SASCloud: Ad hoc Cloud as Secure Storage

Shahid Al Noor, Md. Mahmud Hossain and Ragib Hasan
{shaahid, mahmud, ragib}@cis.uab.edu
Department of Computer and Information Sciences, UAB, AL 35294-1170

***Abstract*—With the emergence of high-speed 4G networks along with reachable Wi-fi systems, cloud computing frameworks can greatly leverage mobile domains. However, obtaining a temporary storage service in a communication-challenged area is difficult due to the unavailability of any secure third-party cloud systems. Although the existing ad hoc cloud architectures facilitate distributed computation and sensing operations, such systems fail to deliver secure ad hoc storage as a service when a client requests for secure storage. The absence of a proper centralized monitoring system in the existing ad hoc clouds is a major obstacle for convincing a client to trust the neighboring mobile nodes for content offloading. In case a client and an outsourced node get disconnected, retrieving the offloaded contents along with ensuring their confidentiality and integrity becomes non-trivial. Additionally, providing a feasible and justified monetary incentive is a complex process for such ad hoc mobile frameworks. In this paper, we propose SASCloud, a centrally controlled ad hoc cloud system that provides a secure and reliable storage service for mobile clients. Our proposed system uses the contextual information of mobile users along with partial environmental knowledge and forms a temporal cloud using the resources of neighboring mobile devices. Along with the detailed reasoning of possible threats in our model, we provide a secure framework for content distribution and retrieval. We provide extensive analysis of our model using simulated experimental modules and demonstrate the feasibility of providing a secure storage service in a network-disconnected area using SASCloud.**

***Keywords*-DTN, Mobile Cloud, Ad hoc Cloud, Distributed Storage**

I. Introduction

Mobile cloud computing enhances the storage and computing power of mobile devices by leveraging third-party cloud systems for data and computation offloading [1]. The introduction of low latency and location-aware fog systems reduces the time and cost associated with incorporating such mobile cloud systems [2]. Most of those mobile cloud systems fail to provide cloud services in a network-disconnected area, such as an area affected by a natural disaster. Although several ad hoc cloud systems have been proposed considering the neighboring mobile devices' computing power, such systems are unable to serve when the client requests storage as a service. On the other hand, the existing delay-tolerant networking (DTN) approaches offer temporary storage service but due to the lack of centralized control, cannot pledge the retrieval of client's contents in time. Therefore, we need an efficient and secure storage-as-a-service for such network disrupted environments.

An ad hoc cloud is composed of mobile nodes with no-exclusive and intermittent resources, where nodes do not have any pre-commitment to each other and can accept multi- faceted tasks [3]. Some of the volunteer mobile computing platforms, such as Hadoop Apache [4], BOINC [5], CloneCloud [6], CellCloud [7], and GEMCloud [8], considered energy efficient mobile devices to provide a feasible and energy efficient mobile cloud. Although they provide secure crowd-source based platforms, such systems require continuous Internet connection between the mobile nodes and the central management system to distribute the tasks among nodes and receive results from them. Therefore, it is not possible for the devices in a rural setting to participate in such systems. Moreover, even in urban areas, we cannot expect the continuous presence of a wireless access point. On the other hand, researchers have proposed the formation of opportunistic cloud systems by considering the neighboring mobile devices and divide and distribute the resource-intensive tasks, such as data collection, image processing, etc., among those devices [9–13]. Although those architectures do not require any Internet connection, there exists a high risk of security as compared to hiring a trusted cloud system. Without the continuous presence of any centralized authority, clients often lose control over the outsourced data, fail to detect dishonest mobile devices, and protect the sensitive information from the attackers.

In this paper, we propose SASCloud, an **A**dhoc cloud system that provides **S**ecure **S**torage service in a network disconnected area. Initially, a mobile user registers his mobile device with the SASCloud central authority (CCA) and becomes a Registered Mobile Node (RM-Node). Each RM-Node also receives a Performance Point (PP) that specifies its trustworthiness within the framework. Also, SASCloud has some registered cloudlets that are used for content collection and integration. When any client requires the storage service, it searches for some interested neighboring RM-Nodes i.e., the interested RM-Nodes within its Bluetooth or wifi range and forms an ad hoc cloud by selecting some of those RM-Nodes. It divides and distributes the content among the RM-Nodes in that ad hoc cloud. When any of the RM-Nodes in that ad hoc cloud meets any cloudlet while moving to a different physical location, it delivers its content to that cloudlet. Each cloudlet integrates the received contents and provides the integrated content to the CCA. The CCA combines all the contents from various cloudlets and delivers the resultant content to the client.

However, the primary challenge of building the SASCloud is ensuring the integrity and confidentiality of clients' outsourced contents. Second, motivating the user for participation, and dividing and distributing the contents among them are complex tasks. Third, ensuring the authenticity of client and RM-Nodes is very challenging. Finally, we require a proper verification mechanism when either a client or RM-Node denies sending or receiving of any offloaded content.

Contributions: The contributions of the paper are as follows.

1) To the best of our knowledge, SASCloud is the first attempt that forms an ad hoc cloud for providing storage service with a centralized control for secure content distribution and retrieval.
2) We propose a black box map-based content distribution

algorithm for efficiently distributing clients' contents without revealing RM-Nodes' future location information to the clients.

3) We demonstrate the feasibility and performance of our proposed SASCloud model via extensive simulation and analysis of experimental results.

The rest of the paper is organized as follows. In Section II, we illustrate the motivation of our research work followed by some of the related research work in Section III. We describe the underlying SASCloud architecture, and the detailed operational and messaging model in Sections IV, V, and VI respectively. We discussed the possible threats in our proposed model along with the countermeasures in Section VII followed by the experimental result in Section VIII. Finally, we discuss and conclude our work in Section IX.

II. Motivation

The existing mobile cloud systems primarily focus on distributing task computation among the high-performance mobile devices. In addition to outsourcing computation, users also require content outsourcing when their devices do not have enough space to accommodate new contents. Although several cloud-based storage/backup services, such as iCloud, Time Machine, and Dropbox facilitate content offloading, those services require continuous Internet connectivity to upload contents to the cloud. However, a vast number of mobile users do not use any data plan. Furthermore, we cannot ensure network connectivity in all places especially in rural areas or regions affected by natural disaster. Therefore, users search for a nearby storage backup system.

