International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Impact Factor (2012): 3.358

Mobile Cloud Computing Applied in Social TV

K Ramu[1], A Jyothi[2]

[1]M. Tech student, Department of CSE, Anurag Group of Institutions, Hyderabad, India

[2]Assistant Professor, Department of CSE, Anurag Group of Institutions, Hyderabad, India

Abstract: ***As the growth in technology the usage of personal mobile devices are providing much richer contents and social interactions to users while moving. This trend however is throttled by the limited battery lifetime of mobile devices and interrupted wireless connectivity, to guarantee the possible quality of service faced by mobile users not feasible. The cloud computing technology, with its resources used to overcome the limitations of mobile devices and connections can potentially provide the best platform to support the desired mobile services. Tough challenges arise on how to use the cloud resources to provide mobile services, especially those with severeg9 interaction delay of requirements. In this paper, we intend the design of a Mobile Cloud Computing Applied in Social TV. The system can utilizes both PaaS (Platform-as-a-Service) and IaaS (Infrastructure-as-a- Service) cloud services to provide the living-room experience of video watching to a group of mobile users who can interact socially while sharing the video.***

Keywords: Mobile CloudMov, Cloudlet, VM Surrogate, Social Cloud, Burst Transmission

1. Introduction

As the rapid growth in technology, many mobile social media applications have emerged, most of the smart devices gaining mass acceptance are still retarded by the limitations of the current wireless and mobile technologies, among which battery lifetime and interrupt connection bandwidth are the most difficult ones. Cloud computing is the newly-emerged computing model for scalable, low-cost, agile resource provider, to support efficient mobile data communication. In this technology infinite hardware and software resources are visualized, the cloud technology can offload the computational tasks that are involved in a mobile application and it can significantly reduce battery consumption at the mobile devices, by implementing appropriate design. The major confront in front of us is how to effectively utilize the cloud services in mobile applications. Few studies have been there on designing mobile cloud computing systems [1][3], but none of them are deal with the spontaneous social interactivity among mobile users. if a proper design is in place application consumption of battery is less at the mobile devices,. The huge task in front of us is the effectively exploit cloud services to facilitate mobile applications. There have been a few applications 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 Mobile Cloud Computing applied in social TV system which can effectively exploit the cloud computing model to provide a living-room experience of video watching to different mobile users with spontaneous social interactions. In Mobile Cloud based social TV, mobile users can watch a live or on-demand video to watch from any video site, invite their friends to watch the same video concurrently, and communicate with their friends through chat while enjoying the video. Now a days people are geographically disparate from their family and friends and hope to share their feelings this type of cases instead of traditional TV watching, mobile cloud social TV is well suited. While social TV empowered by set-top boxes over the traditional TV systems is already available [4], [5], it remains a challenge to achieve mobile cloud social TV, where the concurrently viewing experience with friends is enabled on mobile devices. We design mobile cloud social TV which can effectively utilizes both PaaS (Platform-as-a-Service) and IaaS (Infrastructure-as-a-Service) cloud services.

2. Literature Survey

A number of mobile TV systems have originated in recent years, impelled both hardware and software advances in mobile devices. Some early systems [8], [9] bring the “living-room” experience to small screens while moving. But they cornerstone on barrier clearance in order to perceive the convergence of the television network and the mobile network, than traversing the demand of “social” interactions among mobile users. There is a trend in which efforts are dedicated to extending social elements to television systems [4], [5], [10]. Coppens et al. [4] try to add rich social interactions to TV but their design is finite to traditional broadcast program channels. Oehllberg et al. [5] conduct a several experiments on human social activities while watching different kinds of videos. Though inspiring, these schemes are not that suitable for being applied directly in a mobile atmosphere. Chuah et al. [11] extend the social experiences of watching traditional broadcast programs to smart phones, but have yet to deliver a integrated structure. Schatz et al. [12], [13] have designed a mobile social TV applications, which is personalized for DVB-H networks and Symbian devices as against to a broad audience. Compared to these previous work and systems, we focus at a design for a scalable, portable mobile social TV structure, providing co-viewing experiences among friends over geographical separated through mobile devices. Our structure is open to all Internet based video programs, either live or on-requirement, and supports a broad range of mobile devices with HTML5 compatible browsers installed, without any other obligatory component on the devices.

