June 2020

Eliminating Redundant Media Uploads Using A Media API

Thomas Deselaers

Victor Carbune

Eliminating Redundant Media Uploads Using A Media API

ABSTRACT

Users that share media such as pictures or videos via smartphones or other devices with cellular data connectivity often end up reuploading the same media via duplicate channels, e.g., multiple instant messaging apps, email, photo gallery, etc. This is costly and can cause excess consumption of the user's data plan. This disclosure describes a media API, implemented in a device operating system, that reduces such redundant media uploads. When a user attempts an action that requires media upload via an application, the application invokes the API to determine whether the media was previously uploaded via another application. If a prior uploaded copy of the media is available, the application utilizes the copy to fulfill the user request, thus eliminating the redundant upload action.

KEYWORDS

- Cloud backup
- Data plan
- Upload cost
- Cellular data
- Media upload
- Photo sharing
- Video sharing
- Instant messaging
- Photo gallery

BACKGROUND

Taking pictures and videos using smartphones and other devices with cellular data connections is a common activity. Many users opt for cellular data plans that include a limited amount of data transfer. For such users, uploading pictures or videos taken with their device via the cellular data connection, e.g., for sharing with other users, for backup, etc. can consume a substantial portion of their data transfer quota.

This problem is made worse by the fact that in some cases pictures and/or videos are uploaded multiple times. For example, if the user sends the same picture via an instant messaging or chat application and via email, the picture is uploaded twice, once for each service. Further, if the user has configured cloud-based backup of their picture library, the picture is also automatically backed up to cloud-based storage, causing a third upload.

To limit data usage, many users disable automatic backup of pictures or videos over the cellular data connection. However, this does not reduce the data consumption that occurs when the user sends the same picture or video multiple times, e.g., via different IM services, email, rich messaging, or social network applications.

DESCRIPTION

This disclosure describes a service, made available via an application programming interface (API) to reduce the redundancy in uploads of pictures or videos, thus saving data plan consumption. Per the techniques described herein, every picture or video that is captured by a device camera (e.g., a smartphone or tablet camera, a camera of a wearable device, etc.) is identified and is uploaded only once. For example, the upload may be performed by a default picture gallery or management application (or any other application that the user designates as

the default) to cloud-based backup. The default application is configured to make the uploaded picture or video available to other applications on the user device.

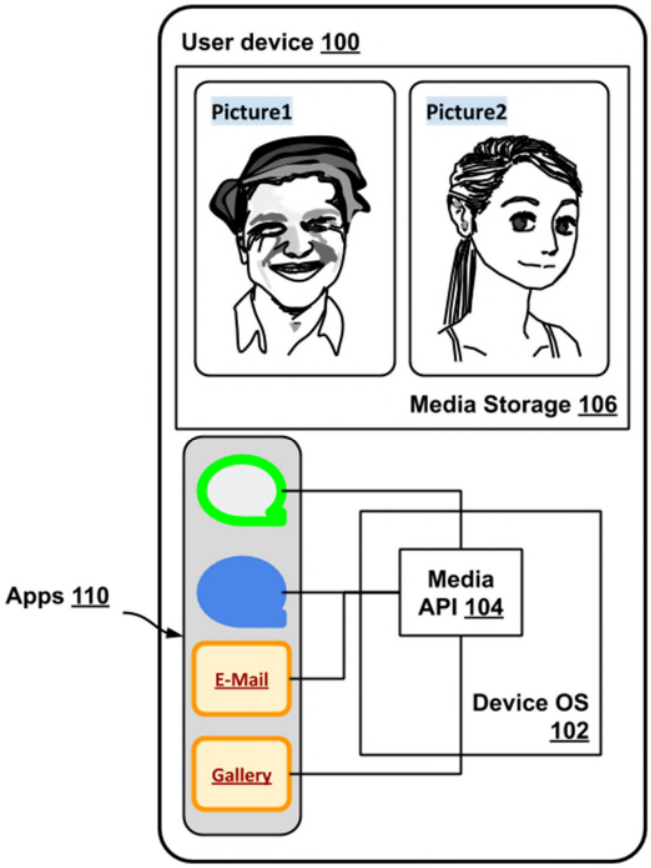

Fig. 1: User device with non-redundant media upload API

Fig. 1 illustrates an example operational implementation of the techniques described herein. A user captures pictures (Picture1, Picture2) using a camera of user device (100) that runs an operating system (Device OS 102) that includes a media API (104) that implements the techniques described herein. The captured pictures are stored in media storage (106) of the user device. The user device includes various applications (110), e.g., messaging apps, email application, and a photo gallery, as seen in Fig. 1.