Consider that John is visiting a historic place and wants to take several photos of the place using his smartphone. He finds that the smartphone does not have enough space to accommodate new pictures. Unfortunately, John did not notice earlier that his data plan was expired. As a result, he cannot take the advantages of the cloud-based storage services for temporary storage backup. However, John can use the SASCloud storage backup service to backup his contents temporarily. The SASCloud uses the neighboring mobile devices as a form of storage backup.

Recent incidents, such as mass email deletion by Gmail [14], Apple's MobileMe post-launch downtime [15], and T-Mobile Sidekick personal data loss [16] indicate that ensuring the integrity of data even for a traditional cloud is very challenging. In the case of an ad hoc cloud, the problem is even more severe, since a client does not have any active control of their data as soon as they transfer those to the crowd-sourced devices. For example, the contacted mobile devices that are used as a temporary data storage might just delete the data or return fake data to the client. Hence, protecting the outsourced data from loss and ensuring the integrity of them is indispensable.

III. Related Work

Researchers proposed several approaches for distributing tasks among the neighboring mobile devices. Vehicular ad hoc network (VANET) is one such approach where a wireless onboard unit (OBU), a roadside unit (RSU), and an authentication server (AS) are installed in each vehicle participated on computation [17]. However, installing those components in each mobile device is very expensive. One major flaw in their proposed VANET system is, the vehicles are considered trustful based on the recommendation of a law executor or other secure vehicles. However, finding a law agent or trusted vehicle for participation, or verifying the signature of a law trustee from a large number of law executors is tough. Mobile cloud is widely used for running several complex applications in a resource-constrained mobile device by hiring some other mobile devices for task offloading [18]. Existing mobile cloud systems can be classified into two categories; i) partially controlled, ii) fully controlled. In the partially controlled system, mobile devices use third party networking infrastructure for task/result transfer whereas, work distribution and monitoring, and negotiation during the hiring of other mobile devices are done by the client itself [19–22]. However, motivating the mobile users for participation, building a trust relationship with them, and monitoring the task progression in such mobile cloud system is tough. Crowdsourcing is one popular approach for motivating participants where some portion of a task is outsourced [23]. It is used in several areas, such as AI [24], business [25], construction [26], etc. In a fully controlled mobile cloud system, a central system is used for task distribution, monitoring, and organization, and verification of security related issues [6–8, 27]. The problem with this type of system is that it requires a continuous connectivity with a central system. Therefore, this approach does not work if a client seeks for a prompt cloud service in a network-disconnected area. To perform complicated operations in a network uncovered area, some researchers proposed an ad hoc mobile cloud system where the client can form a cloud on demand after communicating with its neighboring devices and hire their resources for task processing [28, 29]. Although such ad hoc cloud system is easy to form and requires less time for task processing, creating the trust relationship with the neighboring devices is very challenging. Noor et al. proposed a context-based delay tolerant cloud where the context of the nearby mobile devices is used for efficient content delivery [30]. Additionally, a central control point is used for verifying the authenticity of the participated nodes and their generated result. However, they just considered the computation offloading and did not address anything about the integrity of the outsourced data in case client would like to take storage service from the cloud.

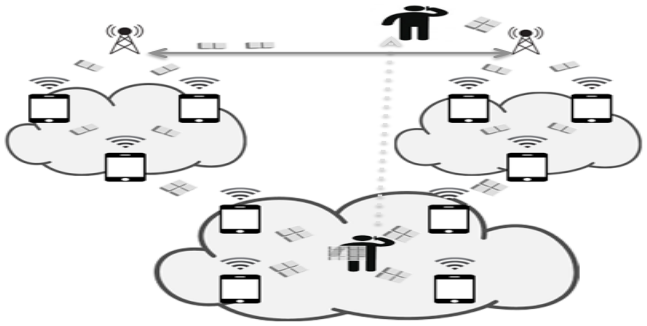


Fig. 1: SASCloud Architecture

IV. Architecture

The conceptual architecture of SASCloud is shown in figure 1. We use the architecture proposed in [30] as the foundation of our proposed model. The users need to register their mobile devices with Cloud Central Authority (CCA) to participate

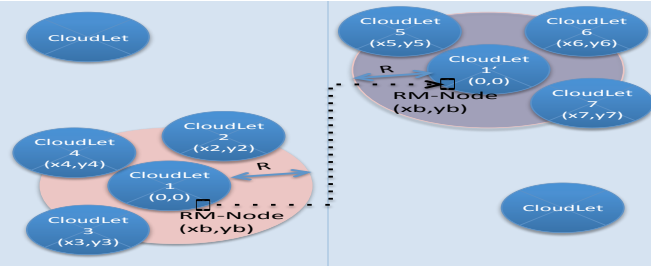


Fig. 2: Process of Location Collection

on the cloud formation. The registered bidder receives a performance point (PP) that specifies how reliable that user is for a task. The PP is increased or decreased depending upon their performance in the previously assigned job. A client in a critical area initiates bidding by sending a query message and the interested RM-Nodes reply to the client. The client verifies all the RM-Nodes' information, selects some RM-Nodes, and forms an ad hoc cloud. The client orders the contents based on their priority and divides and distributes them among the RM-Nodes based on the total interested RM-Nodes, the context of those RM-Nodes, and the available storage in each RM-Node. When an RM-Node inside the ad hoc moves to a different place can further form another level of ad hoc cloud considering the neighboring interested RM-Nodes in that region, and divide and distribute the received contents further among the RM-Nodes in the second level of that ad hoc cloud. When any of the RM-Node in the ad hoc cloud leaves the disconnected area and meets a cloudlet, it delivers its received contents to that cloudlet.