For any application impended at mobile devices, reducing power consumption is repeatedly one of the major concerns and challenges. Flinn et al. [14] exploit cooperation between the mobile OS and the mobile applications to balance the energy consuming and system performance. Yuan et al. [15] explore mobile multimedia streaming, similar to most of the other network, by modifying the CPU power for energy saving, however, according to the present measurement work of Carroll et al. [6], the display and the wireless network card and not the CPU consume more than half of the overall power consumption in smart phones. Our work is able to achieve a significant power saving nearly 30%, by opportunistically switching the device between high-power and low-power transmission modes during streaming. Based on existing work (e.g., Anastasi et al. [15]) have provided precious 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.

Cloud computing had its debut with much flourish and is now considered a most powerful hosting platform in many areas including mobile cloud computing. Satyanarayanan et al. [1] explores 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 structure for mobile code offloading to the cloud computing. Zhang et al. [13] explores an elastic mobile application model by offloading parts of the applications (weblets) to an IaaS cloud service. All those work target at computational job offloading. We instead advocate non-layered coding in such delay-sensitive mobile applications, although the detailed transcoding algorithm designs are out of the scope of this work. In addition, we phenomenally employ a surrogate for each mobile user in the cloud rather than relying on a devoted cluster, which can be more easily implemented in practice. Liu et al. [20] build a mobile cloud based social interaction structure on top of the Google App Engine and offer an iOS implementation. We implemented our design a portable, generic, and robust structure to enable realtime streaming and social interaction synchronously, which is not vault to any specific cloud platform. Even-though our archetype is implemented on only two public clouds, i.e. Amazon Engine and Google App Engine, it can be easily migrated to other cloud systems as long as the targeted cloud platforms conform to the affiliated standard.

3. Architecture and Design of Mobile Cloud Computing Applied in Social TV

As a novel Mobile Cloud Computing Applied in Social TV system 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 might chooses, such as a TV program provider or an Internet video streaming site, with fitted encoding formats and rates for the device each time.

(2) Co-viewing with social exchanges : A user can invite his friends to watch the same video, and exchange text messages about video while watching. The multiple group of friends watching the same video is referred to as a conclave. The mobile user who starts a session is the host of the session. We present the architecture of Mobile Cloud A[[lied in social TV and the detailed designs of the modules in the following.

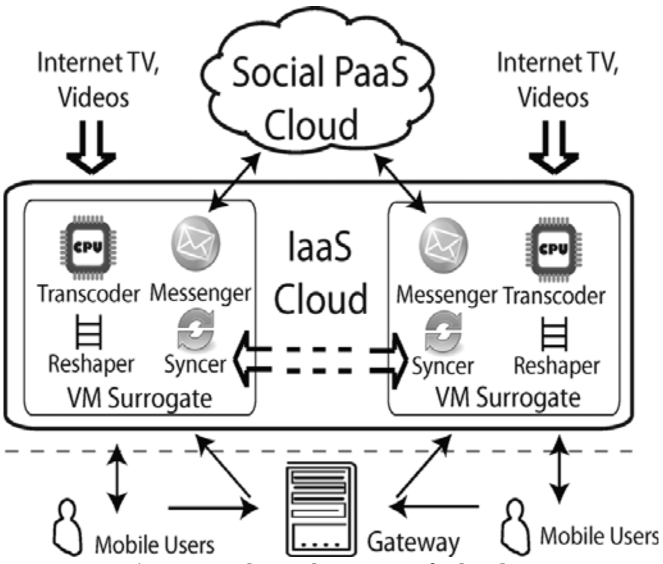

Figure 1: The architecture of CloudMoV.

