents and on-demand streaming of stored contents are supported in Smart T.V. Video processing in each surrogate is designed to work on the fly, i.e., the transcoder conducts real time 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. To support dynamic bit rate switch, the transcoder launches multiple threads to transcode the video into multiple bit rates once the connection speed between the surrogate and the mobile user changes. The IaaS represents an ideal platform for implementing such computation intensive jobs.

3.4. Burst Transmissions

3.4.1 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. Different 3G carriers may customize and deploy complex states in their respective cellular networks. Different states indicate different levels of allocated radio resources, and hence different levels of energy consumptions. For ease of implementation, we consider three basic states in our design, which are commonly employed by many carriers, namely CELL_DCH (a dedicated physical channel is allocated to the UE in both the uplink and the downlink), CELL_FACH (no dedicated channel is allocated but the UE is assigned a default common transport channel in the uplink), and IDLE, in decreasing order of power levels. Contrary to intuition, the energy consumption for data transmission depends largely on the state a UE is working in, but has little to do with the volume of data transmitted, i.e., a UE may stay at a high-power state (CELL_DCH) for data transmission even the data rate is very low (this has also been verified in our experiments in Section V). For example, if a UE working at a high-power state does not incur any data traffic for a pre-configured period of time (measured by a critical inactivity timer), the state of the UE will be transferred to a low-power one; when the volume of data traffic rises, the UE "wakes up" from a low-power state and moves to a high-power one.

3.4.2 Transmission Mechanism

In Smart T.V, 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. Using the HTTP live streaming protocol [13], the mobile device sends out requests for the next segment of the video stream from time to time. The surrogate divides the video into segments, and sends each segment in a burst transmission to the mobile device, upon such a request. When the mobile device is receiving a segment, it operates in the high-power state (CELL_DCH); when there is nothing to receive, it transfers to the low-power state (IDLE) via the intermediate state (CELL_FACH), and remains there until the next burst (segment) arrives.

3.4.3 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. For video streaming using a fixed device without power concerns, it is desirable to download as much of a video as what the connection bandwidth allows; however, for streaming over a cellular network, leading to a waste of the battery power and the cellular data fee due to their download. Hence, the bursty size should be kept small, to minimize battery consumption and traffic charges. We next derive a lower bound on the burst size, which guarantees positive energy saving by such intelligent state transition as compared to the continuous transmission, as follows:

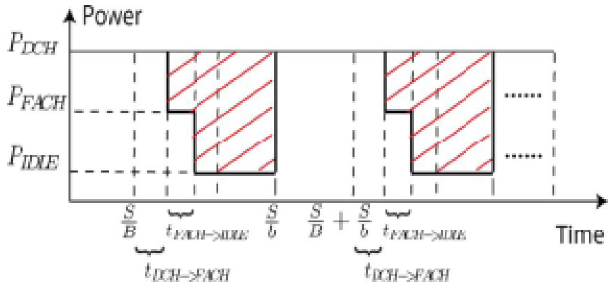

Fig. 2: Power consumption over time.

4. Live T.V: Prototype Implementation

Following the design guidelines in Section 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 Java-based and Python-based applications. 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. Amazon EC2 is a representative IaaS cloud, offering raw hardware resources including CPU, storage, and networks to users. Comparing to a common PaaS cloud, EC2 is an appropriate platform for computation-intensive tasks in Smart T.V.

4.1. Client Use of Smart T.V

All mobile devices installed with HTML5 compatible browsers can use Smart T.V services, as long as the HTTP Live Streaming (HLS) [13] protocol is supported. The user first connects to the login page of Smart T.V, 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 in-memory table of a MySQL database deployed in the gateway). Then the user is automatically redirected to the assigned VM surrogate, and welcomed by a portal page as shown on the right-hand side of Fig. 3. Upon user login, the portal collects the device configuration information by examining the "User-Agent" header values, and this information will be sent to its surrogate for decision making of the video encoding formats. The user can enter the URL of the video or live broadcast he wishes to watch, on the "Subscribe" tab of the portal; after he clicks the "Subscribe" button, the address of the video is sent to the VM surrogate, which downloads the stream on the user's behalf, transcodes and sends properly encoded segments to the user. When watching a video, the user can check out his friends' status (online or offline, which video they are currently watching) on the "Friends" tab (a snapshot is given in Fig. 4(a)), and invite one or more friends to join him in watching the video.. Users in the same session can exchange opinions and comments on the "Chat" tab (a snapshot is given in Fig. 4(b)), where new chat messages can be entered and the chat history of the session is shown.