A. RM-Node

When a user registers its device and becomes an RM-Node, it receives an id and an initial PP. Besides, it also receives a temporary certificate from SASCloud through its assigned cloudlets. Once the certificate expires, the RM-Node has to request for certificate renewal to any designated cloudlet of the SASCloud. The performance of the RM-Node is evaluated, and the PP is updated after the evaluation. The corresponding cloudlet can issue a new certificate during that assessment time. Furthermore, the last cloudlet any RM-Node meets before it reaches to a network disconnected region is considered the representative cloudlet (RC) for that RM-Node. However, if the bidder moves to the coverage area of another cloudlet, that cloudlet becomes its new RC and the information attached by the previous RC is invalidated. As shown in figure 2, the bidder's first RC was cloudlet1 when it was in region1, and as soon as it moves to region2, cloudlet1$^{/}$ becomes its new RC.

B. Cloudlet

Just like a cellular base station, the cloudlet in SASCloud can communicate with the RM-Nodes using any mobile infrastructure. Besides, it has sufficient memory for storing the RM-Nodes' delivered contents and a decent amount of computational power for performing any resource-intensive operations. When an RM-Node is inside the coverage area of a cloudlet, the cloudlet attaches the location coordinate information of itself and n of its randomly selected neighbors. We refer this location information as Representative Cloudlet Context (RCC).

If we observe figure 2, we see that the RC of the RM-Node is cloudlet1. Cloudlet1 finds five more cloudlets within its circular coverage area of radius R. The RC sends the coordinate information of itself and all the other five cloudlets to the RM-Node. The RM-Node can easily interpolate those coordinates information from its current location. Since an RM-Node is continuously moving, the interpolated values will be different for different location. Updating the information of all the other cloudlets with continuous changes of location is expensive. Hence, the RM-Node only updates the coordinate information of the RC with its new location while the other cloudlet location is updated reactively when any other RM-Node requests information during the ad hoc cloud formation.

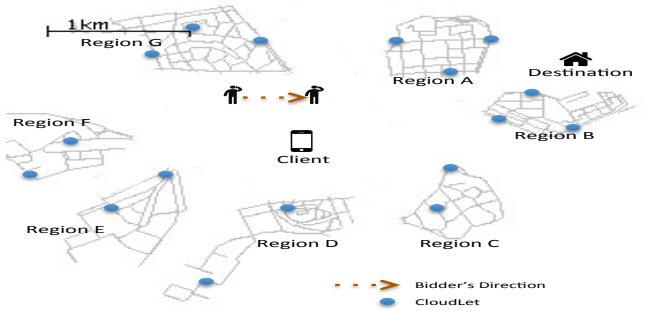


Fig. 3: Process of finding the nearest cloudlet

V. Operational Model

A. Predicting the time of meeting the nearest cloudlet

Each RM-Node can carry a portion of world map just like a mobile GPS. We propose two approaches for finding the nearest cloudlet and determining the time to meet that cloudlet.
Node based approach: In this method, an RM-Node receiving any content estimates the time to meet the nearest cloudlet on its way. The RM-Node first sends its RCC information and requests client to send any other RCC information that it is currently carrying. However, instead of sending all the collected RCCs, the client can send only some them based on the RM-Node's direction of movement. As a sample scenario, we can consider figure 3 where currently a client has information of 7 RCC that is collected from 7 different RM-Nodes.

From the analysis of the direction of movement of the RM-Node, client predicts that the RM-Node will visit either region A or region B. Therefore, it just sends the RCC of those two regions. In addition, in case the RM-Node does not hold the map, it can request the client for the map. Based on the map and collected RCCs, RM-Node estimates its possible time to meet a cloudlet. Suppose, the RM-Node's average speed is V and the RM-Node is planning to stop temporarily in S number of places with an average stopping time of T_s before meeting a cloudlet C at distance D according to the map. Then the time to meet the nearest cloudlet is, $T_c = D/V + S * T_s$.

One advantage of this approach is that the client does not need to reveal their destination address. Another advantage is that since RM-Node knows its future location, the velocity of moving, and intermediate stopping point, therefore, it can more accurately predict the time it might meet the nearest cloudlet. On the other hand, one major disadvantage of this approach is that predicting the closest cloudlet and the time to meet that cloudlet is done by RM-Node. Hence, the RM-Node consumes more power because of some additional computation. Another drawback is that RM-Node might intentionally deliver false

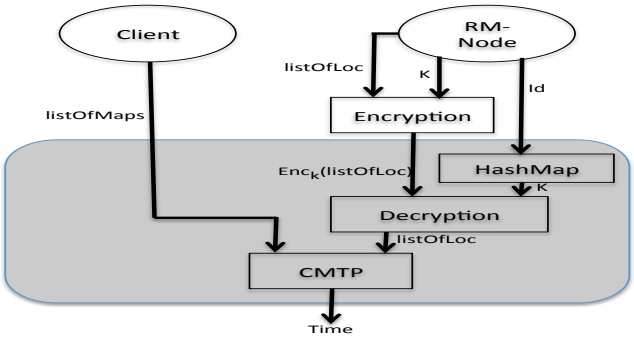


Fig. 4: Blackbox based approach for ensuring privacy during context sharing

information about the time of meeting the cloudlet to receive the content from the client. Since the client has no control over the computation, therefore, it is not possible to validate the information provided by the RM-Node.

Client based approach: The client predicts an RM-Node's nearest cloudlet and the time to meet that cloudlet based on the map and contextual information collected from the other RM-Nodes. We propose algorithm 1 for determining the time the RM-Node will meet its nearest cloudlets:

Algorithm 1: Cloudlet Meeting Time Prediction (CMTP)

Input: status
Input: listOfMaps
Input: angle
Output: time
var $reduceListOfMaps$=getReduceMapList($listOfMaps, angle$)
for $currentStatus$ in $status$ **do**
  for map in $reduceListOfMaps$ **do**
    var $listOfCloudLets$=map.getCloudLetList()
    for $cloudLet$ in $listOfCloudLets$ **do**
      if isLocInCloudLet($cloudLet$,$currentStatus$.getLocation()) **then**
        return $currentStatus$.getTime()
      end if
    end for
  end for
end for
return Null

Here, the status variable is a list of $< loc, time >$ pairs that indicate RM-Node's coordinate position in the map at every δt interval. These coordinates are matched with the cloudlet position in the collected RCC. If a cloudlet is found in the RM-Node's possible future path, the corresponding time is extracted from the $< loc, time >$ pair.