Fig 1.gives an overview of the architecture of Mobile Cloud Applied in Social TV. A surrogate (i.e., a virtual machine (VM) instance), or a VM surrogate equivalently, is created for each mobile user in an IaaS cloud platform. The surrogate acts as a proxy between the smart device and the video sources, provides transcoding services as well as segmenting the streaming traffic for shatter transmission to the user. They are also responsible for handling commonly exchanged social messages among their group of users in a timely and effective manner, conserving mobile devices from unnecessary traffic and improving battery efficient, immediate social interactions. The surrogates interchange social messages via a back-end PaaS cloud platform, which adds robustness and scalability to the system. There is a gateway server in MobileCloudMov that keeps track of participating users and their VM surrogates, which can be achieved by a standalone server in the IaaS cloud platform.

4. Modules

The design of Mobile Cloud Applied in Social TV can be divided into the following major functional modules.

A. Transcoder: It resides in every surrogate, and is accountable for dynamically deciding how to encode the video stream from the video source in the relevant format, dimension, and bit rate. Before delivery to the mobile user, the video stream is further trans coded into a proper transport stream. In our implementation, every video is exported as MPEG-2 transport streams, which is the default standard to deliver digital video and audio streams over lossy channel.

B. Reshaper: The reshaper in each surrogate receives the encoded transport stream from the transcoder, divide it into segments, and then sends each segment in a barrage to the mobile device based on its request to improve the best power efficiency of the mobile device. The barrage size, the amount of data in each barrage, is carefully decided according to the 3G technologies materialized by the corresponding carrier.

C. Social Cloud: It is the client side of the social cloud, populating in each surrogate in the IaaS cloud. The Messenger sporadically queries the social cloud for the social data on behalf of the mobile user and pre-processes the data into a light-weighted format 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 incurved. In the opposite direction, the messenger disseminates this user's messages (invitations and chat messages) to other users via the data store of the social cloud

D. Mobile Client: The mobile client is not required to install any precise client software in order to use Mobile Cloud Applied in Social TV, as long as it has an HTML5 well-suited browser (*e.g.*, Mobile Safari, Chrome, etc.) and supports the HTTP Live streaming procedure [24]. Both are widely supported on most state-of-the-art smart devices

E. Gateway: The gateway provides authentication services for users to log in to the Mobile CloudMov system, and stores users' credentials in a undeviating table of a MySQL database it has installed. It also stores the datain the pool of currently available VMs in the IaaS cloud platform in a further 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 assurance small query latencies, since the VM pool is updated regularly as the gateway assets and destroys VM instances according to the current workload. In addition, the gateway also stores each user's friend list in a plain text format, which is instantly uploaded to the surrogate after it is assigned to the user.

5. Prototype Implementation

Following the design guidelines in Section III, we have implemented a real-world mobile social TV scheme, and deployed it on the Google App Engine (GAE) and Amazon EC2 clouds, which are the two most broadly used open PaaS and IaaS cloud platforms.

GAE, as a PaaS cloud, provides prosperous services on top of Google's data centers and enables rapid utilization of Java-based and Python-based applications. Data accumulate, a thin layer build on peak of Google's famous BigTable [26], handles "big" data retrieve well with linear and modular scalability even for high-throughput usage paradigm. Hence, GAE is an ideal platform for implementing our social cloud, which energetically handles large volumes of messages. On the other side, GAE implement many constraints on application deployment, *e.g.*, lack of carry for multi threading, file storage which may delay both computation-intensive jobs and content delivery applications.

Amazon EC2 [27] is a representative IaaS cloud, presents raw hardware resources including storage, CPU and networks to mobile users. Most EC2 VM instances are launched with the Linux kernels are Xen-para-virtualized as domu guests on head of dom0, which run straight on the bare-metal hardware upon booting. As the chief virtualization technology in the Linux community together with KVM [28], Xen chains para-virtualization on almost all hardware with Linux drivers, and hence gives close-to-native concert, especially for CPU virtualization and I/O virtualization, as has been verified by wide dimensions including all. Comparing to a general PaaS cloud, EC2 is an suitable platform for computation-intensive tasks in Mobile CloudMov.

