CloudMoV: Cloud-Based Mobile Social TV

Mrs. Sowbhagya M P  
Asst. Professor, Dept. of ISE,  
City Engineering College,  
Bangalore  
Visvesvaraya Technological University, Karnataka, INDIA  
sowbhagya.mp@gmail.com

Ms. Manisha  
Student, Department of ISE,  
City Engineering College,  
Bangalore  
Visvesvaraya Technological University, Karnataka, INDIA  
manishanaik7992@gmail.com

Ms. Samata  
Student, Department of ISE,  
City Engineering College,  
Bangalore  
Visvesvaraya Technological University, Karnataka, INDIA  
samatasuma@gmail.com

Ms. Harshitha  
Student, Department of ISE,  
City Engineering College,  
Bangalore  
Visvesvaraya Technological University, Karnataka, INDIA  
Harshitha_lingesh@yahoo.com

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-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 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.

1. INTRODUCTION

Although many mobile social or media applications have emerged, truly 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, and 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 [1]–[3], 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 experience. While social TV enabled by set-top boxes over the traditional TV systems is already available [4], [5], 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 (Infrastructure-as-a-Service) cloud and a PaaS (Platform-as-a-Service) cloud. Our design achieves the following goals: encoding flexibility, battery efficiency, spontaneous social interactivity, portability.
2. 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 [6], [7] 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 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 [4], [5], [8]. Coppens et al. [4] try to add rich social interactions to TV but their design is limited to traditional broadcast program channels. Oehlberger[5] conduct a series of experiments on human social activities while watching different kinds of programs. Though inspiring, these designs are not that suitable for being applied directly in a mobile environment. Chuah et al. [9] extend the social experiences of viewing traditional broadcast programs to mobile devices, but have yet to deliver a well-integrated framework. Schatz et al. [10], [11] 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. A recent work by Oehlberger [5] conduct a series of experiments on human social activities while watching different kinds of programs. Though inspiring, these designs are not that suitable for being applied directly in a mobile environment.

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. A recent work by Oehlberger [5] conduct a series of experiments on human social activities while watching different kinds of programs. Though inspiring, these designs are not that suitable for being applied directly in a mobile environment.

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. A recent work by Oehlberger [5] conduct a series of experiments on human social activities while watching different kinds of programs. Though inspiring, these designs are not that suitable for being applied directly in a mobile environment.

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.

3. CLOUDMOV: ARCHITECTURE AND DESIGN

As a novel Cloud-based Mobile socialTV 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.

A. Key Modules

Fig. 1 gives an overview of the architecture of CloudMov. A surrogate (i.e., a virtual machine (VM) instance), or a VMSurrogate equivalently, is created for each online mobile users in the PaaS 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 responsible 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 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 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) [13]. 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 periodically queries the social cloud for the social data on behalf of the mobile user and pre-processes the data into a lightweight 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 the 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, syncers on different VM surrogates communicate directly with each other as only limited amounts of traffic are involved.

- **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., MobileSafari, Chrome, etc.), and supports the HTTP Live streaming protocol [14]. Both are widely supported on most state-of-the-art smartphones.

- **Gateway.** The gateway provides authentication services for users to log in to 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 by the gateway and destroys VM instances according to the current workload. In addition, the gateway also stores each user’s friend list in a plain text file (in XML formats), which is immediately uploaded to the surrogate after it is assigned to the user.

We describe the key designs in CloudMoVin the following.

### 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 the 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 ease of maintenance of the system. For social message exchanges among friends, CloudMoV employs 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 significantly simplify the architecture, while the extra delay introduced at the VM surrogates is ignorable, as shown in Section V.

### C. Pipelined Video Processing

Both live streaming of real-time 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 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 segments received. To support dynamic bit rate switching, 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) **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 [15]. 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 [15]. 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 [15] (this has also been verified in our experiments in Section V). 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 datatrace 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-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 newradio channels when the UE transits from a low-power state to a high-power one later (state promotion).

2) Transmission Mechanism: In CloudMoV, we aim at maximizing the battery capacity of the mobile device, and design a burst transmission mechanism for streaming between the surrogate and the device. Using the HTTP livestreaming protocol [14], 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 to the mobile device, upon sucha 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; however, 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 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 overheads, 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 mechanism. 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, $P_{DCH}$, $P_{FACH}$ and $P_{IDLE}$ denote the power levels at states CELL_DCH, CELL_FACH, and IDLE, respectively. The timeout of the critical inactivity timer is $t_{DCH\rightarrow FACH}$, which is the timeout for state transition from CELL_DCH to CELL_FACH. Let $P_{IDLE\rightarrow FACH}$, $P_{FACH\rightarrow IDLE}$, and $P_{IDLE\rightarrow DCH}$ 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. An illustration of power consumption in both cases is given in Fig. 3.1. The burst transmission operates at state CELL_DCH to send a total amount of data for a duration of; then it transits to state IDLE via state CELL_FACH, and remains there for duration (is the time taken for the mobile user to play the segment of size).

Note that the power consumption level during transition periods $t_{DCH\rightarrow FACH}$ and $t_{FACH\rightarrow IDLE}$, remains at $P_{DCH}$ and $P_{FACH}$, respectively, although no data is transmitted then. The continuous transmission always operates at the high-power state CELL_DCH with power level $P_{cont}(t) = P_{DCH}$. We calculate the overall energy saving ($\Delta E$) by burst transmission of the video over the time span (multiples of $s/b$), as compared to the continuous transmission, as follows:

$$\Delta E = \int_0^T (P_{cont}(t) - P_{burst}(t))dt = \int_0^{S/b} (P_{cont}(t) - P_{burst}(t))dt \times \frac{T}{b}$$

$$= \frac{T \times b \int_0^{S/b} \left( P_{cont}(t) - P_{burst}(t) \right)dt}{S}$$

$$= \frac{T \times b \times \left( (P_{DCH} - P_{FACH}) \times t_{FACH\rightarrow IDLE} + (P_{DCH} - P_{IDLE}) \times \left( \frac{S}{b} - t_{FACH\rightarrow IDLE} - t_{DCH\rightarrow FACH} \right) - P_{IDLE\rightarrow FACH} - P_{IDLE\rightarrow DCH} - P_{FACH\rightarrow DCH} \right)}{S}$$

(1)

The burst size should be chosen such that positive energy saving can be achieved. A lower bound of the burst size can be decided using $\Delta E > 0$. We also see that the larger $S$ is, the more energy saving we can achieve using burst transmission. However, with a large $S$, a user has to wait for a long time before the first segment is transcoded and delivered, and there is also the risk that it may download contents it will never watch.

Fig. 3.1 Power consumption over time.

4. CloudMoV: 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.

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) [14] protocol is supported. The user first connects to the login page of CloudMoV, as

www.ijert.org

ISSN: 2278-0181

NCRTS’14 Conference Proceedings

International Journal of Engineering Research & Technology (IJERT)
illustrated in the topleft 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. 4. 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.

Fig. 4. Social message exchanges via Google App Engine.

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, 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 and a low-quality stream 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 size depends on both the network bandwidth and video bit rate.

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 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.

5. CONCLUDING REMARKS 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
resources supports and rich functionalities of cloud computing services. We introduce a generic and portable mobile social 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 state 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 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 [14] 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 performance. To sustain the portability of the system, we will stick to standard API interfaces, i.e., JCache (JSR 107), in our system.

REFERENCES
