LOCAL OVERLAY-BASED MOBILE CLOUDS

by

Lavanya Tammineni, B.E.

A thesis submitted to the Graduate Council of
Texas State University in partial fulfillment
of the requirements for the degree of
Master of Science
with a Major in Computer Science
December 2015

Committee Members:

Mina S. Guirguis, Chair

Xiao Chen

Qijun Gu

FAIR USE AND AUTHOR'S PERMISSION STATEMENT

Fair Use

Duplication Permission

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my advisor Dr. Mina S. Guirguis for his patience, motivation, and continuous support without which I would not be involved in research. His guidance and immense knowledge helped me in all the time of research and writing of this thesis.

My sincere thanks to the rest of my thesis committee members, Dr. Qijun Gu and Dr. Xiao Chen for their availability and insightful comments.

Lastly, I thank my family and friends for their support and encouragement throughout my graduate studies.

TABLE OF CONTENTS

Page

LIST OF TABLES

Table **Page**

LIST OF FIGURES

Figure **Page**

ABSTRACT

The growing capabilities of mobile devices provide a rich computing platform for various applications. Most of these devices have the ability to form a mobile cloud among themselves without relying on an existing infrastructure or a centralized administration whereby each device offer its capabilities as a cloud service for other devices to use. Mobile clouds are useful in scenarios where the deployment of access points is not feasible or expensive (e.g., natural disasters, heavily crowded areas, military operations). In this thesis, we propose a mobile cloud that capitalizes on locality.
The mobile cloud provides an efficient collaborative computing framework that enables a group of mobile devices to form an ad-hoc network based on their proximity and advertise their capabilities as cloud services. We propose a hybrid routing algorithm through the combination of reliable flooding and Distributed Hash Tables (DHT) to find the closest nodes that have the desired capabilities for task execution. We evaluated our proposed platform through simulations, and we studied the impact of different factors such as the number of lookup messages generated and the number of lookup hops taken until a response is received. Our results show the feasibility of our proposed mobile cloud platform

I. INTRODUCTION

Cloud computing enables users to consume computing resources as services rather than having to build and maintain computing hardware and infrastructures in-house. Cloud computing utilizes the network of remote servers which are hosted on the Internet to store, manage and process data instead of using the local computer or servers. Similarly, mobile cloud computing is becoming popular these days due to the increase in the use of smart devices (e.g. smartphones) which have tremendous processing, storage and networking capabilities. The devices can form a rich collaborative computing platform by sharing their resources as cloud services among themselves. However, the limited battery life of user devices has dictated the heavy lifting to be pushed out to dedicated servers at the data centers [12-14] and limiting user devices for low power processing needs. While this is a feasible solution, but it depends on network connectivity to the servers, which may not be available in many scenarios (e.g. natural disasters, disturbance in heavily crowded environments, military conflicts, etc.). To address the above challenge, mobile devices should be able to form an on demand ad-hoc network, configure its parameters and provide their capabilities as cloud services. Devices in the network can utilize these cloud services offered by peers to offload the work.

Moreover, every device will not have similar capabilities. For example, some devices may have huge storage space, but don't have long lasting battery life, some may have multiple cores for performing computations, but don't have huge storage space, etc. So a device cannot perform some computation as the software or hardware on that device doesn't support the requirements. A device can overcome its own limitations by taking

advantage of the capabilities of other devices surrounding them by offloading work. This has led to a lot of research in the area of mobile cloud computing over ad-hoc networks focusing on partitioning and distributing the work among the devices efficiently [15 - 17]. These works provide the code distribution models and dynamic cost adjustment for task assignments to provide properties such as load balancing and collocating tasks for execution. Our key focus in this thesis is on reducing the time to find one of the nearest nodes with a required capability in the ad-hoc network so that the request can be routed locally. The social and context awareness of nearby devices can be utilized to complete the work efficiently.

To achieve this goal, in this thesis we present "Local Overlay-based Mobile Clouds" which is a mobile cloud that capitalizes on locality**.** The underlying devices form an ad-hoc network to share their resources as a cloud service and form a rich collaborative computing platform. Also, by finding and offloading work to a nearby capable node, lookup times, as well as the number of hops traversed, are reduced. As a result, the average task completion time is reduced along with bandwidth usage.

