Bandwidth Efficient Adaptive Live Streaming with Cooperative Devices in Mobile Cloud Computing

Xiaoyi Zhang, Geng Xi, Kaiming Qu, and Lin Zhang(✉)

School of Information and Communication Engineering,
Beijing University of Posts and Telecommunications, Beijing, China
Zhangxy_bupt@126.com, as_to@msn.cn, qukm90@gmail.com,
zhanglin@bupt.edu.cn

Abstract. Recent developments have heightened the optimization of Dynamic Adaptive Streaming over HTTP (DASH) services in mobile network condition (e.g. LTE/WIFI networks). The aim of this paper is to discuss how Mobile Cloud Computing (MCC) can assist DASH in a special scenario that a set of neighboring mobile devices take interests in watching the identical video stream. A mechanism is proposed to provide higher resolution live streaming with less expense (both in bandwidth and dollar-cost per device) by the cooperation among devices. The cloud-based live stream server will transcode the original stream segment according to the estimation of the devices' group bandwidth, every device then share received fragments (part of the segment) with each other through the free device-to-device interface, and finally gets the whole segment. An emulation testbed is realized with Android Smartphone implementation according to the proposed improvement to DASH. Experiments results demonstrate its performance with today's commercial players on the Quality of Experience (QoE), Peak Signal to Noise Ratio (PSNR) and network utilization across a range of scenarios.

Keywords: Dynamic Adaptive Streaming · Mobile Cloud Computing · Cooperative · Smartphone

1 Introduction

Recent technological advances are bringing Over-The-Top (OTT)-based video streaming services over mobile networks closer to reality. In mobile network, the available bandwidth differs among each device and changes over time. Compared with the wired Internet, it is even more important and difficult for the Dynamic Adaptive Streaming over HTTP (DASH) [1]service to select the proper video rate. The design of robust adaptive HTTP streaming algorithms is critical not only for the performance of video applications, but also the performance of the mobile network as a whole.

Today's most commercial DASH services are built on standard HTTP servers, the rate selection is solely realized by the client device[2]. However, in consideration of storage costs, it is very difficult to the server to provide numerous options of media

V.C.M. Leung et al. (Eds.): TridentCom 2014, LNICST 137, pp. 293–303, 2014.
DOI: 10.1007/978-3-319-13326-3_28

streaming quality, and if the client device bandwidth condition changes drastically, the appropriate media quality cannot be instantly switched, which degrades user experience. Mobile Cloud Computing (MCC) is an emerging technology to improve the quality of mobile services. MCC can improve the performance of mobile applications by offloading data processing from mobile devices to servers. With the help of MCC, the network condition of the whole mobile and server environment can be taken into consideration to determine the real-time transcoding procedure [3].

In this paper, we consider a scenario that a set of neighboring mobile devices take interests in watching the identical video stream. This scenario will occur when a group of users want to watch a live show or soccer game on their own mobile devices. The best stream coding rate may not be reached *if* the device asks for the stream separately. Worse still, when the mobile devices are within the same cover range of the base station, the network resource will be more scarce. However, if these users are willing to cooperate as a group, the mobile connections can be combined together to apply a better streaming rate and then extended by wireless Device to Device (D2D) links established through WiFi-Direct or Bluetooth [4].

Considering the above scenario, a cloud assisted real-time transcoding mechanism is proposed in this paper, which contains a bandwidth recorder, a media transcoder, a group-based resource manager and a HTTP live streaming server. This paper focus on our improvement to DASH and experiments on the emulated testbed with Android smartphones, aiming at providing better Quality of Experience (QoE) for mobile users.

The structure of the rest of the paper is as follows. Section 2 presents related work. Section 3 gives the scheme and the algorithm in details for each module in the system. Section 4 presents an emulation testbed and gives our analysis to the result. Finally, conclusions are drawn in Section 5.

2 Related Work

It is clear that the coding rate selection is determined by devices in most of the commercial DASH services. However, it is difficult to measure the accurate bandwidth of device-side above HTTP layer [2]. The natural variability of video content is taken into consideration to reduce the quality fluctuation rather than measured network bandwidth in [14], which do not consider the cooperation of mobile devices. The trade-offs problem is studied by [9] between the two QoE metrics—probability of interruption in media playback, and the initial waiting time before starting the playback.