One major problem with this approach is that the client can track all the future location of an RM-Node. The lack of privacy for the RM-Nodes might reduce an RM-Node 's willingness to participate in the cloud formation. However, we can use the black box based approach as depicted in figure 4 to ensure the RM-Nodes' privacy.

As shown in figure 4, instead of providing the location information directly, a RM-Node now sends the encrypted location information. The pre-shared key K between the RM-Node and the CCA is used for encryption. The black box program is installed in every RM-Nodes' device. This program in addition to containing the CMTP algorithm also provides a hash table that maps each RM-Node's id with the pre-shared key. The black box is designed in such a way that the client neither can see nor can manipulate anything inside the black box. In other words, the black box only takes encrypted location information as input and delivers time as output without revealing any key or location information.

B. Evaluation of bidder's performance point (PP)

The PP varies between 0 and 1 based on RM-Node's performance on its previous tasks. Initially, the registered RM-Node receives an initial PP of 0.5. From the map and path information available to the client or the RM-Node, an approximate time t is computed for the RM-Node to deliver the content to a nearby cloudlet. Suppose an RM-Node delivers its content at t_O time. Then after the delivery, an award point (AP) is provided to the RM-Node as follows:

$$AP = \begin{Bmatrix} 1\ if\ t_O <= 2*t & 0.8\ if\ 2*t < t_O <= 4*t \\ 0.6\ if\ 4*t < t_O <= 6*t & 0.4\ if\ 6*t < t_O <= 8*t \\ 0.2\ if\ 8*t < t_O <= 16*t & 0\ otherwise \end{Bmatrix}$$

We use the strategy applied by ICC for their team rating to compute the RM-Node's new PP [31]. According to the strategy, the RM-Node's new PP will be increased a small amount if the AP obtained for the current task is better than its current PP. On the other hand, it will lose more point if the AP is smaller than the current PP. For the first scenario where we assume that the AP is more than the current PP, the new PP is evaluated using the following equation:

$$PP_{new} = min(1, PP + PP * (AP - PP))$$

We use the equation of exponential decay for the second scenario when the AP<PP and computes the new PP value as, $PP_{new} = PP * (e^{-\lambda * (PP-AP)})$.

RM-Node will try to perform more sincerely because losing more points for their bad performance implies it will receive less money for storing any content in future.

C. Formation of Ad hoc Cloud

At the first phase of the ad hoc cloud formation, the client broadcasts a node discovery message (NDM) for knowing the interested neighboring RM-Nodes and their status. The query lifetime (QL) is attached in the NDM that specifies the time by which an RM-Node has to response in order to participate in the cloud formation. The client queries the RM-Node for their PP, available storage, and the time to meet the nearest Cloudlet (node based approach) or the encrypted information of the coordinates (client based approach) once it receives the RM-Node's response within that time interval. The client uses the node selection algorithm as depicted in algorithm 2 and chooses some RM-Nodes as members of the ad hoc cloud.

Algorithm 2: Node Selection Algorithm

Input: listOfNodes
Input: listOfContents
Input: levelOfRelibility
Output: listOfAllocatedContentsToNodes
var $sortedListOfContents$=sortContentsBySensitivity($listOfContents$)
var $sortedListOfNodes$=sortNodesByPP($listOfNodes$)
for $content$ in $sortedListOfContents$ **do**
  var $sensitivity$=$content$.getSensitivity()
  var $currentSensitivity$=0
  while $sortedListOfNodes$.hasNext() & $currentSensitivity < sensitivity * levelOfReliability$ **do**
    var $currentNodes$=$sortedListOfNodes$.next()
    if $content$.getSize()<=$currentNode$.getAvailableResource() **then**
      $listofAllocatedContentsToNodes$.add(< $content, currentNode$ >)
      $currentNode$.takeResource($content$.getSize())
      $currentSensitivity$=$currentSensitivity$+$currentNode$.getPP()
    end if
  end while
end for

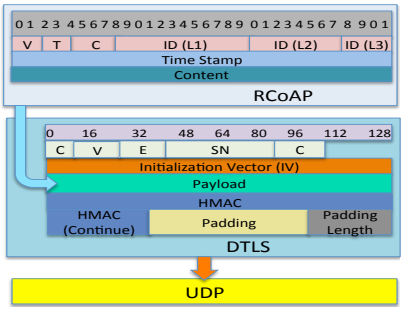


Fig. 5: Protocol Suite

At first, the client divides its contents into several segments and attaches a level of sensitivity with each division. The level of sensitivity indicates how much important that part is for the client. The level of sensitivity varies from 1 to 10. Additionally, the client specifies how much overall reliability it wants for the contents. The degree of reliability can be any number from 0 to 1. The number of RM-Nodes selected as a backup for a segment depends on its sensitivity and reliability. The client first sorts all the interested RM-Nodes according to their PP. The client chooses an RM-Node, verifies whether it has enough resource to hold the particular segment, and selects that RM-Node for offloading that segment if it has enough available resource. For the same content client might choose another RM-Node from the sorted list if the desired level of reliability is not achieved. For example, consider that the level of sensitivity for any content is 8, the degree of reliability that client wants is 0.2, and the size of the segment is 25 MB. Then the client can take two RM-Nodes with at least 25 MB of free storage in each of them and a PP of at 0.9 and 0.7. However, the client can pick three RM-Nodes with at least 25 MB of free storage along with a PP of 0.6 in each of them.

VI. Protocol and Message Format

A. Communication Protocol

We propose Reduced Constrained Application Protocol (RCoAP) that uses the similar packet structure as used in CoAP [32]. We adopt DTLS protocol, which is placed between the RCoAP and unreliable but efficient transport layer protocol UDP for ensuring transport security [33, 34]. UDP saves devices' battery power by allowing longer sleeping time for devices, sending low overhead small size packets, and ensuring more little triggering time for the wake-up and transmission. In addition to one-to-one message forwarding, RCoAP allows one-to-many transfer, which is required to store multiple copies of content in various RM-Nodes to achieve higher reliability. The proposed protocol suite is depicted in figure 5.