5.1 Client Use of Mobile CloudMov

All mobile devices installed with HTML5 compatible browsers can use Mobile CloudMov services, as long as the HTTP Live Streaming (HLS) [24] procedure is supported. The user first access to the login page of Mobile CloudMoV, as illustrated in the top left corner of Fig. 3. After the user successfully logs in the application through the gateway, he is assigned a VM surrogate from the Virtual Machine pool. Then the user is automatically redirected to the assigned VM surrogate, and welcomed by a home page as shown on the right-hand side of Fig. 3. After user login, the portal configure the device configuration information by exploratory the "User-Agent" header ethics, and this information will be sent to its surrogate for assessment making of the video encoding formats. The mobile user can enter the URL of the video to watch, on the Subscribe tab of the portal page; after he clicks the Subscribe tab, the address of the video is transferred to the VM surrogate, on behalf of the user VM surrogate download the video, transcodes and sends properly encoded format to the user. From the substitute to the mobile device, the video stream delivered with HLS is always separated into group of segments, with a playlist file giving the indices. A playlist file may become outdated if new contents are generated. The user can start to play the video as soon as the first segment is received.

While watching a video, the user can check out his friends status on the Friends tab and also invite one or more friends to join with him in watching the video. While watching the video a user can receives an invitation from a friend and decides to join the session. Users in the same session can swap over feelings and comments on the Chat tab, where new messages can be entered and the chat history of the session can be viewed.

5.2 VM Surrogates

All the VM surrogates are provisioned from Amazon EC2 web services. We propose to employ all the video processing related tasks using ANSIC, to assurance the performance. In particular, we install FFmpeg mutually with libavcodec as the groundsill library [11] to expand the transcoding, reshaping and segmentation modules on the VM surrogates. We also installed a Tomcat web server to serve as a Servlet container and server on each surrogate. Both FFmpeg and Tomcat are open the source projects. Whenever a VM surrogate receives a video subscription request from the mobile user, it downloads the video from the respective URL, and processes the video stream by transcoding and segmentation, based on the user device configurations through the portal. For example, in our experiment, the downloaded video is transcoded into a low-quality stream and a high -quality stream.

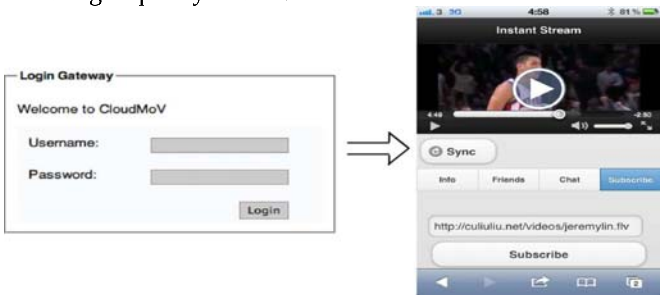

Figure 3: Client UI of CloudMoV.

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 enthusiastically requests segments of these two different video streams, according to his present network bond speed.

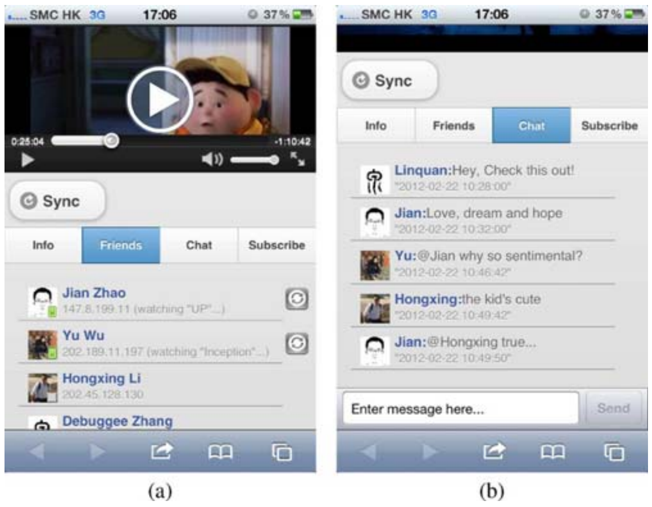

Figure 4: Friend and Chat tabs. (a) Friend tab. (b) Chat tab

The converted stream is further exported to an MPEG-2 transporting stream, which is divided for burst transmission to the user. The barrier sizes depend on both the network bandwidth and video bit rate. We appraise the impact of dissimilar burst sizes on the streaming quality and energy utilization in details in Section V. Fig. 5 shows the streaming design in our modified VM image. Here, the modules on social message exchanges are absent, which will be shown in Fig. 6.