A file downloading system within the adjacent cooperated mobile devices is proposed by [5], however the purpose of this paper is not aiming at streaming video. Furthermore, the paper [6] has considered the similar scenario, but the energy consumption is the main concern in this paper instead of QoE. Moreover, the device cooperation mechanism is quite simple that the captain device downloads the whole media content from cloud and sends it to its members by broadcasting. So the devices could not take advantage of cooperation which may lead to better video quality and less traffic cost. The system in [4] and [7] have nearly the same cooperation with this paper, but it does not consider DASH and cannot provide adaptive streaming to improve video quality.

A DASH based standard is presented in [10] considered the interoperability needs due to devices and servers that come from various vendors. Traditional methods demand multiple versions of the same video content with different bit rates stored on the server which may incur tremendous storage overhead, so cloud-assisted adaptive video streaming is proposed by [3] and [11]. A cloud based transcoder is proposed and implemented by [12] with an intermediate cloud platform to provide the format resolution considered the gap between Internet videos and mobile devices. In [13], a control-theoretic approach to select an appropriate bitrate is proposed with multiple content distribution servers but the author do not take advantage of the server cluster's computing capability and increase the burden of the client. However, these approaches on cloud do not consider the cooperation of mobile devices.

3 Proposed Scheme

In this paper, a prototype of the system in both the server side and the device side is implemented. In the remaining of this section, we give a basic overview of the modules in the functional entity and their interaction with each other. The following symbols are used in this section.

- ✓ N The number of devices.
- ✓ S_n^m The fragment of segment *n* sent to the device *m*, while S_n means the whole segment.
- ✓ L_n The length of the segment *n*.
- ✓ d_n^m The start position of the segment *n* sent to the device *m*.
- ✓ O_n^m The offset of the segment *n* sent to the device *m*.
- ✓ v_n^m The bandwidth of device m at the transmission of segment *n*.
- ✓ T_n^m The tuple (S_n^m, d_n^m, O_n^m) of segment *n* that device *m* share with others.

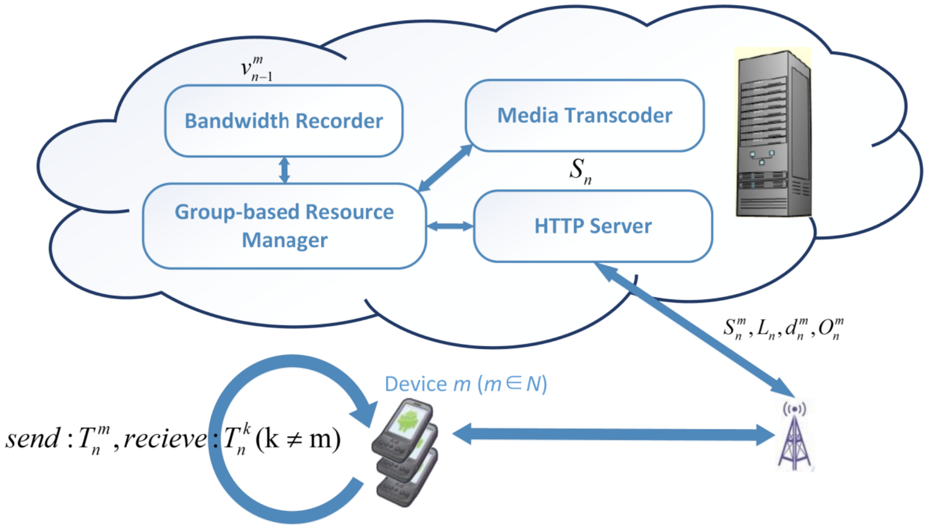

v_{n-1}^{m}
Bandwidth Recorder
Media Transcoder
S_n
Group-based Resource Manager
HTTP Server
S_n^m, L_n, d_n^m, O_n^m
Device m ($m \in N$)
$send: T_n^m, recieve: T_n^k (k \neq m)$

Fig. 1. System architecture

3.1 System Architecture

The system includes a cloud server and several cooperated mobile devices as shown in Figure 1. Before the startup, the devices should have been connected to each other with WiFi-Direct and connected to the server via Internet. And one device of the group should act as the **Captain Device** in charge of coordinating the other devices during the devices-setup period.

