Efficient Data Distribution and Sharing in Mobile Cloud Computing

CHAITANYA VEMULAPALLI, SANJAY KUMAR MADRIA
Department of Computer Science
Missouri University of Science and Technology, Rolla, MO
and
MARK LINDERMAN
AFRL, Information Directorate, Rome, NY

The combination of Mobile computing and Cloud computing gave rise to a new computing paradigm called Mobile Cloud Computing (MCC) that gives the flexibility to access information and computing resources anywhere anytime. In many applications, mobile nodes capture images/video clips and exchange with other mobile peers (on demand) and use local access points for efficient smooth distribution of information in the wireless network. In such a situation, a three-way data management and dissemination technique is helpful because it provides both data management and distribution at different levels of granularity. The main motivation of this paper is that we seek a balance between accessing information from a remote cloud server, at the cost of increasing latency, and accessing data from other mobile hosts, at a cost of power, bandwidth and increased traffic. To manage this balance, we propose a layered architecture supported by mobile hosts, access points and cloud together for efficiency. The MCC architecture described in this paper also provides fault tolerance in case one of the mobile nodes fails or gets disconnected. In this paper, we discuss a layered architecture for MCC and present a data pre-distribution scheme for efficient sharing of data among potential users in MCC environment. We also propose update propagation and cache replacement policies in MCC. In addition, we report complexity analysis for cache accesses by mobile nodes using the proposed architecture in terms of communication messages. We implemented the proposed schemes using EC2 and mobile devices and evaluated the performance of accessing the data from different levels of architecture.

Keywords: mobile cloud computing, data pre-distribution, data access, latency

1. INTRODUCTION

Over the past few years, development of mobile wireless technology has increased at a rapid pace with introduction of sophisticated features like built-in cameras, improved processing capabilities, increased storage capacity, better communication capabilities, etc. But these developments have failed to keep pace with the software demands for storage and processing. Coupling of mobile computing with cloud computing is envisioned as a solution to these constraints in mobile computing. Significantly, in the last decade, mobile Internet access has become ubiquitous, even if some research problems pertaining to efficiency of wireless networks remain. Thus, the foundation is laid for considering how the processing and storage needs of mobile devices can be exported to the cloud servers [Juniper 2010; ABI 2009] while the mobile devices retain only a thin client to display results or files. Two good examples, of such thin clients are the YouTube and Facebook apps in smart phones. Figure1 represents the general architecture of Mobile Cloud Computing.

The rapid development of mobile devices has opened up possibility of new applications like location-based services, information sharing, etc. Location-based services which previously had only basic information about a service can now include more details like photographs, videos, detailed description of those services tagged with geo-locations, etc. Similarly, information shared among mobile users can also be more detailed. For example, a person who likes a restaurant in particular location may take photographs, video clipping of the restaurant, write a brief review

This work is supported by a grant from AFRL, Rome.

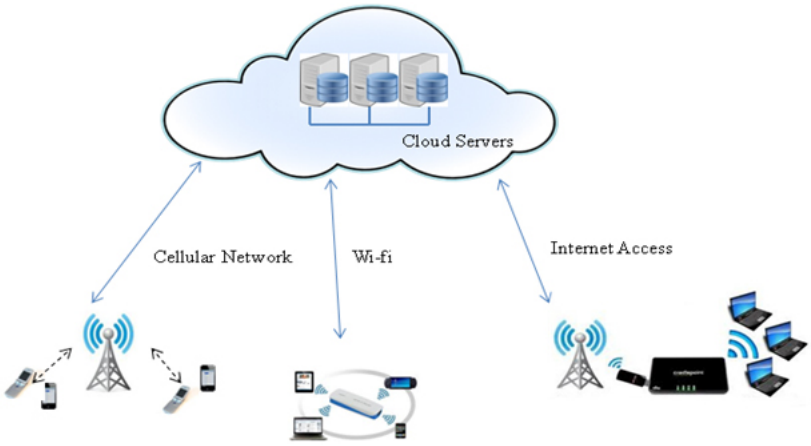

Figure 1: General Architecture of Mobile Cloud Computing

of the restaurant and upload them to the cloud. Another mobile user who is in same location or coming to same location and looking for a good restaurant, may find this information very helpful. Availability of such detailed information can lead to other applications such as one explained in [Satyanarayanan. 2010]. In [Satyanarayanan. 2010], Satyanarayanan et al. mentioned how mobile devices with cameras can also be used to remap an entire location during disaster recovery. What all of these applications have in common is that they are data-intensive. All these applications of mobile cloud computing involve uploading and downloading information from the cloud. So, latency becomes an important performance parameter when dealing with images and video clips.

In this paper, we propose a layered architecture for MCC and design a data pre-distribution scheme to reduce the latency involved in sharing information through cloud resources. Our focus is on the management of image files, video files and text files rather than the specific applications that produce or consume them. In our proposed pre-distribution scheme we propose ideas to reduce the latency in accessing information from cloud by caching data and metadata in the access points, in the cloud, and with potential users. The nodes where metadata is cached and nodes where actual data is cached are determined based on respective user's probability of accessing that information. Since, all the data is location tagged, proximity of nodes to that location is also taken into consideration while determining the probability of a node accessing that data. We have implemented the proposed architecture using EC2 and performed the experiments to evaluate the performance of the protocol.

The rest of the paper is organized as follows. Section 2 discusses some of the previous works related to mobile cloud computing. Section 3 explains our proposed architecture. Section 4 explains proposed data pre-distribution scheme. Section 5 presents the data access algorithm in the proposed architecture. Section 6 presents the update propagation mechanism. Section 7 discusses the cache replacement policy. Section 8 details the performance evaluation and finally Section 9 concludes the paper and finally, section 8 concludes the paper. [wikipedia]

2. RELATED WORK

Mobile Cloud Computing [Hoang et al. 2011] is an emerging technology with various research problems still being explored. Qi et al [Han and Gani 2012] discussed the various trends in mobile cloud computing and research directions. In [Bisdikian et al. 2011], authors discuss data management issues in mobile cloud computing. In [Satyanarayanan. 2010], the author discussed about some of the possible applications of mobile computing in future. It also discusses about the extended architecture of mobile cloud computing and the effective design strategies.Stuedi et al [Stuedi et al. 2010] presented a scheme called'Wherestore' for caching data in mobile cloud