5.3 Data Models in the Social Cloud

We use the back-end data store as GAE to keep the transient states and data of Mobile CloudMOV including users online current status, public chat messages in all the sections. With Jetty used as the fundamental Servlet container, mainly Java-based applications can be easily synchronized to GAE, under restricted usage constraints, where no platform-specific APIs are imposed for the exploitation. GAE provides both its Java Persistence API adapter and a set of proprietary low-level APIs to map the relational data. We prefer to use the former in Mobile CloudMoV such that CloudMoV can be easily migrated to other PaaS clouds.

Once a user logs in to the system and enters the URL of a video to watch, a session ID is created for the new session, by combining the user’s “username” in the application with the time stamp. The gateway sends an HTTP request to a Servlet listener which is running on GAE, to notify that an access for the newly joined mobile user should be added, with the user’s “username” as the surrogate key and other information as the value.

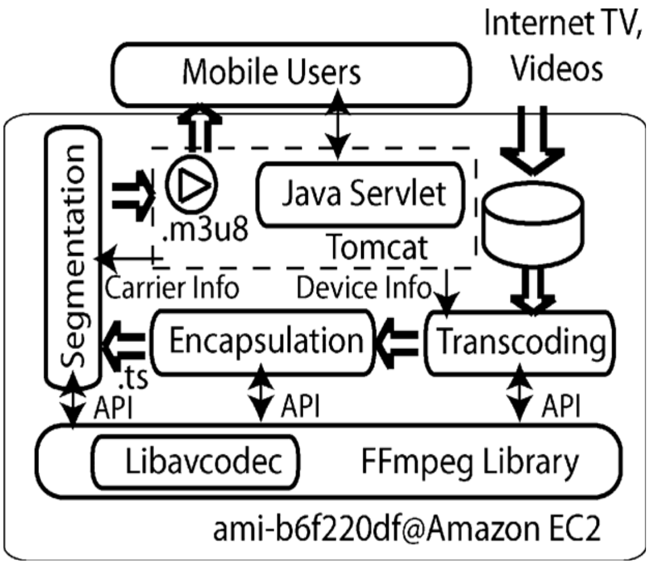

Figure 5: Streaming architecture in each VM image

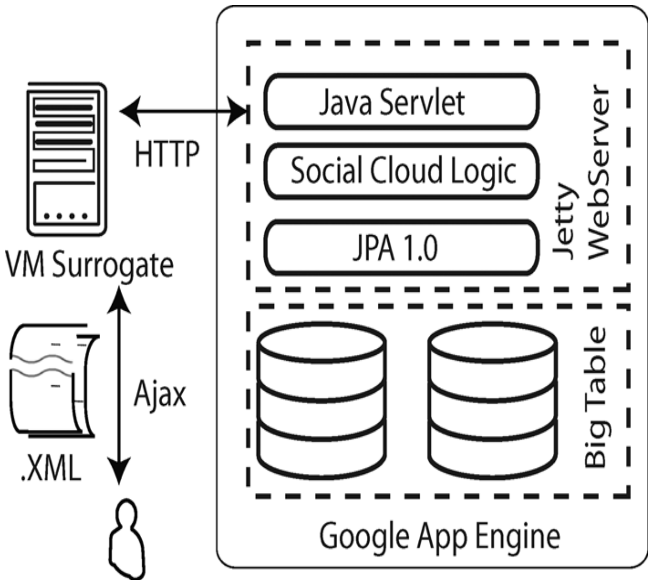

Figure 6: Social message exchanges via Google App Engine.