3.2 Modules on the Cloud Server

The **Group-based Resource Manager** controls the system on server side. When the **Captain Device** requests the playlist with the group number *N*, the **Group-based Resource Manager** creates a random certification key which is used to recognize the client devices as a group. It also takes the bandwidth information v_{n-1}^{m} from the **Bandwidth Recorder** and lets the **Media Transcoder** produce the video segment S_n in appropriate bit-rate. When connection starts, the length of package and the timestamp will be recorded in the **Bandwidth Recorder** and further provided to the **Group-based Resource Manager**. When predicting the network speed according to the group bandwidth, the highest possible value should using a safety margin as $(1-\alpha)v_{n-1}$ where α is usually in the range [0, 0.05].

The **Media Transcoder** produces the requested video segment with proper bit-rate to fit the network condition of the group of devices. When the devices want to get a video segment, the **Group-based Resource Manager** provides the original video segment and the target group bandwidth to **Media Transcoder**. The **Media Transcoder** outputs the appropriate video segment that can fulfill the group bandwidth capability.

The **HTTP Server** handles the HTTP request from the devices. When a captain device requests the play list, the **HTTP Server** notifies the **Group-based Resource Manager** to create a certification key, and sends the requested m3u8 list along with the key back to the device. When a video segment requests during the playing, the HTTP Server forwards the request to the **Group-based Resource Manager** to deal with.

3.3 Modules on Devices

The device side contains three modules which are **Device Resource Manager (DRM$_m$)**, **Device Downloader(DD$_m$)** and **Device Broadcaster(DB$_m$)**. The **DRM$_m$** takes charge of managing the whole operations on the devices such as segment caching, scheduling with other modules and interaction with Graphical User Interface (GUI). There is a list of segments in the server to be downloaded by each device. The **DD$_m$** cares about the fragment download from the server. If the **DD$_m$** receives the failure message from **DRM$_m$** which is broadcasted from other devices with the failure tuple T_n^m , all the devices try to request the server for failure fragment. Only one device can receive acknowledgement from the cloud and start to initiate the **DD$_m$** to download this failure tuple, where the decision mechanism is based on the wireless channel bandwidth and the download speed of each device. The **DB$_m$** is the coordinator that make all the devices available to connect with each other. The **DB$_m$** can deal with the broadcast message from other devices as well as sending the broadcast message to other devices. Finally when all the fragment of one segment are all received, then the **DRM$_m$** is responsible for assembling all the fragment to one whole segment and dispatch it to the video player.

3.4 Algorithm Procedure

Figure 2 shows the general sequence of flow for the device-setup period. During this period, the **Captain Device** requests the live video streaming playlist with the group number N from the server, and the server returns the list along with a random key which could be the certification of the group member. The **Captain Device** then broadcast the playlist and the key to other devices to start the playing.

