IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661, p- ISSN: 2278-8727Volume 16, Issue 2, Ver. IX (Mar-Apr. 2014), PP 129-137
www.iosrjournals.org

Social Interaction Feature for Mobile TV Services Based On Cloud Move

[1]Sakitha V, [2]Sarasvathi V ME (Phd)
[1]Electronics and communication engineering ARJ college of engineering and technology Mannargudi, india
[2](associate professor) Electronics and communication engineering ARJ college of engineering and technology Mannargudi, india

Abstract: *The rapidly increasing power of personal mobile devices (smartphones, tablets, etc.) is providing much 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- as- a-Ser-vice) 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 time-varying wire-less 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 real-world experiments.*

Index Terms: *Computers and information processing, Mobile computing, Communications technology, TV, Mobile TV.*

I. Introduction

T HANKS 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 con-nection bandwidth are the most difficult ones. It is natural to resort to cloud computing, the newly-emerged computing par-adigm for low- cost, agile, scalable resource supply, to support power-efficient mobile data communication. With virtually infi- nite 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. As opposed to traditional TV watching, mobile social TV is well suited to

today's life style, where family and friends may be separated geographically but hope to share a co-viewing ex-perience. While social TV enabled by set -top boxes over the traditional TV systems is already available it remains a challenge to achieve mobile social TV, where the concurrently viewing experience with friends is enabled on mobile devices.

We design CloudMoV to seamlessly utilize agile resource support and rich functionalities offered by both an IaaS (In-frastructure- as-a -Service) cloud and a PaaS (Platform-as-a-Ser-vice) cloud. Our design achieves the following goals.

A. Encoding Flexibility

Different mobile devices have differently 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 platforms, if not only to the current hottest models. CloudMoV 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 for 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 for-mats, while catering to the specific configurations of the mobile device as well as the current connectivity quality.

B. Battery Efficiency

A breakdown analysis conducted by Carroll. 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. We target at energy saving coming from the network module of smartphones 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 transmis-sions. 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 the mobile devices. The burst transmission mechanism makes careful decisions on burst sizes and opportunistic transitions among high/low power consumption modes at the devices, in order to effectively increase the battery lifetime.

C. Spontaneous Social Interactivity

Multiple mechanisms are included in the design of CloudMoV to enable spontaneous social, co-viewing ex-perience. 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 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 read/write and manage by any user.

These mechanisms are seamlessly integrated with function-alities provided by a typical PaaS cloud, via an efficient design of data storage with BigTable and dynamic handling of large volumes of concurrent messages. We exploit a PaaS cloud for social interaction support due to its provision of robust under-lying platforms (other than simply hardware resources provided by an IaaS cloud), with transparent, automatic scaling of users' applications onto the cloud.

D. Portability

A prototype CloudMov system is implemented following the philosophy of "Write Once, Run Anywhere" (WORA): both the front-end mobile modules and the back-end server mod-ules are implemented in "100% Pure Java" with well-de-signed generic data models suitable for any BigTable-like data store; the only exception is the transcoding module, which is im-plemented 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 An-droid 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. Our prototype can be readily migrated to various cloud and mobile platforms with little effort.

The remainder of this paper is organized as follows. In Section II, we compare our work with the existing literature and highlight our novelties. In Section II, we present the architecture of CloudMoV and the design of individual mod-ules. A real -world prototype implementation follows and is described in Section III,

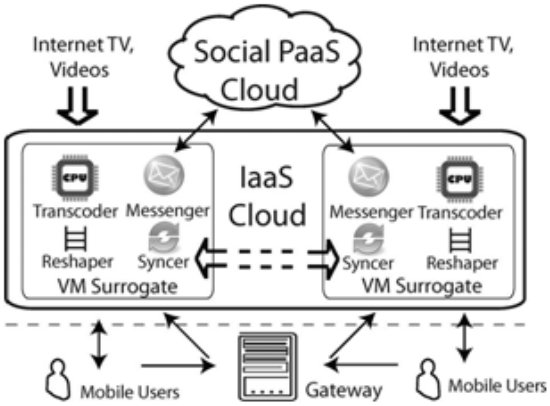

Fig. 1. The architecture of CloudMoV.