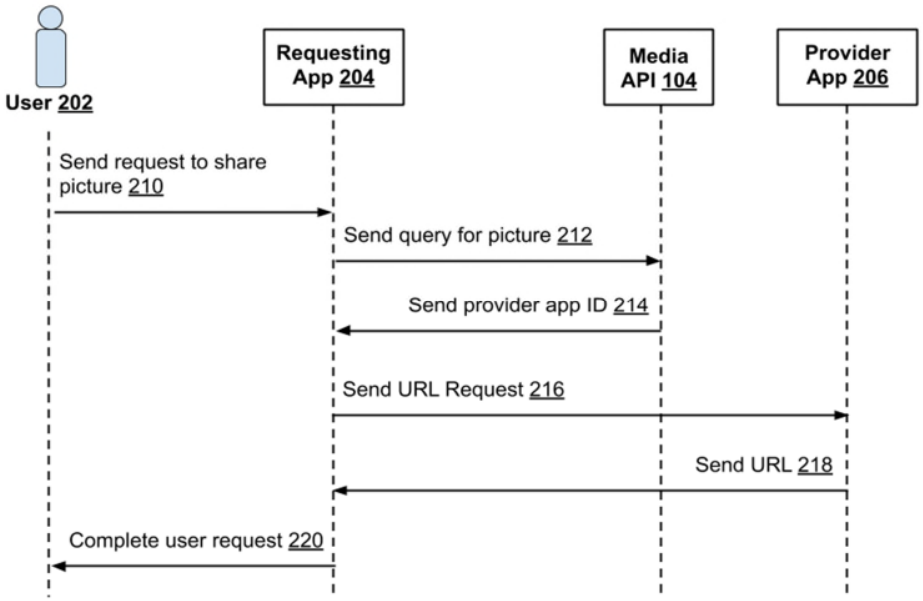

Fig. 2: Media upload API functionality

Fig. 2 illustrates the functionality of the media API. The media API enables applications that utilize the media API to provide indications of their capabilities to other applications. For example, one application may store pictures or videos and share those with other applications, while another application may only access pictures that are already stored in the media storage.

When a user (202) requests to share a picture (210) stored in the media storage via a particular application, e.g., an instant messaging (IM) application, the requesting application (204) generates a request (212) for the picture to the media API (104).

1. The media API receives a query (212) from a requesting application that attempts to use a picture stored in media storage, e.g., to upload the picture, and provides a response (214) indicative of the availability of the image from a provider application that has already uploaded the image. The media API can identify the picture based on the filename (provided by the device operating system) that uniquely identifies pictures. The below pseudocode illustrates such an API call:

```
// App announces that it wants to consume the image that is locally
   available under the given filename.
ImageConsumptionInformation wantToConsumeLocalImage(File filename);
// Returns: Information about the availability of the image
class ImageConsumptionInformation {
     List<ImageProvider> imageProviders;
  };
class ImageProvider {
     Id imageProviderId;
     boolean willUploadInAnycase;
     enum {SMALL, MEDIUM, LARGE, FULL} imageResolution;
     };
```

2. The requesting application receives the identity of the provider application (206) that has either already uploaded the picture or has queued the upload. The requesting application then queries the provider application (216) for a link (e.g., URL) for the picture. The provider application provides a response (218) that includes the link. The response is provided immediately if the picture upload is complete and the URL is available, or asynchronously after the picture upload has been completed. The below pseudocode illustrates a request for a picture:

```
URL requestImageConsumption(File filename, enum
   imageResolution);
```

3. The requesting app then completes (220) the user request to share the picture. If the uploaded picture available via the link is not of a sufficient resolution or has other parameters that do not match the user request, or if the picture has not been uploaded (no provider application available), the requesting application can upload the picture itself and indicates to the media API that it can serve as a provider application for the picture for subsequent requests.

Additionally, other identifiers such as a file descriptor via a hashing function can also be used. In some cases, pictures uploaded to cloud storage of a particular application may be different from the original picture, e.g., edited by application of filters, cropping, or other effects. In such cases, the media API can provide functionality that indicates that the previously uploaded version is an edited version. The requesting application can prompt the user to indicate whether to reshare the edited version or to reupload the original.

Thus, the media API reduces redundancy in media uploads by enabling reuse of previously uploaded media. For any media (picture, video, etc.) that is captured by a device camera, a particular application can be set as the default for upload of the media and the particular application can then make the media available to the other applications. In this manner, the media is uploaded only once using the mobile carrier data channel. Other applications can access the media via the cloud and synchronize independently, without requiring reupload of the media. The media API can thus save data plan consumption, on cellular, WiFi, or other types of network connections.

Example of use

A user Jane has just taken a picture using her smartphone and she attempts to share the picture with her friend Serena via an instant messaging application on the smartphone. The IM app sends a request via the media API that indicates that it has received a request to upload the picture. The other applications on the phone receive the request and can respond whether these applications have already uploaded or are likely to upload the picture. For example, a photo gallery application can respond to the request indicating that it is in the process of uploading (or has already uploaded) the picture in full resolution and that it can make the picture available to the IM app. The IM app then receives the URL from the photo gallery application which then

sends the image to Serena. When Jane attempts to send the same picture to other recipients, e.g., via email, via other IM apps, etc. the respective application can utilize the media API to share the picture, without reuploading.

Further to the descriptions above, a user may be provided with controls allowing the user to make an election as to both if and when systems, programs or features described herein may enable collection of user information (e.g., information about a user's installed or available applications, media library, a user's preferences), and if the user is sent content or communications from a server. In addition, certain data may be treated in one or more ways before it is stored or used, so that personally identifiable information is removed. For example, a user's identity may be treated so that no personally identifiable information can be determined for the user. Thus, the user may have control over what information is collected about the user, how that information is used, and what information is provided to the user.

CONCLUSION

Users that share media such as pictures or videos via smartphones or other devices with cellular data connectivity often end up reuploading the same media via duplicate channels, e.g., multiple instant messaging apps, email, photo gallery, etc. This is costly and can cause excess consumption of the user's data plan. This disclosure describes a media API, implemented in a device operating system, that reduces such redundant media uploads. When a user attempts an action that requires media upload via an application, the application invokes the API to determine whether the media was previously uploaded via another application. If a prior uploaded copy of the media is available, the application utilizes the copy to fulfill the user request, thus eliminating the redundant upload action.