computing. Their idea is to use the past location history to predict the possible future locations of a mobile node and cache the data related to that location. Transition graphs were used for prediction of future location from a current location. In [Chen and Itoh 2010] Chen et al proposed a new architecture for mobile cloud computing to increase the battery life of the physical mobile node. Each mobile node has a corresponding virtual smart phone in the cloud. Their main idea is to save the battery life of mobile device by exporting processing actions to virtual smart phone in the cloud.In another initial work [Samimi et al. 2006], concept of Mobile Service Clouds with dynamic instantiation, composition, configuration, and reconfiguration of services on an overlay network to support mobile computing was presented. In [Kosta et al. 2013], the authors proposed a similar approach to offload data processing of mobile apps to software clones in the cloud to extend the battery life. They implemented a distributed P2P platform called Clone2Clone(C2C) for software clones in the cloud for content sharing, distributed execution, etc. A detailed study of feasibility and associated costs of offloading computation and data storage to cloud resources is performed in [Barbera et al. 2013]. Here also, a similar kind of architecture is used where in each mobile device is associated with a software clone in the cloud. Two types of software clones are considered in their experiment; off-clone to support computation offloading and the back-clone for data backup. Their results showed that the overhead in terms of bandwidth and energy costs is reasonable to sustain the mobile cloud computing. But in the process of offloading computation and storage to cloud, the mobile apps may consider the cloud as another remote support for mobile devices. [Barbera et al. 2013] proposed an approach where cloud will be considered as another resource of same physical device. They achieved by developing a system called CDroid that resides partially on the mobile device and partially on the cloud making it look like part of the device for the mobile apps.Work in [Kosta et al. 2013] was further extended in [Kosta et al. 2013] and a secure real time collaboration system was implemented on top of C2C for smartphone users. Results showed improved energy savings and response time compared to SPORC [Feldman et al. 2010] that runs completely on smartphones.Most of these related works emphasis is on the energy savings that can be obtained by offloading the computation and storage to cloud. In this paper we focus on addressing the latency issues that would be involved in accessing the data stored in cloud.

3. MOBILE CLOUD ARCHITECTURE

In our architecture, each mobile device collects information from its surroundings and has a dedicated Virtual Image (VI) hosted on the cloud. A Virtual Image performs the computation and other actions on behalf of its physical mobile device using the cloud resources. The location of mobile device is periodically reported to its Virtual Image in the cloud. Based on requirement, mobile host may retain a copy of a particular data in its local storage even though it is present in the Virtual image.The idea of having virtual instance of mobile host in cloud to handle the processing requirements was proposed by Chen et al in [Chen and Itoh 2010]. Figure 2 provides a high level view of the proposed architecture.

Each physical location is defined in terms of a reference point (latitude and longitude) and an area of radius 'k' distance around it called its Coverage Area (CA). In the cloud, sets of VIs are grouped together by Coverage Areas that encompass the corresponding physical devices. Multiple access points may serve a Coverage Area. In this scheme our emphasis is on physical location; we are not using the network infrastructure (access points) to identify the physical location but instead using the actual physical location itself. Since each Coverage Area extends up to a distance of 'k' from its reference point, there will be overlapping regions to cover every piece of physical land. So, there is a chance that an access point may exist in overlapping region of two or more Coverage Areas (CAs). In such cases where an access point falls in overlap region of two or more CAs, and is at equal distance from the reference points, it will be taken as a member of the Coverage Area which has fewer number of access points. Figure 3 provides a pictorial representation of this. The VIs corresponding to physical devices under that access point, also

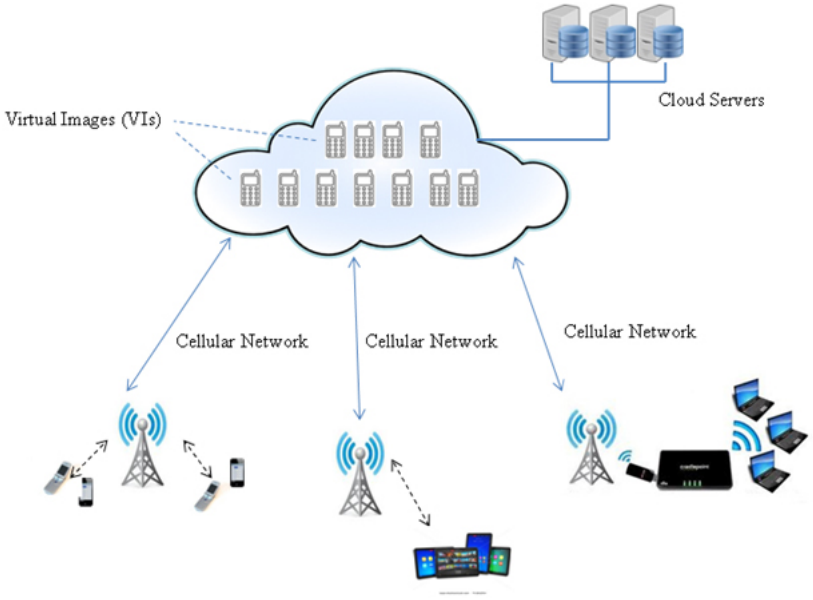

Figure 2: High-level Architecture

become member of that group in the cloud. The grouping information and IP addresses of other access points of the group are provided to access points from the cloud. Every access point thus has the information about its group and the group members.

In the cloud, each Coverage Area is assigned a Master Virtual Image (MVI). It contains a copy of the data that belongs to that location/Coverage Area. Since the nodes involved here are mobile nodes, they constantly move from one location to another. When a mobile host moves from one Coverage Area to another Coverage Area, its corresponding VI migrates to another group. Metadata of content present with this VI is deleted from MVI of old group and updated with the MVI of the new group. Thus, MVIs constantly maintain data that belongs to that location and the metadata of content present with each of the VIs under its jurisdiction.

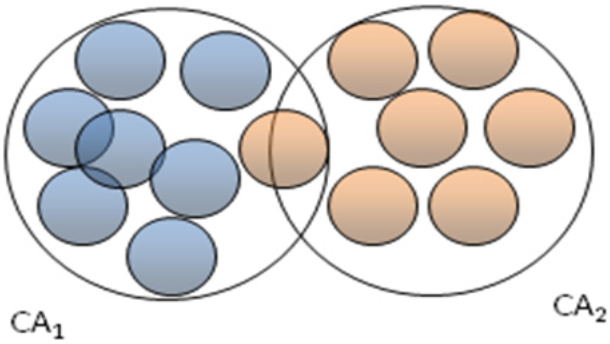

Figure 3: Grouping of Access points

Figure 4 gives pictorial representation of various components inside the cloud network. The front-end server receives incoming data messages from the access points and segregates the data messages from the location information. The data messages are updated with the respective VIs and location information is sent to Location Tracker. Location Tracker maintains the location information of all the physical devices. This information from Location tracker is used