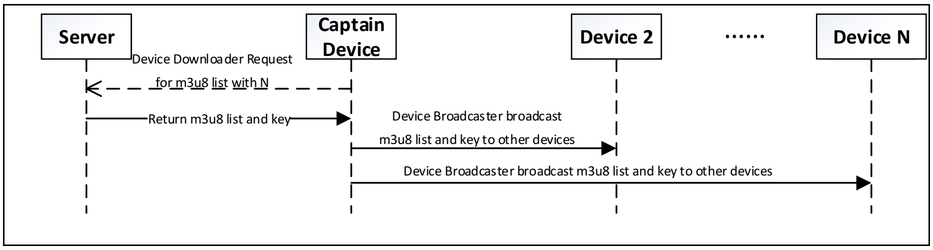
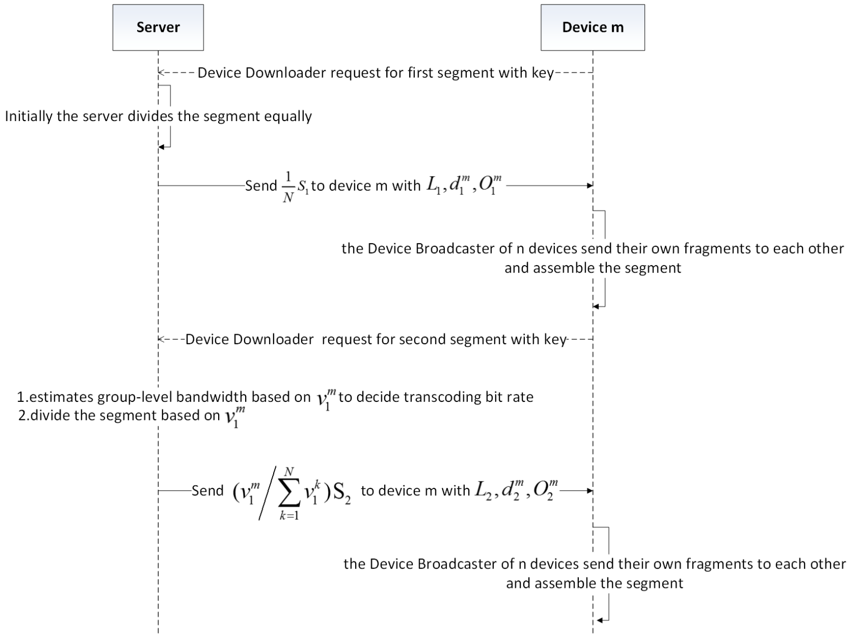

Server
Captain Device
Device 2
......
Device N
Device Downloader Request for m3u8 list with N
Return m3u8 list and key
Device Broadcaster broadcast m3u8 list and key to other devices
Device Broadcaster broadcast m3u8 list and key to other devices

Fig. 2. Flow chart for the device-setup period

After receiving the play list, devices request the live video segments described in the list, as shown in Figure 3. At the first time the **HTTP Server** will simply divide the segment into equal fragments and send them to devices separately. For example, if the number of devices is N. The fragment size of each devices will be $S_1^m = \frac{1}{N} S_1$, where m indicates the mth device, and 1 indicates the first segment. When sending the fragments to devices, the server also provide the segment length L_1, start position d_1^m and offset O_1^m to the mth device so that the devices can record the fragment into appropriate position based on these information. When the **DD_m** receives the fragment, the **DB_m** directly broadcasts the fragment to other devices of the group through D2D interface. When all the fragments are collected, the **DRM_m** will assemble them into one whole segment and send it to the real-time video player. When the **DD_m** sends the request the server for the second segment with the key, the server will predict the network bandwidth of the group, according to the download speed of previous segment. Then the server transcodes the origin video into proper bit-rate to fit the network condition of the whole group, and separates it into fragments with different sizes according to the network ability of each device and send the $S_2^m = (v_1^m / \sum_{k=1}^{N} v_1^k) S_2$ segment to the mth device, where v_1^m indicates the bandwidth of the mth device in the wireless channel at the first transmission. The server also provide the segment length L_2, start position d_2^m and offset O_2^m.



Server
Device m
Device Downloader request for first segment with key
Initially the server divides the segment equally
Send $\frac{1}{N}S_1$ to device m with L_1, d_1^m, O_1^m
the Device Broadcaster of n devices send their own fragments to each other and assemble the segment
Device Downloader request for second segment with key
1.estimates group-level bandwidth based on v_1^m to decide transcoding bit rate
2.divide the segment based on v_1^m
Send $(v_1^m / \sum_{k=1}^{N} v_1^k) S_2$ to device m with L_2, d_2^m, O_2^m
the Device Broadcaster of n devices send their own fragments to each other and assemble the segment

Fig. 3. Flow chart for normal transmission period

4 Testbed Emulation and Performance Analysis

We present an emulated testbed to testify the performance of the designed system, in order to show the advantage in QoE and network utilization. Finally, a PSNR measurement is used to evaluate the quality of the video segment.