B. SASCloud Messaging Model

Each RCoAP message is comprised of three parts: header, time, and content.

Message Header: The message header has four sections as shown in figure 5. The first two bits define the version number of the RCoAP. The version number changes with the advent of newer protocol. The next two bits are used to identify the type of message. Just like the actual CoAP, there can be four types of message, such as Confirmable (00), Non-confirmable (01), Acknowledgement (02), and Reset (03). However, in SASCloud, every message is confirmable and requires acknowledgment from the receiver. Unlike CoAP, we use a fixed size token field, therefore SASCloud message header does not have any TKL field. The next four bits are termed as code field that is used to identify the category of a message (see Table I).

The rest of the 24 bits are used as message id to distinguish one message from another. In SASCloud, content received by an RM-Node can further be divided and distributed among a set of newly hired RM-Nodes in the next level. This process can continue to several levels. Therefore, a simple sequence number cannot be used as a message id for the divided contents. Another RM-Node can assign the same message id to another level. Hence, we apply the internet IP subnetting concept for message id generation. We allow a maximum of a three-level division of a message or content.

Initially, all the 24 bits of the message id is set to 0. When the content is divided for the first time into segments, a 12 bit sequence number is generated and assigned to the first 12 bit of the message id field for identifying those segments. If any of those segments is divided further (second level division and distribution) then the next 8 bit is used to these divisions. The remaining 4 bit is set if any of these contents are divided further on the third level. Hence a Cloudlet can easily identify the root of content which was divided and distributed on multiple levels. The Cloudlet merges such fragmented contents to construct the complete content.

Code	Interpretation (Category)
1	Rating Point Request
2	Rating Point Delivery
3	Map Information Request
4	Map Information Delivery
5	Request for Content Send
6	Confirmation to Content Send Request
7	Content Delivery
8	Content Reject

TABLE I: Code and Corresponding Representation

Message Timestamp: The 32-bit timestamp field is used to represent the sending date and time of a message.

Content: The variable length content field is used to specify user's contents. The size of the content field is determined during the content segmentation process.

VII. Security of SASCloud

In this section, we present the security requirement of SASCloud. We also identify the vulnerabilities and threats that could be associated with our proposed system. Furthermore, we provide mitigation strategies for all of those threats.

A. Security Requirements

1) Authentication: A malicious node can impersonate a legitimate RM-Node to perform malicious activities. A client needs to ensure that the RM-Nodes are trustworthy. The RM-Nodes also require verifying the authenticity of the client. This can be achieved through a mutual authentication scheme.

2) Confidentiality: In SASCloud, the contents of a client are stored in the neighboring RM-Nodes. It needs to be ensured that the contents are not disclosed to unwanted parties.

3) Integrity and content verification: An RM-Node can delete or modify the received content. A Client should be able to verify the integrity of the delivered content.

4) Non-repudiation: A client can deny a previously distributed content. Similarly, an RM-Node can deny the previously received content. Therefore, we need to have a scheme to verify such claims.

5) No Eavesdropping: The communications between a client and an RM-node should be secure. A malicious node can receive a content by eavesdropping and later can claim that the client hired the malicious node for storing the content. Therefore, communication channel needs to be secured.

B. Threat Model

Malicious RM-Node: An RM-Node might fail to deliver a content to the destination Cloudlet. The failing RM-Node will receive negative reward point (RP) by the Cloudlet. However, to avoid the negative RP, the RM-Node needs to prove that the client did not send any content.

Malicious Client: A malicious client can deny the content that it forwarded before to an RM-Node. Moreover, the client can defame the reputation of the RM-Node by not collecting the forwarded content from the cloudlet. In order to do that, the client needs to prove that the RM-Node has failed to deliver the content.

Malicious Cloudlet: A cloudlet might fail to transfer content to the client, which was given by an RM-Node. The failing cloudlet can claim that the RM-Node did not transfer the content for avoiding the penalty. The cloudlet can also argue that the client has received the content without actually delivering that content.

Colluding RM-Node and Cloudlet: A failing RM-Node can collude with a cloudlet to avoid penalty from the Client. The failing RM-Node can offer monetary incentives to cloudlet, so that the cloudlet does not assign negative performance point. Similarly, a failing cloudlet can also collude with an RM-Node to avoid negative reputation.

Colluding Client and Cloudlet: A malicious client can collude with a cloudlet to penalize an innocent RM-Node. Both the client and the cloudlet can claim that the victim RM-Node has failed to deliver content.

Colluding Client and RM-Node: A malicious RM-Node can argue that it has received content from the client and provided the content to the victim Cloudlet. The malicious client supports the false claim. Here, both the client and the RM-Node try to defame the reputation of the innocent cloudlet.

C. Mitigation Strategy

A client would like to ensure the confidentiality and the integrity of the distributed contents. Therefore, content can be encrypted before delivering to any hired RM-Node to ensure confidentiality of the encrypted content. To ensure the integrity of the big data in a cloud, Liu et al. proposed a digital signature based scheme [35]. We also apply a similar but lighter digital signature based strategy in SASCloud to mitigate the threat scenarios. Figure 6 shows the interactions when a Client transfer any content to an RM-Node. The interactions during the content delivery process by the RM-Node, and the content receiving process by the Client are presented in figure 7.