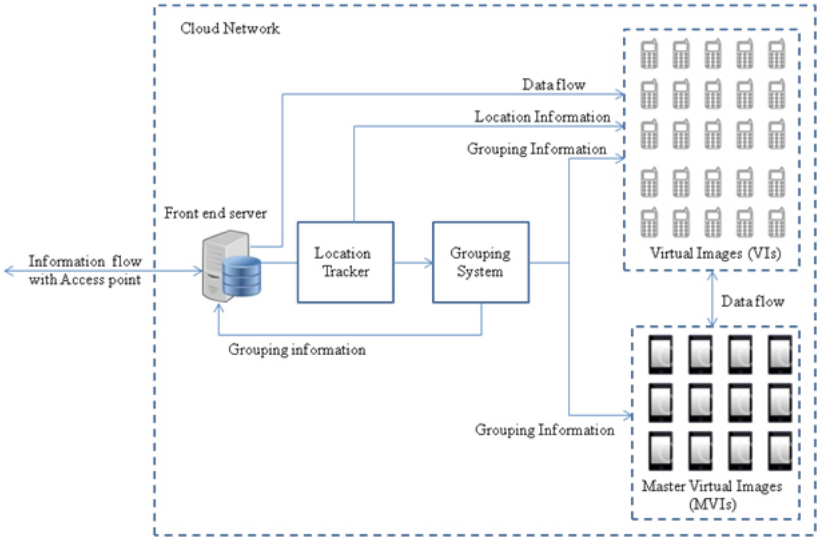

Figure 4: Architecture inside the cloud network

by the Grouping system to manage the grouping of VIs and MVIs dynamically. The grouping information is also feedback to the frontend server to be forwarded to the access points.

4. DATA PRE-DISTRIBUTION

As discussed in the introduction, we are only considering image files, video clips and text files. Since our architecture involves mostly wireless networks, the data-sharing scheme should address the bandwidth limitations. In our proposed scheme, we handle this by leveraging the metadata of the files. Metadata includes information like owner, date of creation, geo-tagging, etc. Present day digital cameras are capable of creating images and video files with this sort of metadata. Even though different manufacturers follow different standards of metadata, the basic information like owner, geo-tagging, date of creation, etc. are common in all. In our scheme, we use this basic metadata to overcome the size constraint. The various phases of data pre-distribution are explained below.

4.1 Creation of new data file

This is the first phase in data pre-distribution process. As soon as the new data file (Image, video or text file) is created by the mobile host, it is automatically uploaded to its Virtual Image (VI) in the cloud, along with its metadata. The VI then copies the data file and its metadata to the Master Virtual Image (MVI) of that CA. During this process, metadata of that file is also copied to the access point of the mobile node. The entries at every access point in a particular CA eventually become consistent. So, when metadata of a new file is uploaded at an access point, the access point broadcasts the new record to all other access points in its group via IP multicast. It waits for the acknowledgements from all the group members. We introduce a new term called Consistency factor (cf) to measure the consistency of content among access points in a Coverage Area. Access point repeats the operation until the consistency factor (cf) is greater than an application defined threshold 'γ' where consistency factor (cf) is defined as

$$(cf) = \frac{No of acknowledgements}{No. of access points in the group} \quad (1)$$

4.2 Pre-distribution of data

Caching of data items is essential to improve the performance of the system. Local caching of data by the VIs in the cloud reduces the time required to serve a data request and also helps

when devices are off-line. An efficient caching scheme needs to replicate/pre-distribute a data file to the nodes/users that have high probability of accessing the file in the future. Identification of these nodes is the key part in the proposed pre-distribution scheme. Nodes/Users that may be interested in the data files related to a particular location are:

(1) Users currently in that location/CA: As image files and video files are location specific, nodes/users belonging to that location are potential users.
(2) Users who are moving to that location: Users travelling to a particular location will be interested to have information about that location to plan their trip ahead.
(3) Users who previously accessed data from that location: Users who have frequently accessed information about a location has high probability of accessing it again.
(4) Previously accessed files from that node: Sometimes, users may be interested in data being uploaded by a particular user like a restaurant, coffee shop, etc. Hence, users who accessed information from a node frequently in the past are also potential users.

In order to identify the above mentioned four sets of potential users, the location tracker and VIs together maintain the following four lists:

(1) (L_{fv}): List of all the VIs whose respectivemobile hosts visited a particular location/Coverage Area in the time period 'T'.
(2) (L_{ds}): List of all the VIs, which accessed data, related to a particular location/Coverage Area in Time period 'T'.
(3) (L_{du}): List of all the VIs, which specifically queried for data from a particular user in Time period 'T'.
(4) (L_{ca}): List of all VIs whose respective mobile hosts are currently within the Coverage Area.

Though these lists contain the potential users of a particular data, they could be very large depending on the density of population in those locations. Pre-distributing a new data file to all these VIs in the cloud would be inefficient. Moreover, the constant movement of mobile nodes and the associated change in group configurations in the cloud would make this kind of replications a major communication overhead among VIs. So, in our proposed scheme we filter the lists to obtain a more probable list of users and also employ a two level replication scheme to improve the efficiency. In first level, only the metadata is replicated at VIs having good probability of access. In the second level, actual data is replicated to few VIs with highest probability of accessing that data. Identification of VIs for first and second levels is done as explained in next two subsections.

4.2.1 *Pre-distribution of metadata.* In the first level of replication, to pre-distribute the metadata of a new data file, nodes are selected from each of the four lists mentioned in the previous section. From each list, set of nodes (S) whose frequency is greater than the average (A) in that list is selected by location tracker.

(1) (S_{fv}): Set of VIs whose mobile hosts frequency of visit to the location is higher than average.

$$S_{fv} = \{X_{fv_i} \in L_{fv} : X_{fv_i} > A_{fv}\} \tag{2}$$

(2) (S_{ds}): Set of VIs whose number of requests for data from this location is higher than average.

$$S_{ds} = \{X_{ds_i} \in L_{ds} : X_{ds_i} > A_{ds}\} \tag{3}$$

(3) (S_{du}): Set of VIs whose number of requests for data from this source node is higher than average.

$$S_{du} = \{X_{du_i} \in L_{du} : X_{du_i} > A_{du}\} \tag{4}$$

(4) (L_{ca}): List of VIs whose mobile hosts are currently within the Coverage Area.

Union of these sets without duplicates is considered to be the pre-distribution set to which the metadata is replicated.

$$R_{md} = (S_{fv} \cup S_{ds} \cup S_{du} \cup L_{ca}) \tag{5}$$

4.2.2 *Pre-distribution of actual data.* In the second level of replication, actual data is replicated to set of nodes, which have highest probability of requesting the data. In the previous sub-section, metadata is replicated to set of nodes whose frequency value is higher than the average among that list. But these nodes could be outside the Coverage Area or inside Coverage Area. If, the node is inside the Coverage Area and near to that location to which the data file belongs to, then there is a greater probability that the user would be interested in that data file. In such cases, actual data should be cached with the VIs of those nodes for quick access to the data. Such nodes are identified from the four sets of nodes described in the previous subsection. VIs to which we replicate actual data are identified as

$$R_{ad} = (S_{fv} \cap L_{ca}) \cup (S_{ds} \cap L_{ca}) \cup (S_{du} \cap L_{ca}) \tag{6}$$

—$(S_{fv} \cap L_{ca})$ = High frequency visitors who are currently within Coverage Area.