4.1 Emulation Environment

Due to the complex smartphone system of the experiment environment and the control over the process, some conditions are set differently from the modules which will not affect the result. We use WiFi as HTTP downloading interface to simulate the 3G/4G connection in real system. Because it is more accurate to control network bandwidth as we want for the emulation in WiFi network than in 3G/4G, in order to repeat the scenarios in emulation. While WiFi module on mobile device is occupied by HTTP downloading, the Bluetooth on the device is used to exchange data between devices in the group. The details of transcoder in cloud are not closely related to the whole system, so we prepared up to 29 fixed streaming rate levels transcoded from an original 5000kbps video stream, in order to simulate the real-time transcoding approximately. The video stream lasts for 200 seconds which is divided into 20 trunk segments. While the minimum and maximum are 200kbps and 3000kbps with a step of 100kbps. And the small deviation towards the real appropriate transcoding will not affect our performance analysis and conclusions.

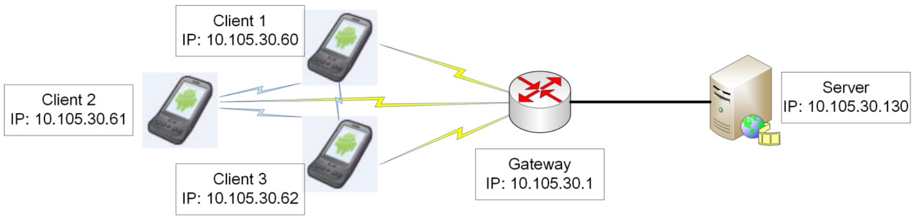
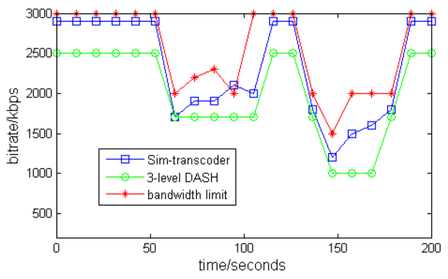


Fig. 4. Emulation environment architecture

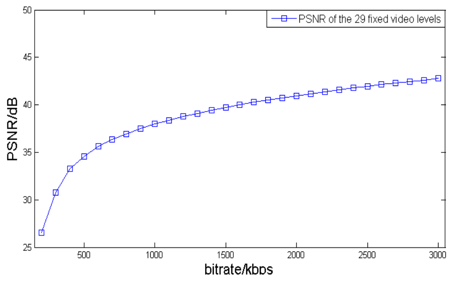
The emulation environment network topology consists of three Android smartphones and a Linux server. Android devices in the system include client 1 as Samsung GT-I9070, client 2 as Samsung GT-I9100 and client 3 as Google Nexus S. The server has an 8 cores Intel Xeon E5-1620 CPU and 16GB RAM, and uses CentOS as its operating system.

4.2 Validation of Scheme on the Cloud Transcoder

This section shows the benefit of more effective network bandwidth utilization by the Media Transcoder. Traditional DASH server usually provides several fixed level of video stream as smooth, standard definition (SD) and high definition (HD) to face the network vibration. While in the transcoder scheme, definition level is more adaptive. Figure 5 (a) shows that using the transcoder scheme, the devices can get a video stream with a proper bit-rate closer to the limitation. The normal device get the DASH video with three fixed level which are 1000kbps, 1700kbps and 2500kbps in this case while the sim-transcoder device use our approach. The comparison indicates the video quality and network utilization with cloud transcoding is obviously improved. The bandwidth limit is emulated with network vibration by the Linux kernel Traffic Control (TC) module.

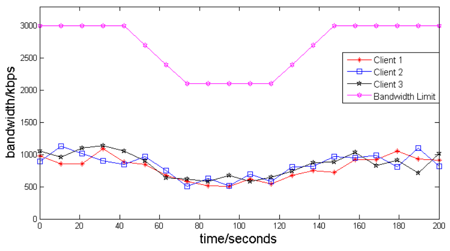


(a) Bandwidth of using normal DASH and Sim-transcoder

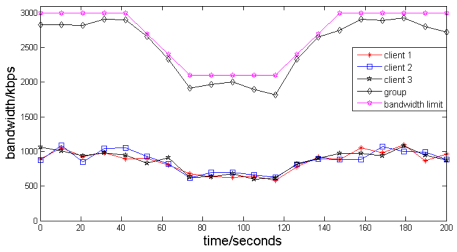


(b) PSNR of the 29 fixed video levels

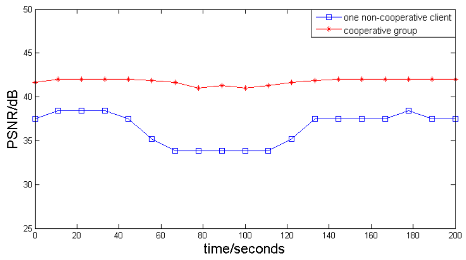
Fig. 5. Performance of the Cloud Transcoder Scheme