Content Transfer Process: In step 1 and 2, the client and the RM-Node perform certificate-based mutual authentication to authenticate each other. In step 3, the client and RM-Node establish a secure channel using a shared key for further communications. In step 4 and 5, the client encrypts the content and forwards the encrypted content to the RM-Node. In step 6,

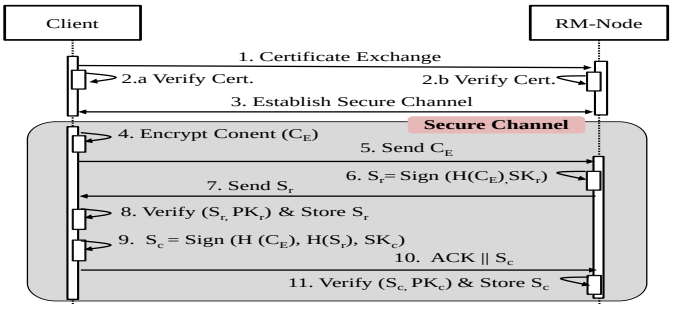


Fig. 6: Secure content transfer security at origin

the RM-Node creates a hash ($H(C_E)$) of the received content and signs the hash using its private key (SK_r). In step 7, the RM-Node sends an acknowledgement (ACK) and attaches the signature (S_r) with the ACK. In step 8, the client verifies the signature using the RM-Node's public-key (PK_r) and ensures that the RM-Node receives the entire content. In step 9, the Client creates a signature (S_c) using its private key (SK_c). The inputs of the signature scheme are the hash of the content and the hash of the signature of the RM-Node. In step 10, the client sends a signed acknowledgment to the RM-Node. In step 11, the RM-Node verifies the signature using the client's public key (PK_c) and stores S_c for future reference. The RM-Node also stores the certificate of the client ($Cert_c$).

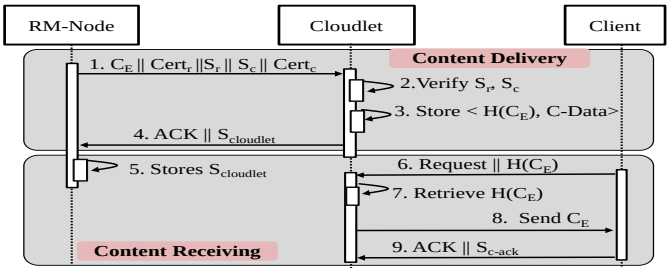


Fig. 7: Secure content delivery and receiving security at destination

Content Delivery and Receiving Process: In step 1, the RM-Node delivers the encrypted content, and signatures and certificates of the RM-Node and client to the cloudlet. In step 2, the cloudlet verifies the signatures of the RM-Node and client. The cloudlet discards the content if any of the signatures does not match. In step 3, the cloudlet stores the hash and the metadata of the content ($C - Data$), such as RM-Node's id, C_E, signatures of the Client and the RM-Node. In step 4 and 5, the cloudlet creates a signature ($S_{cloudlet} = Sign(H(H(C_E)||C - Data), SK_{cloudlet})$), and sends the $S_{cloudlet}$ to the RM-Node. The RM-Node stores the $S_{cloudlet}$ as proof that the cloudlet has received the content. In step 6, the client sends a request message to the cloudlet along with its certificate and hash value of the content. In step 7, the cloudlet retrieves the $C - Data$ using the provided hash and verifies S_c. In step 8 and 9, the cloudlet delivers the content to the client and receives a signed acknowledgment (S_{c-ack}). The cloudlet also stores the signature as proof that the client has received the content.

D. Security Analysis

The failing RM-Node cannot claim that the client did not offload the content, since the client stores the content

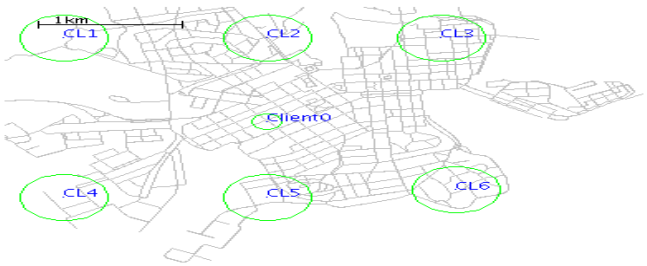


Fig. 8: Simulation Environment

forwarding proof S_r. Besides, the RM-Node cannot defame a cloudlet as it does not have the content delivery proof, $S_{cloudlet}$.

The malicious client cannot deny the content the client forwarded to an RM-Node or the content that it received from a cloudlet. The RM-Node has S_c as a proof for the forwarding content. Similarly, the cloudlet has the evidence S_{c-ack} for the received the content. A failing cloudlet cannot claim that the client received the content for avoiding the penalty since the cloudlet cannot show the client's signature S_{c-ack}.

For the same reason, the colluding RM-Node and the cloudlet cannot charge an innocent client, who did not receive the forwarded content. The colluding client and the cloudlet cannot penalize an innocent RM-Node, since the RM-Node stores the signature $S_{cloudlet}$. The colluding client and the RM-Node cannot defame an innocent cloudlet, since the RM-Node would not be able to show the signature $S_{cloudlet}$.

VIII. Simulation Setup

We design our model using the Opportunistic Network Environment (ONE) simulator, a Java-based open source simulator for delay tolerant network [36]. The RM-Nodes communicates with each other using wifi interface that has a transmission range of 50 meters. On the other hand, the cloudlet has a transmission range of 500 meters. We assume that the client is stationary and does not belong to the coverage area of any cloudlet whereas the RM-Nodes are moving randomly 2 to 4 m/s speed with a pause time of 1 to 120 minutes at some random intervals. Our model is simulated using the Helsinki city map, and we assume that the boundary of the city is covered by six cloudlets whereas there is no cloudlet in the center part of the city. Our simulation environment is shown in figure 8.

We assume that client has 1000 contents each with 64 KB size and with different level of sensitivity. The RM-Nodes' PPs are also randomly generated between 0 to 1. We use Raspberry Pi devices with different configurations, such as Pi-1 and Pi-2, as Clients and RM-Nodes. We use a power meter to record the power consumption for sending and receiving a discovery message of size 128 bytes. We find that the power drop for sending the discovery message is from 10mW to 20 mW. Next, we record the transmission time for sending a maximum sized UPD message of length 64KB. We also record the power consumption for carrying the maximum sized UDP payload. We find that the transfer time varies from 0.1 to 0.2 seconds, while the corresponding power consumption lies in between 100 to 200 mW. We use this information to measure the time and the cost associated to content offloading. We provide monetary incentive corresponds to the power consumption cost during the content transferring process. Additionally, we assume that an RM-Node receives $10 * P$ amount of financial incentive for holding a content of 64KB size for one minute, where P is the performance point of the RM-Node. In our first experiment, we measure the performance of SASCloud by varying the number of RM-Nodes. We consider the RM-Node size, 500,1000,1500, and 2000. The result is shown in figure 9.