—$(S_{ds} \cap L_{ca})$ = Users who access data of this location previously and are currently within Coverage Area.

—$(S_{du} \cap L_{ca})$ = Users who accessed data from this source node previously and are currently within Coverage Area.

This also achieves an increased granularity in the pre-distribution scheme. Set of nodes with high probability of accessing data and far away from origin location of file are replicated with only metadata. But if the node is inside the origin location (CA) of the data file it is replicated with actual data file.

4.2.3 *Proactive data caching at each access point.* The two level scheme for replication of data among VIs in the cloud is expected to reduce the time required to access data. This can be improved further by caching highly accessed data closer to user. Some of the data items that are highly requested in a particular location (CA) can be pushed to access points in that CA. In this way, requests for those data items can be served at the access points itself instead of being served from cloud level. Each access point stores hot data of that CA and two sets of metadata:

—MD0 = Metadata of all the content within itsCoverage Area (CA) including that of hot data stored with it.

—MD1 = Metadata of neighboring Coverage Areas(One-hop)

At every time period T_r, a snapshot of each CA is taken. At this instant, the access points upload the access logs of various data items, which were served at access point itself to the MVI. MVI of a CA computes data access frequencies for all data items (including replicas) in that group in that snapshot and creates a list H_d in the descending order of access frequency. So, the items at the top of this list are the hot data or most accessed data in this Coverage Area. Let d_{si} denotes the size of each of data item d_i and let N denote the no. of data items from the top of the list H_d such that $\sum_{i=1}^{N} d_{si} M_c$ where M_c is the size of memory storage at each access point dedicated to contain hot data. These N data items are pushed to all the access points in that CA. After this, all the one-hop access points exchange the metadata MD0 present with them. But only the access points on the peripheral will have contact with access point from another CA, which have different metadata. This adjacent CA metadata is then propagated to all the members of the group. So, all the access points store MD0 metadata of finite numberof neighboring CAs as MD1.

When a mobile host (MH) moves from one CA to another CA the peripheral access point passes on the metadata of the content present in the MH to the new cell tower along with handover

information. The receiving access point again broadcasts this new metadata to other access points in its group. Thus, at any given instant of time every access point has metadata of all the content in its Coverage Area.

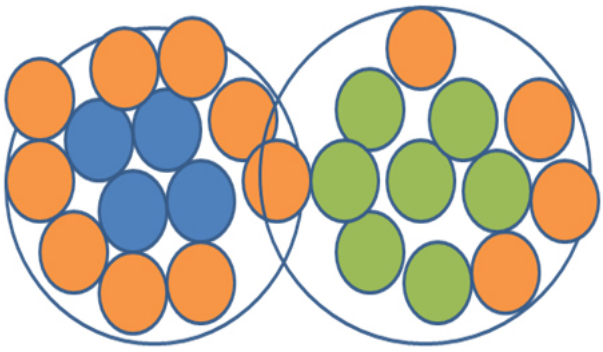

Figure 5: Peripheral Access points in CAs

5. DATA ACCESS

In this architecture, data requests can be served at two places. It can be served at access point level or at VI level in the cloud. At access point level, the request could be served from the hot data or through formation of MANETs among MHs. Another assumption we make here is that the user can choose whether he wants the actual data itself or he is ready to receive device-id, which is containing the requested data.Following subsections present the data access algorithms at access point and at cloud level.

5.0.4 *Access point receives request from mobile host.* User of mobile host MH_Q requests for data item $'d'$ by placing query (Q_d) with the access point. The data access algorithm at access point takes this user query (Q_d) as input and returns the requested data item $'d'$ or device id containing the data item based on set of conditions. According to the algorithm, hot data of the access point (AP_{hd}) is first searched for the requested data item. If data is found in hot data, the data item is returned to the mobile host MH_Q. Else, if a match is found for metadata of requested data (M_d) in MD0 and the userś choice $U_c =' Y'$ i.e user is ready to receive device id of mobile host ID_{MH} containing the data item$(D_{MH} \ni d)$, then the nearest device id is returned. Then the requesting mobile host can form a MANET with that device and retrieve the data. Else if a match is found for metadata of requested data in MD0 but user's choice is $U_c =' N'$ i.e. user is not willing to receive the device id, then the access point downloads the data from virtual image (VI_{MH}) of mobile host containing the data item $(D_{MH} \ni d)$. Else if a match is found for metadata of requested data (M_d) inmetadata of neighboring Coverage area $(MD1_{CA_x})$ then the query is forwarded to access point of that Coverage Area CA_x and the data is retrieved from there. Later, the retrieved data item $'d'$ is forwarded to the requesting mobile host. If no entry is found with access point in any of the three categories, then the query is forwarded to the corresponding virtual image (VI_Q) in the cloud. Figure 6 provides the algorithm for data access at access point when the request is received from a mobile host in its cell site.

5.0.5 *Access point receives a request from another Access point in neighboring Coverage Area.* As stated in the previous case, if the metadata of the requested data item is found in MD1 i.e. metadata of neighboring Coverage Area, the query is forwarded to that CA and data is retrieved. So, a peripheral access point can receive a data request from another access point also apart from a mobile host in its cell site. In this case, the access point uses Access point Algorithm-2. According to this algorithm, hot data of the access point (AP_{hd}) is first searched for the requested data item. If data is found in hot data, the data item is returned to the requesting

Access point Algorithm-1
Input: *Query for data item d (Q_d) from Mobile host MH_Q*
Output: *Returns requested data item d to MH_Q*

1. *If ($d \in AP_{hd}$)*
2. *Return d from AP_{hd}*
3. *Else if ($M_d \in MD0$)&& ($U_c = 'Y'$)*
4. *Return $ID_{MH} \mid D_{MH} \ni d$*
5. *Else if ($M_d \in MD0$)&& ($U_c = 'N'$)*
6. *Download d from corresponding $VI_{MH} \mid D_{MH} \ni d$*
7. *Return d to MH_Q*
8. *Else if ($M_d \in MD1_{CA_y}$)*
9. *Forward Q_d to CA_x and retrieve data d*
10. *Return d to MH_Q*
11. *Else forward Q_d to VI_Q in cloud*

Figure 6: Data access algorithm at access point

access point (AP_Q). As explained in the data pre-distribution section, the access points always maintain the metadata (i.e. MD0) of all the content present with mobile hosts in that Coverage Area. If the requested data is not present in the hot data, metadata repository MD0 is searched for a possible match. If there is a match, it implies that one or more of the mobile hosts in that Coverage Area contain the requested data file. Access point downloads the actual data file from the corresponding virtual image (VI_R) in cloud and returns it to the querying access point (AP_Q). Figure 7 provides the formal algorithm for data access at access point when the request is received from another access point belonging to neighboring CA.