(a) Bandwidth usage in the non-cooperative scheme



(b) Bandwidth in the cooperative scheme



(c) PSNR comparison in the cooperative scheme and the non-cooperative scheme

Fig. 6. Bandwidth and PSNR Comparison in the cooperative scheme and the non-cooperative scheme

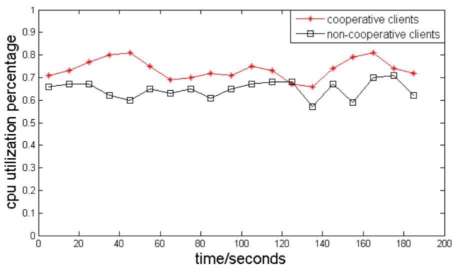
This paper uses the PSNR to measure the video streaming quality by calculating the image pixel difference between the original one and the received one. The computing formula of PSNR is well-known as $PSNR = 10 \times \log_{10}\left(\frac{(2^n - 1)^2}{MSE}\right)$ where MSE stands for the Mean Square Error between the original image and the compared one and n represents the number of bits occupied by each sampling point. The higher PSNR means the less distortion. For a video stream consists of N frames, the PSNR of the video is presented as the average value of the N frames' PSNR. In our case, a Linux open source tool called VQMT is used to calculate the value. Figure 5 (b) shows the PSNR of the 29 fixed video levels compared with the original 5000kbps one.

4.3 Performance of D2D Cooperation

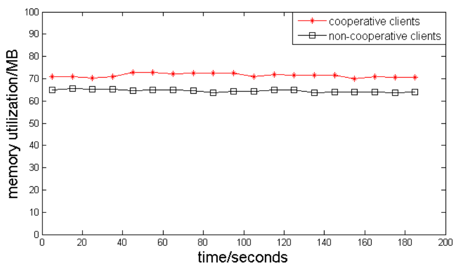
This section describes the emulation result of the cooperative scheme and the non-cooperative scheme. In the non-cooperative scheme, three adjacent devices are watching the same live streaming video and competing for the network bandwidth in the range of same access point. Figure 6 (a) shows that devices in the non-cooperative scheme could own approximately one third of network bandwidth as competitors. As a result every device can only consume the video quality with rate lower than one third of the network bandwidth.

In the cooperation scheme, the requested bandwidth of the three devices is calculated as a group watching a streaming video synchronously. As shown in Figure 6 (b), although each device can only obtain part of the network bandwidth, the acquired video quality achieve better performance and more smooth facing to the network vibration. What's more, the network bandwidth is under higher resource utilization. Figure 6 (c) shows the PSNR between the non-cooperative and cooperative devices as mentioned in Figure 6 (a) and (b). It is observed that the PSNR is increased on average, which present that the media quality viewed by the user is on average better than the original mechanism.

As more component established on the device, more resource of computation and memory storage is cost. Figure 7 shows the CPU utilization and the memory usage of the each device in the cooperative scheme and the non-cooperative scheme. The CPU utilization of one device in the cooperative scheme is average 10 percent higher than those in the non-cooperative scheme, mainly because of the broadcasting mechanism. The same 10 percent higher usage also happens in the memory of the device.