Without any central administration or infrastructure, individual nodes in an ad-hoc network need to determine the routing path to find the capable nodes with available resources, offload their work and get the results back. There have been some Android prototype implementations to form ad-hoc networks recently, one of the open source that we explored is SPAN [11]. The routing protocols supported by this prototype are Optimized Link State Routing (OLSR) [19] and simple proactive protocol using the shortest path algorithm (Dijkstra). If the destination node is known, these supported

protocols can be used to identify the route. But to find a device in the network that has the desired capability for task execution, flooding based search technique need to be used, which yields results quickly when the required capability is highly available. However, flooding is poorly suited for finding rare capabilities. Structured peer-to-peer (P2P) overlays such as Chord [7], CAN [8], Pastery [9], and Tapestry [10] provide a scalable, highly available robust network and ensure that any node can search the network for a resource efficiently. In the structured P2P, node IDs are assigned by picking a random identifier from a hash space using variants of consistent hashing (e.g. SHA1 [5]). One problem with this simple DHT structure is that the overlay topology is not congruent with the physical topology due to the random nature of node IDs. As a result, the number of messages to lookup a capability can be high. Our main goal is to find one of the nearest devices that is capable of executing the task to utilize the social and context awareness of that device. This goal can be achieved if the logical distances in the overlay reflect their physical distances between the devices on the ad-hoc network. This can be made possible by exploiting the locality information of devices while assigning node IDs in DHT. So, we are proposing a method to interleave the Global Positioning System (GPS) coordinates i.e. latitude and longitude values of the devices, which will be used as a prefix for the node ID in the overlay space along with a suffix of random bits.

One other important aspect in Chord is maintaining the resource information. A key associated with a resource obtained using variants of consistent hashing is mapped onto the node with highest ID less than or equal the key, and that node is responsible for that

resource. The number of responsible nodes is critical to ensure high availability and for the responsible node to be closer to all the nodes in the network. We are proposing to divide the hash space into regions and pick the responsible nodes in each region based on the replication count chosen by the user as one of the configuration parameters.

With structured peer-to-peer networks, one of the nearest capable devices is found efficiently with less bandwidth usage but they incur higher overhead in lookup time to find the highly replicated resources when compared to flooding based search techniques. To take advantage of both the flooding and DHT, there have been some proposals for hybrid protocols [30 – 32]. In all of these protocols, some of the devices are identified to maintain the statistics related to resources (i.e. rare or popular) among the devices in the network. Based on those statistics a routing algorithm (flooding or DHT) is selected, which is not feasible for dynamic topologies as the overhead to maintain statistics will be high, even though those protocols are decentralized and adaptive.

So, in this thesis, we present a hybrid routing algorithm using the combination of reliable flooding and DHT to find one of the closest nodes that has desired capabilities for task execution in the mobile cloud. We evaluated our proposed platform through simulations. We studied the impact of different methods on various performance metrics such as the average number of lookup messages and the average number of hops taken until a response is received under different node arrival and departure rates. Our results show the feasibility of our proposed mobile cloud computing platform.

Contributions: In this thesis, we make following key contributions:

1. We present a mobile cloud which provides a collaborative computing framework that enables mobile devices to form an ad-hoc network based on their proximity and advertise their capabilities as cloud services.
2. We present a new hashing mechanism that utilizes the latitude and longitude values of the mobile devices for node ID assignment to induce congruity between the physical and overlay topologies.
3. We propose a hybrid routing algorithm through the combination of reliable flooding and Distributed Hash Tables (DHT) to find the closest node with the desired capabilities for task execution.
4. We evaluate the proposed framework through simulations under different arrival rates following Poisson distribution and ON period following Pareto distribution.

Thesis Organization: The thesis is organized as follows: Section II describes the background information and related work. In Section III, we present the proposed system model, device architecture, hashing algorithm based on GPS coordinates, and the hybrid routing protocol. We present the simulation results for evaluation in Section IV followed by our conclusion and future work in Section V.

II. RELATED WORK

The work in this thesis relates to a number of areas of active research. In this section, we present the related body of work and put our work in context.

2.1 Mobile Cloud Computing (MCC)

A significant amount of work is focused on mobile cloud computing to offload processing to external servers, so as to reduce the workload on the device. MAUI [13] enabled automated offloading. Developers just need to decide which functions to be offloaded and the offloading engine will do the work to ship computation to a remote server. COMET [12] is another platform in which threads are allowed to migrate between machines depending on workload. Dpartner [14] analyzes the Android application bytecode and rewrites it to support the on demand computation offloading. Collaborative Computing On-Demand [16] proposes a collaborative computing framework that maps a job as a set of tasks to execute onto the nearby mobile devices by considering different factors like, requirements of the job, capabilities of the devices, dependencies of the tasks, and adjacency of the devices. All the above research concentrates on communication and execution of tasks but not so much on finding the capable devices for executing a task. In [15], a probabilistic code distribution model that enables a mobile device to execute code segments on nearby mobile devices has been proposed. In Transient Clouds [17], task allocation based on capabilities of the devices in the ad-hoc network is done in a centralized approach where one device acts as a group leader since the Android prototype relies on the Wi-Fi Direct framework which assumes a group leader. This existing work focused on how to partition and distribute code segments

efficiently. In [12-14], computation is offloaded to the public server that requires access to infrastructure, which may not be available in many scenarios. In this thesis, we focus on establishing a mobile cloud that capitalizes on locality to induce congruity between the physical and overlay topologies. It enables nodes in the network to find a device capable of executing the task that is nearest in physical topology and to route queries locally.