Access point algorithm-2:
Input: *Query for data item d(Q_d) from AP in neighboring CA*
Output: *Returns requested data item d to requesting AP*

1. *If ($d \in AP_{hd}$)*
2. *Return d from AP_{hd}*
3. *Else if, ($M_d \in MD0$)*
4. *Download d from corresponding (VI_R)*
5. *Return d to AP_Q*

Figure 7: Data access algorithm at access point when request is received from neighboring CA

5.1 Data access algorithm at VI in cloud

Access point forwards the data request to respective virtual image in the cloud (VI_Q) when it does not have any entry corresponding to the requested data. As mentioned in the architecture section, each location has a dedicated Master Virtual Image (MVI) in the cloud. Along with the actual data files belonging to that location, each MVI maintains following two types of metadata:

—Local Metadata(MD_L)= Metadata of all the data items belonging to that location (hereby termed as local data)
—Transit Metadata(MD_{CS})= Metadata of contents present with VIs which are currently under its jurisdiction.

When the virtual image (VI_Q) corresponding to the querying mobile host (MH_Q) receives the query, it follows Virtual Image data access Algorithm mentioned below. According to this

algorithm, if the requested data d is found among the data contained with the virtual image (D_{VI_Q}) as a result of prior pre-distribution, it is returned to the querying mobile host (MH_Q) via the access point (AP_Q). Else, it forwards the query to its master virtual image (MVI_Q). Master virtual image checks its metadata repository of local data (MD_L) for a match with the metadata (M_d) of queried data item. If a match is found, it implies that requested data item belongs to that location (local data). The requested data is retrieved from MVI's local data set (D_{MVI}) and returned to the querying virtual image VI_Q, which in turn returns to mobile hostvia access point. Else, if there is no match, then the MVI checks for match in MD_{CS}. If there is a match, then it implies that data is not local data but it is present with one of the virtual images, which joined the group as its mobile host moved to this Coverage Area. The data item d is retrieved from the virtual image containing the data (VI_R) and returned to querying virtual image. Else if there is no match, then based on information from location tracker and grouping system, master virtual image (MVI_Q) gets the identity (ID_{MVI_R}) of master virtual image corresponding to the location to which data item belongs to (MVI_R) and retrieves data item from it. It then returns data item to the querying mobile host via its corresponding VI and the access point. If the data item is not found with the other master virtual image also, then a "Data not found" message is returned to the user. Figure 8 provides the formal algorithm for data access in the cloud when VI receives the request from the access point.

Virtual Image data access Algorithm
Input: *Query for data item d (Q_d) from Access point*
Output: *Returns requested data item d or "Data not found" message*

1. *If ($d \in D_{VI_Q}$)*
2. *Return d to MH_Q via AP_Q*
3. *Else VI_Q forwards Q_d to MVI_Q*
4. *If ($M_d \in MD_L$)*
5. *Retrieve d from D_{MVI}*
6. *Return d toVI_Q. Return d toMH_Qvia AP_Q*
7. *Else if ($M_d \in MD_{CS}$)*
8. *Retrieve d fromVI_R.*
9. *Return d to VI_Q and to MH_Q viaAP_Q.*
10. *Else retrieve ID_{MVI_R} from Grouping system*
11. *Retrieve d from MVI_R*
12. *Return d to VI_Q. Return d to MH_Q via AP_Q*
13. *If ($NOT(d \in D_{MVI_R})$)*
14. *Return "Data not found" message*

Figure 8: Data access algorithm for VI in the cloud

5.2 Message complexity

As per our proposed scheme, the best possible case in data access is when the data requested by user of a mobile host is present in hot data of its access point. In such case, it takes a total of only two messages to get the requested data i.e. one request message and one response message. So, message complexity would be O(1).

The worst-case scenario would be when the requested data does not belong to current location and it does not have any entry at access point, VI and also current MVI. In such case, as explained in the algorithm of Figure8, the current MVI checks with grouping system for the identity of owner MVI based on the location information present in the request. Then, it sends the request to that owner MVI and retrieves the requested data and sends it to requesting mobile

host through the VI and access point. If the term 'nodes' represent routing points through which the request is served, then according to the above mentioned mechanism, for 'N' nodes it takes 'N' messages for sending the request and N-2 messages to receive the requested data. Therefore, the total number of messages in worst case scenario is 2(N-1). Therefore, the message complexity in worst case scenario would be O(N).

6. UPDATE PROPAGATION

Image and video files will not have any updates other than may be metadata updates such as change in resolution, etc. Actual data file does not change i.e. actual picture or video does not change. But the other types of data we are considering here are the text files, which can have updates. Hence, update propagation is a very important aspect to mention. One important assumption is that only the owner can update a file. Since caching of data is done both at access point and also at VIs in the cloud, update propagation also needs to happen at both the places.

6.1 Update propagation among access points

The process of update propagation among access points is similar to the process when a new data file is created. When a text file is updated, it is immediately uploaded to its VI in the cloud. At the same time, its metadata is updated at the access point. The access point checks if the updated file is a hot data, if so the actual data file is updated in the hot data list. As mentioned earlier, all the access points in a Coverage Area need to have the same entries. So, the access points transmit the updated metadata or the updated hot data to all other access points in its Coverage Area and waits for acknowledgements similar to the process when a new data file is created. The peripheral access points update the same with the access points in the neighboring Coverage Areas.

6.2 Update propagation among VIs

When the VI receives an updated file, we have two cases based on the current location of the corresponding physical device. In other words, based on the group to which the VI belongs.

6.2.1 *Case 1: The corresponding MH is in same location as origin location of the file..* VI updates the MVI (in this case, it becomes the owner MVI) with updated file. MVI broadcasts the updated file to all MVIs in the system. MVIs which have an entry for the metadata of that file, pick up the updated file and forward it to the VI holding the replica of that file. VI after receiving the updated file from MVI sends a notification to the MH about this update.

6.2.2 *Case 2: The corresponding MH is not in same location as origin location of the file:.* VI updates the local MVI with metadata of the updated data file. Since all the files are location tagged, VI can find the owner MVI of the file from the grouping system. VI then sends the updated file along with metadata to MVI corresponding to origin location of the file via its current MVI. Owner MVI broadcasts the updated file to all MVIs in the system. MVIs which have an entry for the metadata of that file picks up the updated file and sends to VI holding the replica of that file. VI after receiving the updated file from MVI sends a notification to the mobile host about this update.