(a) CPU utilization



(b) Memory usage

Fig. 7. Resource usage of cooperative and non-cooperative

5 Conclusions and Future Work

In this paper, we have proposed an optimization solution for high resolution streaming video over HTTP in mobile network. We considered the scenario that a set of neighboring mobile devices take interests in watching the identical video stream. We have taken a pragmatic stance to work within the network constraints (w.r.t. resource sharing and adaptation) that have spurred the growth of video traffic by the help of mobile cloud computing. We proposed a framework to improve the quality of video services with the MCC and D2D technology. This article designs a cloud-assisted real-time transcoding mechanism based on DASH protocol, implements the bandwidth recorder, a media transcoder, a group-based resource manager and a HTTP live streaming server, and provides the optimum media quality. The experiments on our emulated network with Android smartphones show that this framework provides higher resolution live streaming with less expense, better QoE and more effective network utilization. According to the experimental results of the implemented testbed, the bandwidth utilization rate can be higher than 80 percent when the bandwidth is in steady state, even if the bandwidth is unstable, the bandwidth utilization rate is maintained at 60 percent - 80 percent. The PSNR analysis also shows that the video stream quality increased even with the network vibration due to the mechanism.

Fairness, efficiency, and stability are three potentially conflicting goals that a robust adaptive bit-rate selection algorithm must strive to achieve the Quality of Experience. In the future, we will try to present a principled understanding of bit-rate adaptation of DASH service in mobile cloud computing environment and analyze through at least three main components: bandwidth estimation, bit-rate selection, and chunk scheduling.

References

1. Pande, A., Ahuja, V., Sivaraj, R., et al.: Video delivery challenges and opportunities in 4g networks. IEEE MultiMedia **20**(3), 88–94 (2013)
2. Huang, T.Y., Handigol, N., Heller, B., et al.: Confused, timid, and unstable: picking a video streaming rate is hard. In: Proceedings of the 2012 ACM conference on Internet Measurement Conference, pp. 225–238. ACM (2012)
3. Lai, Y.X.U.N., Wan, J.: Cloud-Assisted Real time Transrating for HTTP Live Streaming. IEEE Wireless Communications 3 (2013)
4. Seferoglu, H., Keller, L., Cici, B., et al.: Cooperative video streaming on smartphones. In: 2011 49th Annual Allerton Conference on Communication, Control, and Computing (Allerton), pp. 220–227. IEEE (2011)
5. Al-Kanj, L., Dawy, Z.: Optimized energy efficient content distribution over wireless networks with mobile-to-mobile cooperation. 2010 IEEE 17th International Conference on Telecommunications (ICT), pp. 471–475. IEEE (2010)
6. Yaacoub, E., Dawy, Z., Abu-Dayya, A.: On real-time video streaming over LTE networks with mobile-to-mobile cooperation. In: 2012 19th International Conference on Telecommunications (ICT), pp. 1–6. IEEE (2012)

7. Abedini, N., Sampath, S., Bhattacharyya, R., et al.: Realtime streaming with guaranteed QoS over wireless D2D networks. In: Proceedings of the fourteenth ACM International Symposium on Mobile ad Hoc Networking and Computing, pp. 197–206. ACM (2013)
8. Sprintson, A., Sadeghi, P., Booker, G., et al.: A randomized algorithm and performance bounds for coded cooperative data exchange. In: 2010 IEEE International Symposium on Information Theory Proceedings (ISIT), pp. 1888–1892. IEEE (2010)
9. ParandehGheibi, A., Médard, M., Ozdaglar, A., et al.: Avoiding interruptions—a qoe reliability function for streaming media applications. IEEE Journal on Selected Areas in Communications **29**(5), 1064–1074 (2011)
10. Sodagar, I.: The mpeg-dash standard for multimedia streaming over the internet. IEEE MultiMedia **18**(4), 62–67 (2011)
11. Wang, X., Kwon, T.T., Choi, Y., et al.: Cloud-assisted adaptive video streaming and social-aware video prefetching for mobile users. IEEE Wireless Communications 20(3) 2013
12. Li, Z., Huang, Y., Liu, G., et al.: Cloud transcoder: Bridging the format and resolution gap between internet videos and mobile devices. In: Proceedings of the 22nd International Workshop on Network and Operating System Support for Digital Audio and Video, pp. 33–38. ACM (2012)
13. Zhou, C., Lin, C., Zhang, X., et al.: A Control-Theoretic Approach to Rate Adaption for DASH over Multiple Content Distribution Servers (2013)
14. Li, Z., Begen, A.C., Gahm, J. et al.: Streaming video over HTTP with consistent quality. arXiv preprint arXiv:1401.5174 (2014)