II. Cloudmov: 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 -de-mand video from any video sources he chooses, such as a TV program provider or an Internet video streaming site, with tai-lored 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 ses-sion is the host of the session. We present the architecture of CloudMoV and the detailed designs in the following.

A. 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,iscreatedforeachonlinemobileuserinan 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. Besides, they are also responsible for handling frequently exchanged social messages among their corresponding users in a timely and efficient manner, shielding mobile devices from unnecessary traffic and enabling battery efficient, spontaneous social interactions. 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 CloudMoV 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 CloudMoV can be divided into the following major functional modules.

Transcoder. It resides in each surrogate, and is respon-sible for dynamically deciding how to encode the video stream from the video source in the appropriate format, dimension, and bit rate. Before delivery to the user, the video stream is further encapsulated into a proper trans-port stream. In our implementation, each video is exported as MPEG-2 tansport streams, which is the de facto standard nowadays
to deliver digital video.

Reshaper. The reshaper in each surrogate receives the en- coded 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 de vice. The burst size, i.e., the amount of data in each burst, is carefully decided according to the 3G technologies im-plemented by the corresponding carrier.

Social Cloud. The social cloud is built on top of any gen- eral PaaS cloud services with BigTable-like data store toyield better economies of scale without being locked down to any specific proprietary platforms. Despite its imple- mentation on Google App Engine (GAE) as a proof of con- cept, our prototype can be readily ported to other platforms. It stores all the social data in the system, including the on- line statuses of all users, records of the existing sessions, and messages (invitations and chat histories) in each ses- sion. 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. It is the client side of the social cloud, residing in each surrogate in the IaaS cloud. The Messenger perio- dically queries the social cloud for the social data on beh- alf of the mobile user and pre-processes the data into a light-weighted format (plain text les), at a much lower frequency. The plain text files are asynchronously delivered from the surrogate to the user in a traffic friendly manner, i.e., little trafic 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 usersin the same session (if th e user chooses to syn- chronize witho thers). To achieve this, the syncer period- ically retrievesthe current playback progress of the session host and instructs its mobile user to adjust its playback po- sition. In this way, friends can enjoy the "sitting together" viewing experience. Different from the design of communication among messagers, syncers on different VM surrogates communicate directly with each other as only limited amounts of trafic are involved.

Mobile Client. The m any speci c 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 LiveStreaming protocol. Both are widely supported on most state-of-the-art smartphones. mobile client is not required to install

Gateway. The gateway provides authentication services for users to log into the CloudMoV 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. In addition, the gateway also stores each user's friend list in a plain text le (in XML formats), which is immediately uploaded to the surrogate after it is assigned to the user.

B. 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, almost any mobile device is ready to gain access to the CloudMoV services, as long as it is installed with an HTTP browser. The VM surrogates provisioned in the IaaS cloud cooperate 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 ortability and easy maintenance of the system. For social message exchanges among friends, CloudMoV em- ploys asynchronous communication. All the exchanged messages are routed via the surrogates to the social cloud, which efficiently organizes and stores the large volumes of data in a Big Table-like data store. The VM surrogates query the social cloud frequently and processes the retrieved data into XML files, for later retrieval by users in an asynchronous fashion. Such a design effectively separates the mobile users from the social cloud to signi cantly simplify the architecture, while the extra delay introduced at the VM surrogates is ignorable.

C. Pipelined Video Processing

Both live streaming of realtime contents and on-demand streaming of stored con tents are supported in CloudMoV. Video processing in each surrogate is designed to work on the fly, i.e., th e 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. 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 cloud where the surrogates are deployed, represents an ideal platform for implementing such computation intensive jobs.

D. Burst Transmissions

1) 3 G 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. A 3G carrier may commonly transfer a UE from a high-power state to a low-power state (state demotion), for releasing radio channels allocated to this UE to other users. For example, if a UE working at a high-power state does not incur any data traffic for a p reconfigured 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-powerone. Timeouts of the critical inactivity timers for state transitions are properly set by the carrier to guarantee Performance in both delay and energy consumption, since extra delay and energy consumption are potentially incurred for acquiring new radio channels when the UE transits from a low-p ower state to a high-power one later (state promotion).