4.2. 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 [15]. Due to the intensive computation involved, we propose to implement all the video processing related tasks using ANSI C, to guarantee the performance. We have also installed a Tomcat web server (version 6.5) to serve as a Servlet container and a file server on each surrogate. Both FFmpeg and Tomcat are open source projects. Once a VM surrogate receives a video subscription request from the user, it downloads the video from the source URL, and processes the video stream by transcoding and segmentation, based on the collected device configurations by the portal. For example, in our experiments, the downloaded stream is transcoded into a high-quality stream from a low-quality stream depends on the device capability.

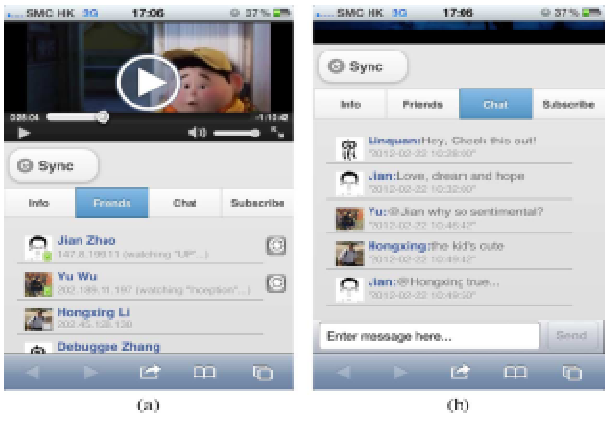

Fig. 4. "Friend" and "Chat" tabs. (a) "Friend tab" (b) "Chat tab".

The high-quality stream has a "480 272" resolution with 24 frames per second, while the low-quality one has a "240 136" resolution with 10 frames per second. Fig. 5 shows the streaming architecture in our customized VM image. Here, the modules on social message exchanges are omitted, which will be presented in Fig. 6.

4.3 Data Models in the Social Cloud

We use GAE mainly as the back-end data store to keep the transient states and data of Smart T.V, 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.

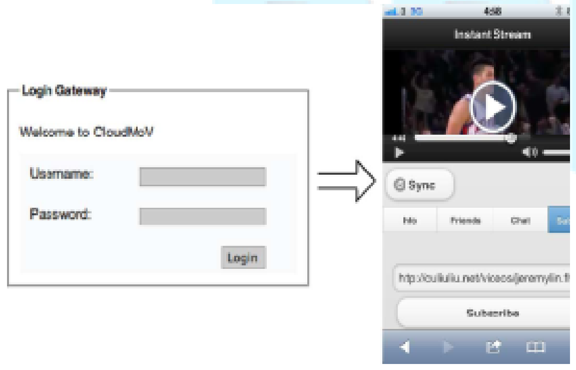

Fig. 3. Client UI of Smart T.V

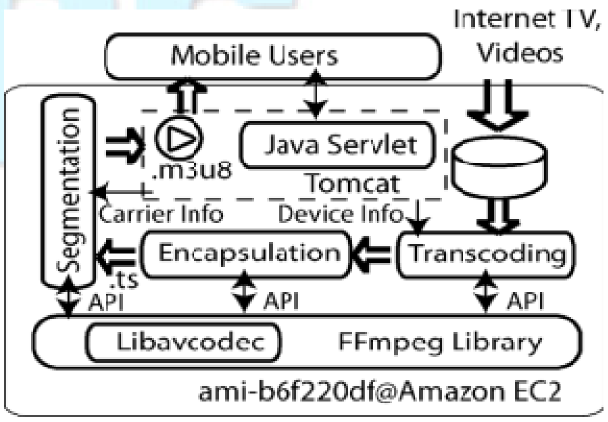

Fig. 5. Streaming architecture in each customized VM image (ami-b6f220df).