6.3 CACHE REPLACEMENT

Till now, we have discussed about the pre-distribution and caching scheme to reduce the latency in data access in MCC. According to our proposed scheme, data/metadata is being cached at three places; Access points(APs), Master Virtual Images (MVIs), and Virtual Images of mobile hosts (VIs). The amount of storage at each of these places is limited. Though, the MVIs and VIs reside in the cloud, we have to consider the amount of storage available to each of the MVIs and VIs as limited since pricing in cloud services is based on storage space used. In such a scenario, the limited storage available should be used effectively by caching the correct data files. Thus, a proper cache replacement scheme plays an important role in the overall efficiency of the caching

scheme. Important factors that need to be considered in this cache replacement policy would be (a) Storage limitation. (b) Frequency of access (No. of hits) for a particular record (c) Amount of time in seconds left before the file becomes invalid (Validity of data) and (d) Size of the record. Here, we define a new parameter called Cache Retention Factor (CRF). It is calculated as

$$CRF = (\beta + No.ofhits) \times Validityofdata \tag{7}$$

Where β is the numeric constant determined by the application. In the following subsections we discuss the cache replacement mechanism at each of the three places MVIs, VIs and access points where the data is stored/cached.

6.4 Cache replacement at MVIs

As per our proposed scheme, each Coverage Area has a dedicated master virtual image in the cloud. Every MVI stores three kinds of information

—*Local data*: The actual data files belonging to that location i.e. the data files created in that location. Data is deleted from this list when the file becomes invalid. Other than that, when the memory allocated to this listis full, the cache replacement follows the following replacement order:
(1) Entry with least CRF is replaced first.
(2) If two files have the same least CRF then file with less hits is replaced first.
(3) If the number of hits is also same, then the file with bigger size is replaced first.
(4) If sizes are also equal, then least recently used data is replaced first.

—Local metadata (MD_L): This is simply the metadata of content belonging to that location i.e. local data. So, there is no explicit replacement policy for this list. When a local data file is removed from cache, its metadata is also removed from this list.

—Transit metadata (MD_{CS}): Metadata of content belonging to some other location but present with the VI under its jurisdiction. An entry is deleted from the cache when the validity of file expires. Apart from that, cache eviction takes place in two other cases.

—Case 1: When mobile host moves out of the Coverage Area (CA), the corresponding VI in the cloud also groups with new MVI. Then the old MVI immediately deletes the metadata corresponding to that VI.
—Case 2: If the cache allocated to this list is full, then the record with lowest CRF value is removed. If more than one metadata entries have same least CRF value,then the record with least no. of hits is replaced first. If the no. of hits is also same for the records then the record which is least recently accessed is replaced first.

6.5 Cache replacement at individual VI

According to our proposed scheme, individual virtual image of each mobile host caches three kinds of data.

—*Saved data*: This includes the data created by its own mobile host and also data fetched/accessed from other VIs as per users request. If the memory is full, the user has to manually delete the unwanted files to accommodate the new file.

—*Pre-distributed unsaved data*: According to our scheme, when someone creates a new data file in the current location of the user, it is immediately cached to the user's VI in the cloud. If the user accesses that data it is moved to the saved data cache. If not, they will remain in this temporary cache. This data is deleted from the cache when the validity of the file expires. Apart from that, cache replacement takes place in two cases
—Case 1: When the mobile host moves out of the current location (CA), then the entry is deleted.

—Case 2: If the cache allocated to this list is full, with mobile host still in the location, the replacement is done based on FIFO. A simple FIFO replacement would suffice for this list because if the user was interested in a particular data in this list, then this would have moved to the Saved data as soon as it is accessed by the user. Oldest entry in this list implies that it is not of interest. Hence, oldest entry is replaced first.

—*Pre-distributed metadata*: According to our proposed pre-distribution scheme, if a particular user is considered to be a potential user for a data created in another location, then its metadata only is pre-distributed to it. Records in this cache are replaced based on FIFO or when the file becomes invalid whichever is earlier. A simple FIFO replacement would suffice for this list because if the user were interested in a particular data associated with a metadata in this list, then this would have moved to the saved data as soon as it is accessed by the user. Oldest entry in this list implies that it is not of interest to the user and hence, the oldest entry is replaced first.

6.6 Cache replacement policy at access points

The cache memories at the access points store the following three sets of data:

—Hot data: According to our proposed scheme each access point stores the high frequency access data items for that Coverage Area called hot data.

—MD0: Metadata of the all the content present with the mobile hosts present in that Coverage Area.

—MD1: Metadata of content present in neighboring Coverage Areas.

According to our pre-distribution scheme, hot data and MD1 at each access point are refreshed after every time period $'T_r'$. During the same process records with whose validity is over are discarded. An explicit cache replacement policy is not required for these two lists. However, MD0 at each access point includes metadata of hot data and metadata of all the content present with the mobile hosts in that Coverage Area. Hence, there is possibility that latter part of MD0 can increase and fill the cache before next snapshot after $'T_r'$ timeperiod. In such case, the cache replacement policy checks for cache retention factor (CRF) of the metadata records. Record with lowest CRF value is replaced first. If more than one record has same least CRF value then the record with least no. of hits is replaced first. If the number of hits is also same for the records then the record, which is least recently used is replaced first.

7. EXPERIMENTS AND RESULTS

The basic objective of the proposed scheme is to reduce the latency in accessing information data when communication between mobile devices is offloaded to the cloud. So, the experimental setup for this scheme is similar to the C2C platform mentioned in [13] with appropriate customizations that are required for our scheme. Pictorial representation of architecture is shown below. In real environment, mobile devices access the Internet through access points or cell towers, which are connected by fixed network. In the proposed scheme, the access points are assumed to have limited storage and processing capabilities. In order to simulate this scenario, we use a desktop systems connected to the Internet via Ethernet and we enable Internet Connection Sharing (ICS) on these machines in Wi-Fi mode. Mobile devices connect to these machines in wireless mode to access the Internet. In other words, the desktop machines act as access points with limited storage and processing capabilities. When the user creates a new file, it is transmitted to the desktop system acting as access point with limited capabilities using the Wi-Fi connection and from there it is transmitted to its corresponding virtual image in the cloud via the Ethernet connected to Internet.