2) Transmission Mechanism: In CloudMoV, we aim at maximum conservation of the battery capacity of the mobile de-vice, and design a burst transmission mechanism for streaming between the surrogate and the device. Using the HTTP live streaming protocol [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 seg-ment 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) 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; how-ever, for streaming over a cellular network, we should avoid downloading more than what is being watched for one main reason: users may switch among channels from time to time and those prefetched contents are probably never watched, 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. On the other hand, state transitions introduce latency and energy over-heads, so the burst should be large enough to avoid frequent state transitions; otherwise, such overheads may diminish the energy saving achieved by an intelligent state transition mech-anism. We next derive a lower bound on the burst size, which guarantees positive energy saving by such intelligent state transition.

Let B be the average available bandwidth over a wireless connection, S be the burst size in the CELL_DCH state, and b be the video playback rate at the mobile user.PDCH ,PFACH, and PIDLE denote the power levels at states CELL_DCH, CELL_FACH, and IDLE, respectively. is the timeout of the critical inactivity timer (the transition time) for state transition from CELL_DCH to CELL_FACH, and Tfach→ IDLE transition from CELL_FACH to IDLE. Let ,PIDLE→FACH, PFACH→DCH, and PIDLE→DCH be the energy needed for state promotion denoted by the respective subscript. We ignore the delay overhead in the state promotion from a low-power state to a high-power state, since its value is small—less than one second—based on our real-life measurements as reported in Section V. 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 Pcount(t) being the power

Level at time t during the transmission. An illustration of power consumption in both cases is given in Fig.2.The burst transmission operates at state CELL_DCH to send a total amount of data S for a duration of S/B ;then it transits to state IDLE via state CELL_FACH, and remains there for duration S/b — S/B – TDCH→FACH – tFACH →IDLE (S/b is the time taken for the mobile user to play the segment of size). 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 Pcount(t) = PDCH. We calculate the overall energy saving (ΔE) by burst transmission of the video over the time span T (multiples of S/b), as compared to the continuous transmission, as follows:

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

The burst size S should be chosen such that positive nergysaving, , ΔE > 0, can be achieved. A lower bound of the burstsize can be decided using ΔE > 0 . We also see that the larger

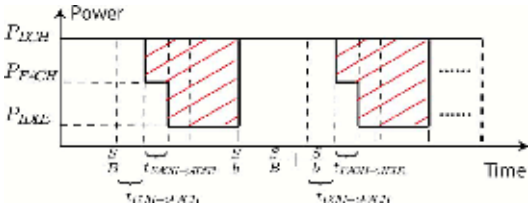

Fig. 2. Power consumption over time

III. Cloudmov: Prototype Implementation

Following the design guidelines in Section II, we have im-plemented 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. Data store, a thin layer built on top of Google's famous BigTable , 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, fi le 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) 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.com-pute-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 client starts to play the video as soon as the first segment is received.

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. Due to the intensive computation involved, we propose to implement all the video processing related tasks using ANSI C, to guarantee the performance. In particular, we install FFmpeg together with libavcodec as the groundsill library to develop the transcoding, segmentation and reshaping modules on the VM surrogates. 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,

Fig. 3. Client UI of CloudMoV.

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

In real time with H264/AAC codecs. 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. A mobile user dynamically requests segments of these two different video streams, according to his current network connection speed. The transcoded stream is further exported to an MPEG-2 transporting stream (.ts), which is segmented for burst transmission to the user. The burst sizes depend on both the network bandwidth and video bit rate. We evaluate the impact of different burst sizes on the streaming quality and energy consumption in 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.

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 con-tainer, 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

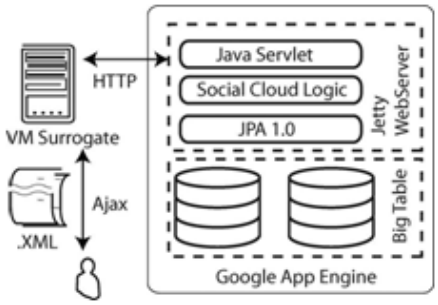

Fig. 5. Streaming architecture in each customized VM image (ami-b6f220df).