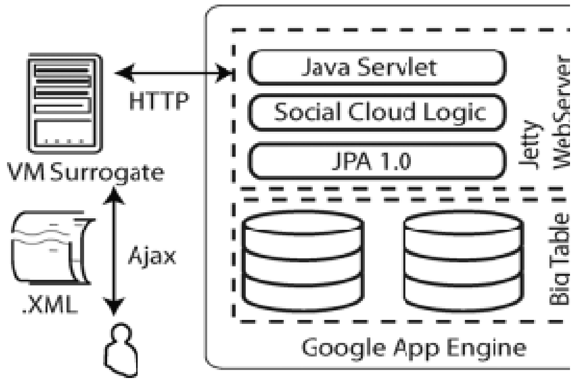

Fig. 6. Social message exchanges via Google App Engine.

5. REAL-WORLD EXPERIMENTS

We carry out both unit tests and performance evaluations of Smart T.V deployed on Amazon EC2 and Google App Engine. The experiments are conducted over the 3G cellular network of 3HK, which is one of the largest Telecom operators in Hong Kong. Fig. 7 shows the power consumption levels on the phone over time, in terms of portions of the highest device power level. The red vertical lines represent the starting points of playback periods when the Safari runs in the foreground, and the green lines represent the finish times of playback periods when the Safari is suspended in the background. We can see that our state transition model in Fig. 2 is verified by these real-world measurements: when there is data transmission, the device operates at the high power mode; when data transmission stops, the transmission power of the device first decreases to an intermediate level, and then to a very low level.

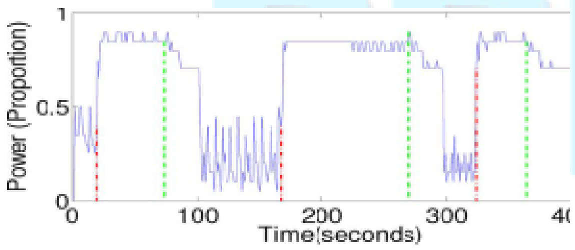

Fig. 7. Power consumption over time on an iPhone 4S.

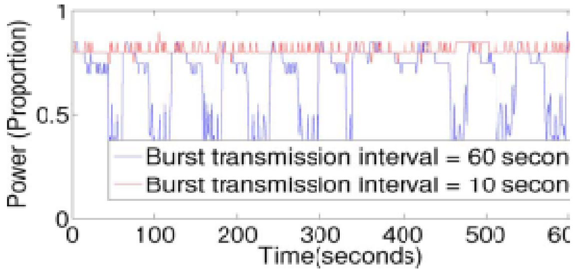

Fig. 8. Power consumption over time with different burst transmission sizes

5.1. Startup Latency of Video Playback

We evaluate the transcoding performance on the surrogates in Smart TV, 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. In our experiments, we tested the network connection bandwidth between the Amazon EC2 instances and the YouTube website, and found that video downloading from YouTube website to the instances is very fast. Fig 9 shows that in general, the longer the burst interval is, the larger the segment of video to transcode is, and thus the longer the startup latency.

5.2. Social Interaction Latencies

The service latency of Google App Engine is critical to theoverall performance of Smart TV. 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.

uding

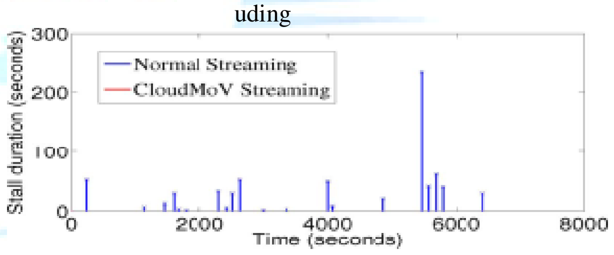

Fig. 9. Jitters and the stall durations

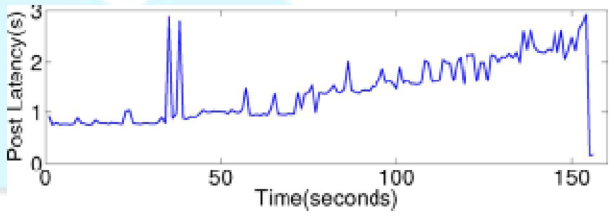

Fig. 10. Post latency to GAE.