2.2 Mobile ad-hoc networks (MANET's) and routing over ad-hoc

MANET is an ad-hoc wireless network created on the fly by a group of mobile devices. A node form links with other nodes that fall within its radio range. If this node falls into other's radio range, then the link between them will be bidirectional or else remains unidirectional. Links between any nodes can be unidirectional or bidirectional. If a node moves out of range, the broken links can be easily reestablished by periodic communication between the nodes or a new route can be reestablished reactively when the request is raised. Similarly, when a node leaves the network, affected nodes can request new routes.

Every node in the MANET acts as both a host and a router. As the transmitting range of the devices is limited to be able to reach all the nodes, it may be necessary to take several hops. Without any central administration or infrastructure, the nodes in the network need to determine the routing path. Basic routing protocol for ad-hoc networks is reliable flooding, and the most adapted protocols specific to ad-hoc networks are: OLSR (Optimized Link State Routing), DSR (Dynamic Source Routing) and AODV (Ad-hoc On-Demand Distance Vector). OLSR is a proactive protocol in which routes are

evaluated continuously when a packet needs to be forwarded, the route is already known. AODV and DSR are reactive protocols in which the routes are determined on demand.

2.2.1 Reliable Flooding: Sequence Number Controlled

In sequence number controlled flooding [18], every packet will have a sequence number, time to live (TTL), and the IP address of the source. The nodes that receive the packet will forward it to their neighbors if the packet is not expired; the node has never seen the packet before, and the sequence number of the packet is greater than the sequence number of the packets that are already received from that node. To achieve these goals, each node maintains a list of highest sequence numbers seen from other nodes. The sequence number will get incremented each time a new packet is sent out so that any packet that have a sequence number less than the latest will be obsolete and those packets will be dropped. The packet will not be rebroadcasted to the node from which the packet is received.

2.2.2 Optimized Link State Routing Protocol (OLSR)

OLSR [19] is a proactive routing protocol, and it is an optimization of link state protocol for mobile wireless networks. The Hello messages are exchanged periodically by nodes to detect their neighbors. In OLSR, a multipoint relay (MPR) is selected which take care of flooding the topology information and only MPRs will forward the topology control (TC) messages. This topology information is used by the nodes to calculate the next hop destinations to all the other nodes in the network. Every node maintains a routing table with routes to all the other nodes in the network.

2.2.3 Ad-hoc On-Demand Distance Vector Routing (AODV)

AODV [20] is a reactive protocol. When a node wants to reach another node, it sends route request (RREQ) to all its neighbors. RREQ propagates through the network until it reaches the destination or a node with a route to the destination. The routes to the destination are calculated and the route that has the least number of hops will be used by the source node to send the packet. Intermediate nodes calculate the routes upon request, record the packets they have seen and maintain the temporary route to the source node for the route reply packet (RREP). Unused routes will be deleted after some time interval. The nodes forward the route requests only if they have not forwarded the packet before, the sequence number of the packet is larger than the previously seen packets from that source and the packet's time to live is not expired so that, the number of messages will be reduced and the network bandwidth usage is decreased. If an intermediate node moves then that node's neighbors send a link failure notification to its upstream neighbors till it reaches the source upon which source can reinitiate the request, if needed.

2.2.4 Dynamic Source Routing (DSR)

Dynamic Source Routing [21] is a source-routing on-demand protocol that forms routes on demand. In source routing, the header of the packet will have the ordered list of nodes that need to be traversed. Every node caches source routes into route cache and updates them as and when it learns new routes. When a node wants to reach any particular destination, it searches in the route cache, and if none is available, then route request (RREQ) is raised. Intermediate nodes process RREQ if they have not been seen before. A

RREP is generated when RREQ reaches the destination node or when a node with an unexpired route to the destination node exists in its cache.

There are some works comparing the performance of these protocols under different network traffic patterns. According to [22 - 24], OLSR is better regarding end-to-end delay and data delivery ratio even though OLSR routing load is higher compared to AODV and DSR protocols.

Recently, there have been some works on specific implementation of MANET's using Android devices. In Android 4.0 APIs [2], Wi-Fi Direct enables devices to connect among themselves without a wireless access point. But Wi-Fi direct assumes that one mobile device acts as group leader (centralized) which does not scale. There is some other interesting work i.e. SPAN [11], which presents "MANET Manager" app for Android devices that enables the devices to form an ad-hoc network based on transmission ranges. There is no central administration, and OLSR is used as the routing protocol. So, each device in the network will maintain the complete topology information. This work can be utilized to form the ad-hoc network to communicate with each other. In addition to establishing a network, we want the devices in the ad-hoc network to be able to share their resources with the other devices in the network as cloud services.