.

The social cloud maintains a Logs entry for each existing session in Mobile CloudMov with the current session ID as the primary key and the list as value, which associates with individual messages in current session. When a user in a session posts a comment, this message is first transferred to his VM surrogate, which is further post the message into the public cloud through another Servlet listener. The message is stored as a Message entry in the public cloud, with the message content as the value, and an auto generated integer as the key. Access “Logs” and “Message” are annotated by a One To Many relationship, to assist 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 theXML files from the surrogates at a lower frequency (with default interval 1minute), in order tominimize the power consumption and the traffic. Fig. 6 presents social message exchanges among a mobile user, his VMsurrogate, and the GAE.

A large number of entries in the social cloud becomes outdated 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.

6. Real-World Experiments

We carry out both unit tests and performance evaluations of *CloudMoV* deployed on Amazon EC2 and Google App Engine, using a number of iPhone 4S smart phones (iOS 5.01) as the mobile clients, which have been registered on the Apple developer site. The gateway is implemented on a Virtual Private Server (VPS) hosted by Bluehost [10]. Unless stated otherwise, the experiments are conducted over the 3G cellular network of 3HK[11], which is one of the largest Telecomoperators in Hong Kong.

6.1 Measuring the RRC States

We primary design dimension experiments to discover the timeout values of the critical immobility timers employed in 3HK's 3G network, as discussed in Section III-D. We facilitate logging functions on an fully charged iPhone 4S and use the Mobile Safari to watch a YouTube video using Mobile CloudMoV services. The battery utilization traces on the phone are profiled by "Instruments", a prevailing tool of Xcode [32]. The playback rate of the video on the phone is about 256 Kbps.

Fig. 7 shows the power expenditure levels on the phone over time, in expressions of portions of the main device power level. The red perpendicular lines represent the initial points of playback intervals when the Safari runs in the forefront, and the green lines signify the end times of playback periods when the Safari is hanging in the background. We can see that our state evolution model in Fig. 2 is verified by these real-world measurements: when there is data communication, the device operates at the high power mode; when data broadcasting stops, the communication power of the device initial decreases to an transitional level, and then to a very low level. We also discover that timeouts of the immobility timers and are roughly 16 and 18 seconds, correspondingly. It shows that the period between the end of one burst communication to the start of the subsequently burst communication should be at least 50 seconds, to allow the phone to pierce a low-power advance. Following these measurement results, in our following experiments conducted over 3HK 3G network, we set the evasion burst communication phase to 70 seconds, the time from the initiate of one burst communication to the start of the next burst communication, consequent to the playback time of one burst subdivision, , where is the burst size. over time with dissimilar burst communication sizes.

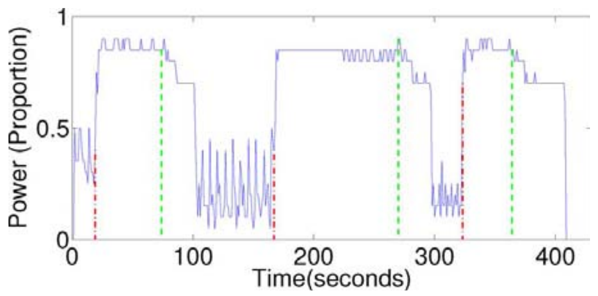

Figure 7: Power consumption over time on an iPhone 4S.