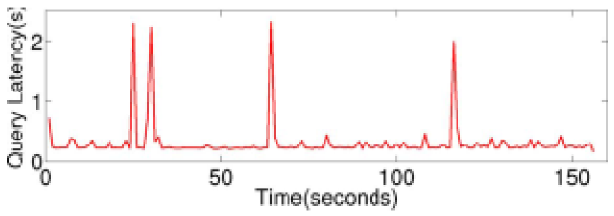

Fig. 11. Query latency from GAE.

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. Figs. 10 and 11 show the average values of the two types of latencies among surrogates in all regions, during a 155-second run of the experiments. Our results are mostly consistent with the 978-ms post latency and 106-ms query latency (our latencies additionally include a round-trip time between a surrogate and the GAE). Confirming the detailed reason is part of our future work.

6. Concluding Remarks and Future Enhancement

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 social TV framework, Smart T.V, 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 Smart T.V, 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. The experimental results prove the superior performance of Smart T.V, in terms of transcoding efficiency, power saving, timely social interaction, and scalability. Much more, however, can be done to enhance Smart T.V to have product-level performance. In the current prototype, we do not enable sharing of encoded streams (in the same format/bit rate) among surrogates of different users. In our future work, such sharing can be enabled and carried out in a peer-to-peer fashion, e.g., the surrogate of a newly joined user may fetch the transcoded streams directly from other surrogates, if they are encoded in the format/bit rate that the new user wants. For implementing social interactions, most BigTable-like data stores (including GAE) support memcache [17] which is a highly efficient secondary storage on the data stores.

References

[1] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for VM-based Cloudlets in mobile computing," IEEE Pervasive Comput.,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. IEEE INFOCOM, 2012.
[3] T. Coppens, L. Trappeniners, and M. Godon, "AmigoTV: Towards a social TV experience," in Proc. EuroITV, 2004.
[4] N. Ducheneaut, R. J. Moore, L. Oehlberg, J. D. Thornton, and E. Nickell, "Social TV: Designing for distributed, sociable television viewing," Int. J. Human-Comput. Interaction, vol. 24, no. 2, pp. 136–154, 2008.
[5] What is 100% Pure Java. [Online]. Available: ttp://www.javacoffeebreak.com/faq/faq0006.html.
[6] M. Chuah, "Reality instant messaging: Injecting a dose of reality into online chat," in CHI '03 Extended Abstracts on Human Factors in Computing Syst., 2003, ser. CHI EA '03, pp. 926–927.
[7] R. Schatz, S. Wagner, S. Egger, and N. Jordan, "Mobile TV becomes social – Integrating content with communications," in Proc. ITI, 2007.
[8] R. Schatz and S. Egger, "Social interaction features for mobile TV services," in Proc. 2008 IEEE Int. Symp. Broadband Multimedia Syst. And Broadcasting, 2008.
[9] G. Anastasi, M. Conti, E. Gregori, and A. Passarella, "Saving energy in Wi-Fi hotspots through 802.11 psm: An analytical model," in Proc Workshop Linguistic Theory and Grammar Implementation, ESSLLI2000, 2004, pp. 24–26.
[10] X. Zhang, A. Kunjithapatham, S. Jeong, and S. Gibbs, "Towards an elastic application model for augmenting the computing capabilities of mobile devices with cloud computing," Mobile Netw. Applicat., pp. 1–15, Apr. 2011.
[11] W. Zhu, C. Luo, J. Wang, and S. Li, "Multimedia cloud computing," IEEE Signal Process. Mag., vol. 28, pp. 59–69, 2011.
[12] R. Pereira and K. Breitman, "A cloud based architecture for improving video compression time efficiency: The split & merge approach," in Proc. DCC'11, 2011, pp. 471–471.
[13] HTTP Live Streaming. [Online]. Available: http://tools.ietf.org/html/draft-pantos-http-live-streaming-01.
[14] 3GPP TS 25.331. [Online]. Available: html-info/25331.htm.
[15] Amazon EC2. [Online]. Available: http://www.3gpp.org/ftp/Specs/http://aws.amazon.com/ec2/.
[16] FFmpeg. [Online]. Available: http://ffmpeg.org/.
[17]What is Memcached. [Online]. Available: http://memcached.org/.r.