7.1 SETUP

Our test bed consists of 9 Android mobile devices to be used as physical mobile devices described in the design. The configuration of these Android mobile devices ranges from Samsung S III

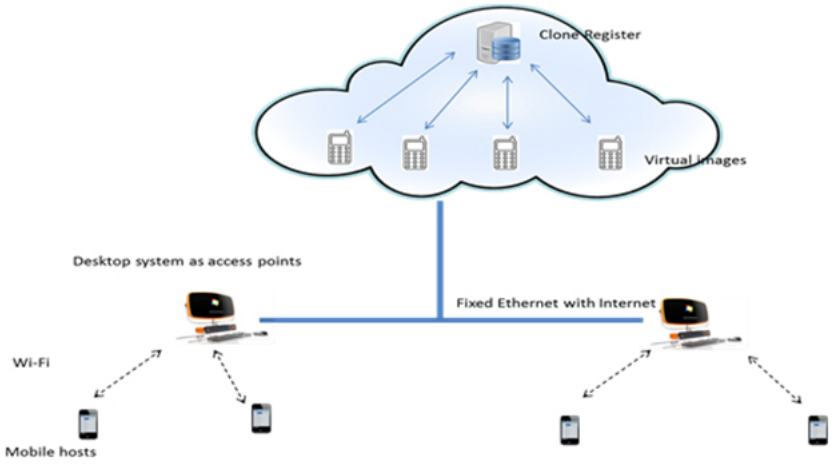

Figure 9: Simulation architecture

with OS Android 4.0.4 to Google Nexus 7 with OS Android Kit Kat (4.4.2) operating system. We choose Android devices mainly for two reasons. First reason, it is an open source mobile operating system and second reason is the compatibility with the software clone in the cloud. The software clone, which we used as virtual image of the smart phones in the cloud is based on Android operating system. Details about the software clone are explained in the following paragraphs. Since, the propose scheme is not specifically dependent on any mobile OS, the results should be applicable to mobile devices with other operating systems as well. The simulated access point is Dell Optiplex 780 desktop system running Windows 7 operating system with Intel Core 2 quad core processor, and 8 GB RAM. To enable mobile devices to communicate with this simulated access point, we use Edimax EW-7811Un Wifi USB micro adapter. This system is in turn connected to the Internet through Ethernet.

On the cloud end of the architecture, we have selected Amazon EC2 as the cloud service provider. Each virtual phone in the cloud will be a micro instance with upto 2 ECUs (EC2 computing units), 1 virtual CPUs and 0.613 GB Memory. From a software engineering perspective it would be appropriate to use Android operating systems only for virtual images as well. However, Android OS is originally designed for ARM architecture in mobile devices. As Amazon EC2 is not ARM architecture, we use Android-x86 operating system for the clones in the cloud server. It is a modified Android OS built to work on the system with x86 architecture. But virtualization environments such as VirtualBox are not compatible with Amazon EC2 except for QEMU emulator but it requires a Linux instance to be launched on Amazon EC2. Thus, it creates three layers of hypervisors, which is far from being efficient. Amazon EC2 uses Xen virtualization environment on top of which Amazon Machine Images (AMIs) are run. AMIs are pre-configured bundles with operating systems and required application software. There is no ready to use Android-x86 AMI for Amazon EC2 and hence was required to be created with customization according to the requirements of Amazon EC2. We used such an Android AMI provided by the authors of the paper [13]. For more details about the customizations required for Android-x86 and process to create Amazon AMI, refer to the paper[13]. The clone register is a Linux instance with a relational database to store the details of clones residing in the cloud.

7.2 PERFORMANCE

In order to prove that the proposed architecture and scheme reduces the latency data in Mobile cloud computing, we need to compare the response time taken to get the requested data in under three cases; when it is fetched from hot data, when it is fetched from adjacent access point and finally when it is fetched from cloud. To evaluate these cases, we try to fetch a 4MB file, 10MB

file and 25 MB file from them.

We compare the time taken to fetch the 4MB file in each case. We observe that the time taken to fetch the file from hot data access point is the lowest as expected. Next best is when the requested data is fetched from the access point in the neighboring coverage area. This time is 12.3 % more than when it is fetched from hot data collection in the access point. The time taken when the data has to be fetched from cloud is 656 % more than when the data is fetched from hot data in the access point and it is 573 % more than when the data is fetched from adjacent access point.

The time taken when the file has to be fetched from cloud after broadcasting it to other VI's (5 instances) is almost 20 times the time taken to fetch the file from the hot data access point. The time taken to fetch the file after searching for the file in 15 instances is 32% more than time taken to fetch the file after searching for it in 5 instances and 20% more than time taken to fetch the file after searching for it in 10 instances. Below figures indicate these results for 4 MB file. Note that the time taken to fetch an image from the cloud is done off-line based on pre-fetching based on the application demand. Thus, the time to get a file from the cloud to the device is not impacted in most of the cases.

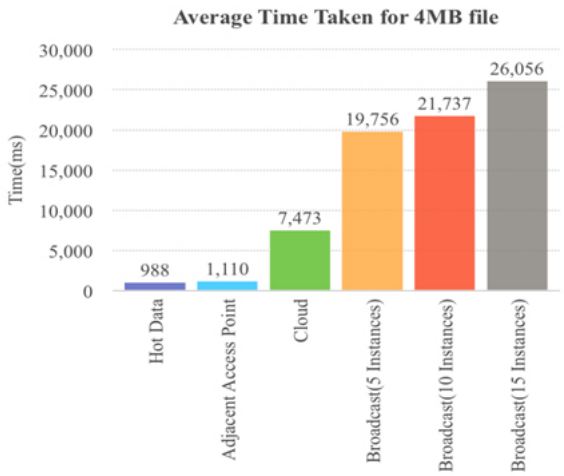

Figure 10: Average time taken to fetch 4 MB file in each case

Similarly, for 10MB file the time taken to fetch the file from adjacent access point is 13% more when compared to time taken to fetch from hot data and time taken to fetch from cloud is 654% more than when it is fetch from hot data and 566% more than when it is fetched from adjacent access point.

The time taken when the file has to be fetched from cloud after broadcasting it to other VI's (5 instances) is almost 18 times the time taken to fetch the file from the hot data access point. The time taken to fetch the file after searching for the file in 15 instances is 26.7% more than time taken to fetch the file after searching for it in 5 instances and 16.7% more than time taken to fetch the file after searching for it in 10 instances. Below figures indicate these results for 10 MB file. For 25 MB file also, the overall results observed are to be same as earlier. The time taken to fetch the file from hot data is the lowest and time taken to fetch the file from virtual image in the cloud is the highest. For this file, the average time taken to fetch it from adjacent access point is 9% more than when it is fetched from hot data. Similarly, the average time taken for file from virtual image in the cloud is 512% more than when it is fetched from hot data in the access points.

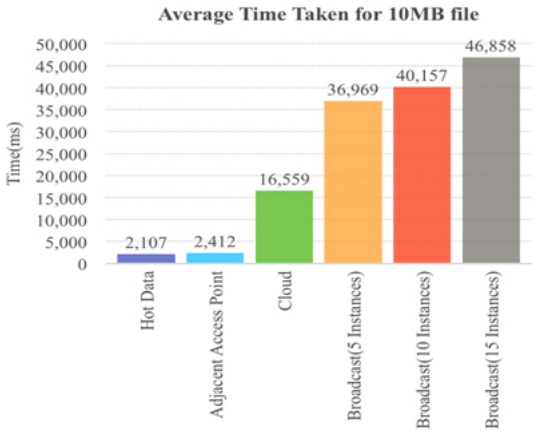

Figure 11: Average time taken to fetch 10MB file in each case

The time taken when the file has to be fetched from cloud after broadcasting it to other VI's (5 instances) is almost 22 times the time taken to fetch the file from the hot data access point. The time taken to fetch the file after searching for the file in 15 instances is 5.8% more than time taken to fetch the file after searching for it in 5 instances and 3.3% more than time taken to fetch the file after searching for it in 10 instances. Below figure illustrate these results.