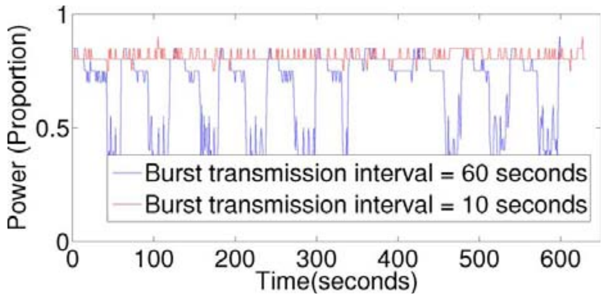

Figure 8: Power consumption

6.2 Impact of Burst Size on Power Consumption

The procedure of video segmentation is broadly employed in video streaming applications, but frequently for ease of division and not for battery effectiveness at potential mobile users. Apple Inc., which projected the HTTP Live Streaming protocol [24], suggests 10-second-playback segments, which has been followed in various streaming applications. We locate this segment size is challenging and can consume the battery of a mobile device rapidly. In Fig. 8, we compare the power utilization levels when burst communication period of 20 seconds and 70 seconds are used, correspondingly, for the iPhone 4S to stream a 12-minute YouTube flash video. We observe that, iOS devices can not play flash videos, but Mobile CloudMoV helps transcode the flash to the H264/AAC stream, which is compatible with our iPhone 4S.

We note that the device residue at the high power mode if the 20-second segmentation is used, since the state conversion takes slightest 30 seconds, as given in Section V-A. On the other hand, using 70-second burst communication periods, Mobile CloudMoV may transfer the device to the low power mode (IDLE) via the transitional power mode (CELL_FACH) from time to time. In this way, Mobile CloudMoV can realize around 39.1% power saving. We also examine some unpredicted performance of the power levels around 450 seconds, where the power does not drop. After examination Tomcat server logs on the VM surrogate, we locate that the device requested the play file double around that time, perhaps due to packet loss, and the tiny traffic of the playlist destitute the device from a "sleep" ability.

To verify whether such playback list updates may always exclude the device's power level from reducing, we have further

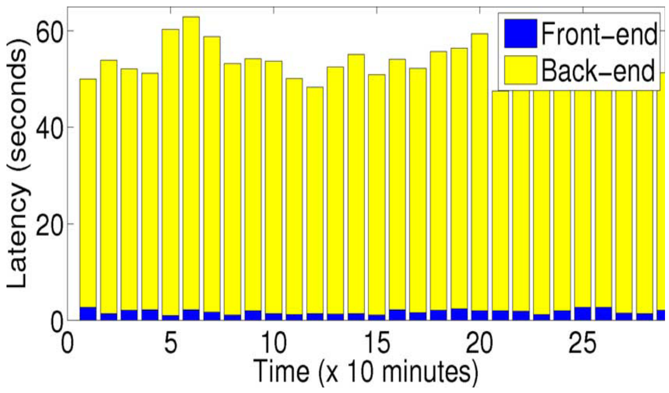

Figure 9: Average user sign-in latencies over time

Conducted tests by creating a live transmit stream from the same flash video and deploy it on Mobile CloudMoV. We discover that despite of the burst communication intervals we set, the device's power level does not drop due to repeated play list update in a live streaming. We omit the plots of the results since they are similar to the red curve in Fig. 8. Different browsers may configure different update frequencies, since the value is not specified in the HTTP Live Streaming protocol. This value should be suspiciously set, for potentially increasing battery lifetime at the mobile users.

6.3 Startup Latency of Video Playback

We estimate the transcoding performance on the surrogates in Mobile CloudMoV, first by calculating the playback startup latency on the surrogates, at the time of video subscription request is obtained from the mobile user to the time when the first transcoded burst segment is generated. We deploy the VM surrogates (ami-b6f220df) on three types of instances generated by Amazon EC2, with the complete configurations shown in Table I. For fair relationship, all the instances are deployed in the zone "east-1-c", and then transcode the same flash video used in experiments of Section V-B. Fig. 10 shows the playback startup latencies when dissimilar VM instances are used as the surrogate for an iPhone 4S, and dissimilar burst communication intervals are employed.

Configuration of VM Instances

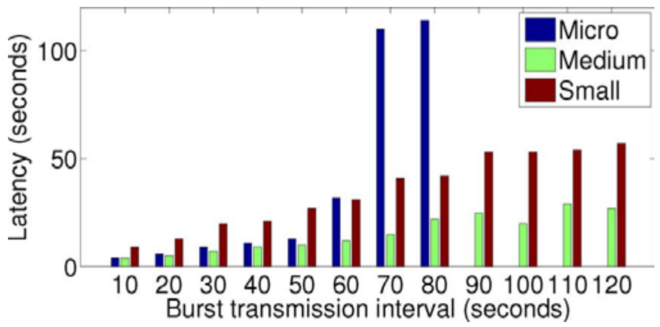

Figure 10: Startup latency at different burst sizes.