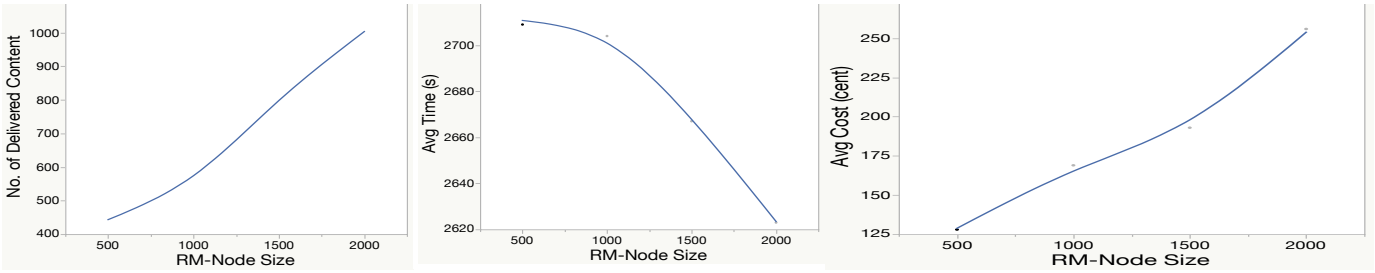
From figure 9, we see that adding more number of RM-Nodes increases the number of delivered contents. On the other hand, the average cost of offloading the contents also increases with increasing size of RM-Nodes. However, the number of RM-Nodes has no effect on the content transfer time. Since we just form a single level of ad hoc cloud, therefore, the progression to meet a cloudlet for any RM-Node is very slow. Moreover, for simplicity in our simulation, instead of using the CMTP algorithm for RM-Node selection during content distribution, we just followed the greedy based approach for selecting the RM-Nodes. Hence, the content delivery time does not change with RM-Node size and is larger than expected.

In our second experiment, we assume that we have 1000 number of RM-Nodes and an RM-Node can fail at any time while carrying a content. If any node fails, it will lose all the contents that it carries. The probability that an RM-Node will fail depends on its PP. If the probability of failure is 0.5 then the likelihood that an RM-Node would fail to deliver its contents given that its PP value is p is, 0.5*(1-p). We measure the total number of successfully delivered contents and the total number of lost contents considering the above-mentioned failure model. Next, we increase the level of redundancy during content delivery to 2, 3, and 4 times and measure the total successfully delivered contents and total lost contents. The corresponding graph is shown in figure 10.

Figure 10 shows that we can deliver a higher number of contents if we consider smaller redundancy during content distribution. However, the ratio of total lost contents compared to the total distributed contents also increases when we consider less redundancy. The reason is that for large redundancy, the client needs to find a sufficient number of RM-Nodes before sending the contents to each of them. Since our RM-Node size is fixed, therefore, the client does not get enough number of nodes for content offloading. On the other hand, since a larger number of RM-Nodes receive the content, it can still reach to the cloudlet when one or more RM-Nodes fail. Moreover, the average cost for offloading each unit of content becomes higher with higher redundancy as client pays money to more number of RM-Nodes than before for the additional replication.

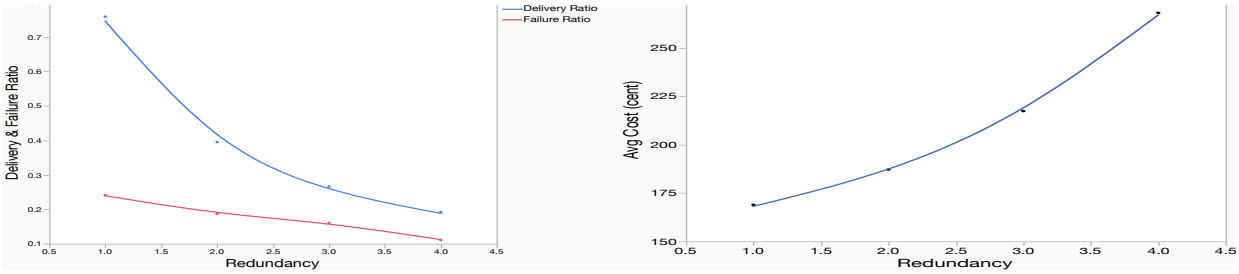
IX. Conclusion

In this paper, we presented a novel centralized-control ad hoc cloud system that provides storage service in a delay tolerant network. We proposed an efficient RM-Node selection strategy based on the RM-Node's contextual information and existing knowledge of some parts of the world map. We introduced a reduced version of CoAP and designed a message format for the SASCloud to make it secure and energy efficient during content distribution. We addressed all the possible threats in our proposed model with the countermeasures. Our experimental result based on a simulated environment of the SASCloud showed that building such model is feasible. However, our simulation is based on the prototype of the



(a) Total Delivered Content (b) Average Delivery Time (c) Average offloading cost

Fig. 9: Performance of SASCloud with different node size



(a) delivery and failure ratio (b) Average offloading cost

Fig. 10: Performance of SASCloud with different redundancy value

SASCloud. Therefore, RM-Node size has rarely any effect on content delivery time. The time would be much smaller if we would consider multiple levels of ad hoc cloud. Also, for simplicity, we did not implement our proposed CMTP algorithm in the simulation. CMTP will reduce a significant amount of content delivery time along with subsiding the content failure to total distributed content ratio. As a future work, we are planning to develop a protocol for efficiently retrieving the contextual information of RM-Node and efficient representation of the collected map information so that the contextual information can be easily linked to the map during the deployment of CMTP algorithm.

X. Acknowledgements

This research was supported by the National Science Foundation CAREER Award CNS-1351038.