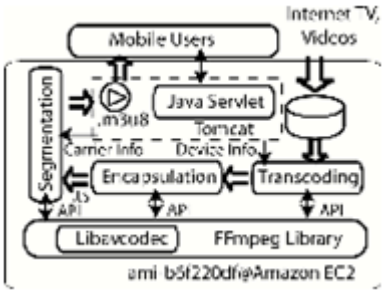

Fig. 6. Social message exchanges via Google App Engine 827

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 such that CloudMoV can be easily migrated to other PaaS clouds as well. Once a user logs in to the system and enters the URL of a video to watch, a session ID is generated for the new session (corresponding to viewing of this video), by combining the user's "username" in the system with the time stamp when the session is created. The gateway delivers an HTTP request to a Servlet listener running on GAE, to notify it that an entry for the newly joined user should be added, with the user's "username" as the key and other information (URL of the subscribed video, the session ID, etc.)

as the value. This entry will then be periodically retrieved through a public Servlet interface by surrogates representing the user's friends, in order to learn the updated status of the user over time. The default interval for retrieving updates of friends' online status is five minutes. When the user goes offline, the user online status record will be deleted.

Whenever a user decides to join a session hosted by his friend upon invitation, his VM surrogate switches to download the video of the session, and at the same time sends an HTTP request to the social cloud, for updating the session ID in this user's entry to the new one. If the user wishes to synchronize his playback progress with that of the session host, his VM sur-rogate synchronizes with the session host to maintain the play-back "currenttime" value (HTML5 property).

The social cloud maintains a "Logs" entry for each existing session in CloudMoV, with the session ID as the primary key and an array list as the value, which corresponds to individual messages in this session. When a user in a session posts a com-ment, this message is first sent to his VM surrogate, which fur-ther injects the message into the social cloud via another Servlet listener. The message is stored as a "Message" entry in the social cloud, with the message content as the value, and an auto-gen-erated integer as the key. Entries "Logs" and "Message" are an-notated by a @OneToMany relationship, to facilitate the data management. VM surrogates of users in the same session send periodical HTTP query requests to the social cloud for the latest comments from others. The default interval for retrieval of new comments is 10 seconds. The retrieved messages are stored and updated on the surrogates, which process them into well-formed XML formats for efficient parsing at the user devices. The user devices retrieve the XML files from the surrogates at a lower fre-quency (with default interval 1 minute), in order to minimize the power consumption and the traffic. Fig. 6 presents social mes-sage exchanges among a mobile user, his VM surrogate, and the GAE.

A large number of entries in the social cloud becomes out-dated very soon, since users may switch from one session to another, quit the system, and so on. We launch a cron job behind the scene every 10 minutes to clear those outdated entries. For example, for sessions of which everybody has left, their "Logs" entries and all the associated "Message" entries are deleted in a single transaction.

IV. Concluding Remarks

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 frame-work, CloudMoV, that makes use of both an IaaS cloud and a PaaS cloud. The framework provides efficient transcoding ser-vices for most platforms under various network conditions and supports for co -viewing experiences through timely chat ex-changes 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 en-ergy-effi cient burst transmission mechanism that can effectively increase the battery lifetime of user devices.

We have implemented a realistic prototype of CloudMoV, de-ployed 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 mes-sage 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 scala-bility. The experiments also highlight the drawbacks of the cur-rent 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.

Much more, however, can be done to enhance CloudMoV 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 which is a highly efficient secondary storage on the data stores. We seek to integrate memcache support into CloudMoV, by possibly memcaching the data (e.g., chat histories) of sessions where friends chat actively, so as to further improve the query perfor-mance. To sustain the portability of the system, we will stick to standard API interfaces, i.e., JCache (JSR 107), in our system.

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

[4] T. Coppens, L. Trappeniners, and M. Godon, "AmigoTV: Towards a social TV experience," in Proc. EuroITV, 2004.

[5] 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.
[6] A. Carroll and G. Heiser, "An analysis of power consumption in as smartphone," in Proc. USENIXATC, 2010.
[7] What is 100% Pure Java. [Online]. Available: http://www.javacoffee-break.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 gen-eration mobile networks," Comput. Netw., vol. 52, pp. 228–247, Jan. 2008.
[9] DVB-H. [Online]. Available: http://www.dvb-h.org/.
[10] K. Chorianopoulos and G. Lekakos, "Introduction to social TV: En-hancing the shared experience with interactive TV," Int. J. Human-Comput. Interaction, vol. 24, no. 2, pp. 113–120, 2008.