In our experiments, we experienced the network connection bandwidth flanked by the Amazon EC2 instance and the YouTube, and observed that video downloading from YouTube to the instances is very speedy. Therefore, the establish latency depends mainly on the burst period setting and the transcoding rapidity at the VM surrogate. Fig. 10 shows that the longer the burst period is, the bigger the segment of video to transcode is, and therefore the longer the startup latency is. We can see the intermediate instance achieves better transcoding concert with larger calculation ability, as compared to the small instance. We consider that it is caused by the expenses of load balancing among its two cores. This shows that the performance can be enhanced by a more proficient transcoding algorithm targeting at multi-core platforms, that will be part of our future work.

7. Conclusion

This paper presents our view of what might become a trend for mobile Social TV, i.e., mobile social TV based on sprightly resource supports and prosperous functionalities of cloud computing services. We launch a standard and convenient mobile social TV skeleton, Mobile CloudMoV, that makes utilize of both an IaaS cloud and a PaaS cloud. The paradigm provides efficient transcoding services for most platforms beneath various network situation and supports for co-viewing experiences through appropriate chat interactions amongst the screening users. By enrolling one surrogate VM for each mobile user, we attain decisive scalability of the scheme. Through an in-depth examination of the power states in business 3G cellular networks, we then recommend an energy-efficient burst communication mechanism that can successfully increase the battery lifetime of user devices. We have impended a practical prototype of Mobile CloudMoV, deployed on Google App Engine and Amazon EC2, where EC2 instances serve as the mobile users' surrogates and GAE as the public cloud to handle the huge volumes of public message interactions. We conducted carefully planned experiments on iPhone 4S platforms. The investigational results confirm the higher performance of Mobile CloudMoV, in terms of transcoding effectiveness, power saving, appropriate social interaction and extensibility.

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, "Cloudstream: Delivering high-quality streaming videos through a cloud-based SVC proxy," in *Proc. INFOCOM'11*, 2011

[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

[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.

[11] M. Chuah, "Reality instant messaging: Injecting a dose of reality into online chat,"

[12] R. Schatz, S. Wagner, S. Egger, and N. Jordan, "Mobile TV becomes social – Integrating content with communications," in *Proc. ITI*, 2007.

[13] R. Schatz and S. Egger, "Social interaction features for mobile TV ser- vices," in *Proc. 2008 IEEE Int. Symp. Broadband Multimedia Syst. and Broadcasting*, 2008.

[14] J. Flinn and M. Satyanarayanan, "Energy-aware adaptation for mobile applications," in *Proc. 17th ACM Symp. Operating Syst. Principles*,1999, SOSP '99.

[15] W. Yuan and K. Nahrstedt, "Energy-efficient soft real-time CPU sched- uling for mobile multimedia systems," in *Proc. 19th ACM Symp. Op- erating Syst. Principles*, 2003, SOSP '03, pp. 149–163.

[16] 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, ESSLLI-2000*, 2004, pp. 24–26.

[17] 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.

[18] W. Zhu, C. Luo, J. Wang, and S. Li, "Multimedia cloud computing,"*IEEE Signal Process. Mag.*, vol. 28, pp. 59–69, 2011.

Author Profile

K Ramu received the B.Tech degree in computer science and Engineering from JNTU Hyderabad in 2011 and pursuing M. Tech. degree in Computer science and Engineering from JNTU Hyderabad, Telangana State.

A. Jyothi working as assistant professor in Computer Science Engineering from CVSR College of Engineering from Anurag Group of Institutions Venkatapur(V), Ghatkesar(M), Ranga Reddy District, Hyderabad-500088, Telangana State.