References

[1] H. T. Dinh, C. Lee, D. Niyato, and P. Wang, "A survey of mobile cloud computing: architecture, applications, and approaches," *Wireless Communications and Mobile Computing*, vol. 13, no. 18, 2013.

[2] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *MCC Workshop*, 2012.

[3] G. McGilvary, A. Barker, and M. Atkinson, "Ad hoc cloud computing," *CoRR*, 2015.

[4] X. Fan, J. CaoFan, J. Cao, and H. Mao, "A survey of mobile cloud computing," http://wwwen.zte.com.cn/endata/magazine/ztecommunications/2011Year/no1/articles/201103/t20110318_224532.html, 2011, zTE Communications.

[5] "Boincoid - an android port of the boinc platform," http://boincoid.sourceforge.net/.

[6] B. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti, "Clonecloud: Elastic execution between mobile device and cloud," in *Computer Systems*, 2011.

[7] S. Al Noor, R. Hasan, and M. Haque, "Cellcloud: A novel cost effective formation of mobile cloud based on bidding incentives," in *IEEE Cloud*, 2014.

[8] H. Ba, W. Heinzelman, C.-A. Janssen, and J. Shi, "Mobile computing - a green computing resource," in *WCNC*, April 2013.

[9] N. Fernando, S. W. Loke, and W. Rahayu, "Dynamic mobile cloud computing: Ad hoc and opportunistic job sharing," in *UCC*, Dec 2011.

[10] G. Fortino, D. Parisi, V. Pirrone, and G. Fatta, "Bodycloud: A saas approach for community body sensor networks," *Future Generation Computer Systems*, vol. 35, 2014.

[11] C. Doukas and I. Maglogiannis, "Bringing iot and cloud computing towards pervasive healthcare," in *IMIS*, 2012.

[12] S. Chatterjee and S. Misra, "Target tracking using sensor-cloud: Sensor-target mapping in presence of overlapping coverage," *IEEE Communications Letters*, 2014.

[13] N. Zingirian and C. Valenti, "Sensor clouds for intelligent truck monitoring," in *IEEE Intelligent Vehicles Symposium (IV)*, 2012.

[14] M. Arrington, "Gmail disaster: Reports of mass email deletions," http://techcrunch.com/2006/12/28/gmail-disaster-reports-of-mass-email-deletions/.

[15] M. Krigsman, "Apple's mobileme experiences post-launch pain," http://www.zdnet.com/article/apples-mobileme-experiences-post-launch-pain/.

[16] M. Shiels, "Phone sales hit by sidekick loss," http://news.bbc.co.uk/2/hi/technology/8303952.stm.

[17] S. Sultan, M. Doori, A. Al-Bayatti, and H. Zedan, "A comprehensive survey on vehicular ad hoc network," *Journal of Network and Computer Applications*, vol. 37, 2014.

[18] M. Prasad, J. Gyani, and P. Murti, "Mobile cloud computing: Implications and challenges," *Journal of Information Engineering and Applications*, vol. 2, no. 7, 2012.

[19] S. Dashti, J. Reilly, J. D. Bray, A. Bayen, S. Glaser, M. R. Ervasti, and N. Davies, "Using personal devices to deliver rapid semi-qualitative earthquake shaking information," *GeoEngineering Report*, 2011.

[20] A. Mednis, G. Strazdins, R. Zviedris, G. Kanonirs, and L. Selavo, "Real time pothole detection using android smartphones with accelerometers," in *DCOSS*, 2011.

[21] P. Angin, Bharat, and K. Bhargava, "Real-time mobile-cloud computing for context-aware blind navigation," *IJNGC*, vol. 2, no. 2, 2011.

[22] D. Hoang and L. Chen, "Mobile cloud for assistive healthcare (mocash)," in *APSCC*, 2010.

[23] "Amazon mechanical turk," https://www.mturk.com/.

[24] H. Lieberman and et al., "Common consensus: a web-based game for collecting commonsense goals," in *IUI*, 2007.

[25] P. G. Ipeirotis, F. Provost, and J. Wang, "Quality management on amazon mechanical turk," in *HCOMP*, 2010.

[26] R. Gao, M. Zhao, T. Ye, F. Ye, G. Luo, Y. Wang, K. Bian, T. Wang, and X. Li, "Multi-story indoor floor plan reconstruction via mobile crowdsensing," *TMC*, vol. 15, no. 6, 2016.

[27] R. Hasan, M. M. Hossain, and R. Khan, "Aura: An iot based cloud infrastructure for localized mobile computation outsourcing," in *IEEE Cloud*, 2015.

[28] C. Funai, C. Tapparello, H. Ba, B. Karaoglu, and W. Heinzelman, "Extending volunteer computing through mobile ad hoc networking," in *GLOBECOM*, 2014.

[29] E. Miluzzo, R. Cáceres, and Y.-F. Chen, "Vision: Mclouds - computing on clouds of mobile devices," in *MCS*, 2012.

[30] S. A. Noor and R. Hasan, "D-cloc: A delay tolerant cloud formation using context-aware mobile crowdsourcing," in *CloudCom*, 2015.

[31] J. Beck, "How to calculate icc rankings," http://www.ehow.com/how_6916968_calculate-icc-rankings.html.

[32] Z. Shelby, K. Sensinode, C. Hartke, and Bormann, "The constrained application protocol (coap)," https://tools.ietf.org/html/draft-ietf-core-coap-18, 2013.

[33] T. Kothmayr, C. Schmitt, W. Hu, M. Brunig, and G. Carle, "Dtls based security and two-way authentication for the internet of things," *Ad Hoc Networks*, vol. 11, no. 8, 2013.

[34] M. Luk, G. Mezzour, A. Perrig, and V. Gligor, "Minisec: A secure sensor network communication architecture," in *IPSN*, 2007.

[35] C. Liu, C. Yang, X. Zhang, and J. Chen, "External integrity verification for outsourced big data in cloud and iot: A big picture," *Future Generation Computer Systems*, vol. 49, 2015.

[36] "The opportunistic network environment simulator," http://www.netlab.tkk.fi/tutkimus/dtn/theone/.