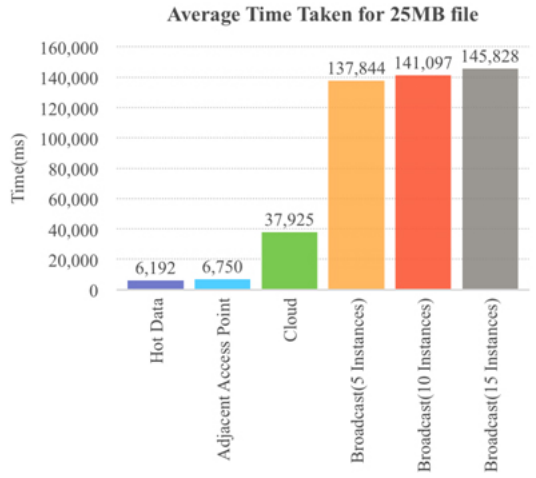

Figure 12: Average time taken to fetch 25MB file in each case

Thus, with respect to latency in accessing data in Mobile cloud computing, the proposed three-layered architecture and data management schemes above are justified.

8. CONCLUSIONS

In this paper, we have proposed a three-layered design and development of an efficient data transfer mechanism among mobile devices using Mobile cloud computing paradigm. In Mobile cloud computing, users connect to cloud service providers over the Internet and leverage the cloud resources to perform their processing, storage and communication tasks. We also proposed a pre-distribution scheme based on this architecture for efficient data sharing among potential users with supporting data access mechanism, update propagation mechanism and cache replacement mechanisms. We also performed complexity analysis for data access using the proposed

architecture and scheme. Finally, we implemented the proposed architecture and scheme with actual devices and verified the efficiency of the schemes using EC2 cloud services.

REFERENCES

ABI. 2009. Mobile cloud computing subscribers to total nearly one billion by 2014. Tech. rep., ABI, http://www.abiresearch.com/press/1484/.

Barbera, M. V., Kosta, S., Mei, A., Perta, V. C., and Stefa, J. 2013. Cdroid: Towards a cloud-integrated mobile operating system. In *IEEE INFOCOM 2013*. Turin, Italy.

Barbera, M. V., Kosta, S., Mei, A., and Stefa, J. 2013. To offload or not to offload? the bandwidth and energy costs of mobile cloud computing. In *IEEE INFOCOM 2013*. IEEE, Turin, Italy.

Bisdikian, C., Mitschang, B., Pedreschi, D., Tseng, V. S., and Bettini, C. 2011. Challenges for mobile data management in the era of cloud and social computing. In *12th IEEE Intl. Conf. on Mobile Data Management,Sweden*. IEEE.

Chen, E. Y. and Itoh, M. 2010. Virtual smartphone over ip. In *IEEE International Symposium on a World of Wireless Mobile and Multimedia Networks (WoWMoM)*. IEEE, pp.1–6.

Feldman, A. J., Zeller, W. P., Freedman, M. J., and Felten, E. W. 2010. Sporc:group collaboration using untrusted cloud resources. In *OSDI'10*.

Han, Q. and Gani, A. 2012. Research on mobile cloud computing: Review, trend and perspectives. In *Second International Conference on Digital Information and Communication Technology and it's Applications (DICTAP)*. pp.195–202.

Hoang, T. D., Chonho, L., and DusitNiyato, P. W. 2011. A survey of mobile cloud computing: architecture, applications, and approaches. Wireless Communications and Mobile Computing.

Juniper. 2010. Mobile cloud computing: $9.5 billion by 2014. Tech. rep., http://www.readwriteweb.com/archives.

Kosta, S., Perta, V. C., Stefa, J., Hui, P., and Mei, A. 2013. Clone2clone (c2c):peer-to-peer networking of smartphones on the cloud. In *USENIX HotCloud 2013*. San Jose, CA, USA.

Samimi, F. A., McKinley, P. K., and Sadjadi, S. M. 2006. Mobile service clouds: A self-managing infrastructure for autonomic mobile computing services. In *Self-Managed Networks, Systems, and Services*. Springer Berlin Heidelberg, pp.130–141.

Satyanarayanan., M. 2010. Mobile computing: the next decade. In *1st ACM Workshop on Mobile Cloud Computing & Services: Social Networks and Beyond, MCS'10*. ACM, New York, NY, USA, pp.1–5.

Stuedi, P., Mohomed, I., and Terry, D. 2010. Wherestore: Location-based data storage for mobile devices interacting with the cloud. In *MCS*.

wikipedia. `http://en.wikipedia.org/wiki/Metadata#Photographs`.

Siva Naga Venkata Chaitanya Vemulapalli earned his Bachelor's degree in Electronics and Communication Engineering from Jawaharlal Nehru Technological University, India and Master in Computer Science from the Missouri University of Science and Technology, Rolla, USA in 2014. His research interests are in cloud computing, and mobile computing.

Sanjay Kumar Madria is a full professor in the Department of Computer Science at the Missouri University of Science and Technology (formerly, University of Missouri-Rolla, USA) and site director, NSF I/UCRC center on Net-Centric Software Systems. He received his PhD from IIT Delhi, India in 1995 in the area of Nested Transaction Management in Database. He has published over 200 Journal and conference papers in the areas of mobile data management, Sensor computing, and cyber security and trust management. He won three best papers awards including IEEE MDM 2011 and IEEE MDM 2012. He is the co-author of a book published by Springer in Nov 2003. His research is supported by several grants from federal sources such as NSF, DOE, AFRL, ARL, ARO, NIST and industries like Boeing, Unique*Soft, etc. He has also been awarded JSPS (Japanese Society for Promotion of Science) visiting scientist fellowship in 2006 and ASEE (American Society of Engineering Education) fellowship at AFRL from 2008 to 2012. In 2012-13, he was awarded NRC Fellowship by National Academies. He has received faculty excellence research awards in 2007, 2009, 2011 and 2013 from his university for excellence in research. He served as an IEEE Distinguished Speaker, and currently, he is an ACM Distinguished Speaker, and IEEE Senior Member and Golden Core awardee.

Mark Linderman is the Technical Advisor at the Air Force Research Laboratory Information Directorate in Rome, New York where he earlier led the information management technology area for over 15 years. He has worked on reliable distributed repositories, peer-to-peer systems and data streaming research in recent years. He is also working in context/situation aware systems. He received his Ph.D. from Cornell University in 1995 and a BS in EE from the University of Delaware in 1986. He is a Senior Member of the IEEE.