2.3 Peer-to-Peer (P2P) networks

Peer-to-peer networking [6] is a paradigm shift from the client-server model to a more decentralized one. Every device in the network is equipotent and can act as either a client or a server in the application. Nodes in these systems share a portion of their resources

such as processing power, storage files, etc. with peers in the network. P2P networks are capable of self-organizing, and they are organized as overlay topologies on top of the underlying physical topology either in unstructured or structured manner.

In unstructured P2P networks, nodes forming connections with each other will form the overlay without any predefined structure. These networks can be built and maintained easily as there is no definite structure that need to be followed leaving no option than flooding to search for a resource, which consumes a significant amount of bandwidth. Structured P2P networks solve this problem by imposing structure for the overlay to ensure that any node can find the resources in the network efficiently. Most commonly, structured P2P networks implement distributed hash tables.

2.3.1 Distributed Hash Table (DHT)

DHT provide a lookup for the nodes in the network with a key value pairs similar to a hash table. Any node in the network can retrieve the value with the given key. The responsibility of maintaining the key value pairs is distributed among all the nodes in the network so that there will be a minimal disruption when a node joins or leaves the network. This makes DHT scalable to any extent. DHTs use some variant of consistent hashing to assign keys to nodes, because of which keys and nodes are distributed uniformly in the identifier space with a negligible possibility of a collision.

Many P2P networks adopt DHTs such as Chord [7], CAN [8], Pastery [9], and Tapestry [10]. In this thesis, we organize the cloud based on the Chord framework. Chord is one of the popular solutions for building an overlay, mainly because of its simplicity and efficiency. Chord uses basic consistent hashing technique (e.g. SHA1 [5]) to assign keys

to nodes. Nodes are arranged in a virtual ring that has at most 2^n nodes from 0 to $2^n - 1$ for an n-bit ID space. Each node maintains a portion of the Chord, which contains its successor, predecessor and itself. Every node with ID i, will maintain a finger table to other nodes with up to n fingers from $(i + 2^k) \bmod 2^n$ where k is an integer ($0 \leq k < n$).

Due to the random nature of the node ID assignment, there will be no correlation between the overlay topology and physical topology. As a result, a single hop on the overlay may correspond to multiple hops on the physical layer which mean that the number of hops the overlay messages traverse in the physical topology can be higher. To address this challenge, there has been some works [25 -29] on location based node ID assignment for the overlay topology to be congruent with the physical topology.

2.3.2 Location based Chord

MESHCHORD [25], and WILCO [26] are variations of Chord proposed specific to wireless mesh networks. The node ID assignment is based on the assumption that mesh routers will be mostly static and mesh clients will be dynamic. Most significant bits of the mesh client ID are based on the mesh router to which it is connected. In LDHT [27], most significant bits of the node ID is assigned according to the Autonomous System Numbers (ASNs) based on the domain of the device relying on the network connection that may not be available in many scenarios. On the other hand, the hashing technique that we are presenting in this thesis do not rely on the network connection and each device in the network is considered the same.

In Location Based Node ID [29], the node ID consists of a prefix of hierarchical components and a suffix of randomly generated bits. The node ID structure is:

Node ID = Region + Subregion + Leaf region + Random bits

Spatial regions are determined by using a space filling Hilbert curve.The boundaries between the different regions are not fixed. The prefix region length is determined based on the density of the nodes in a region. The entire space is divided into regions based on the density, and the prefixes are assigned based on abstractions similar to sub-netting and super-netting. This could be effective in inducing congruity between the overlay topology and the physical topology for static networks with a centralized administration. To divide the hash space into regions, the geographical information should be maintained in a centralized database or locally at each node. A centralized database is not feasible in an ad-hoc network and maintaining such information at each node is too costly. Moreover, when the network is dynamic, regions should be re-divided, and the node IDs need to be frequently reassigned, which will consume significant processing power and bandwidth for the devices.

Another recent work, Geo-Chord [29] constructs Chord ring based on geographical location dividing the nodes into different subregions based on the Euclidean distance. Each subregion ring head node will form the main Chord ring. Subregion rings are constructed based on a Euclidean distance matrix and a hop distance matrix of the nodes in the network which will be well suited for static networks than dynamic topologies as the matrices keep changing, and the regions should be re-divided accordingly.

With DHT based protocol Chord as the overlay, every node can find any resource efficiently with significantly lower bandwidth usage but the lookup time will be increased relatively when compared to flooding. To take advantage of DHTs efficiency in finding

rare resources and avoid the higher overhead in lookup time when finding popular resources compared to flooding based search techniques, there has been some works on hybrid protocols. Those were proposed to overcome the disadvantages of unstructured network designs like Gnutella [3] and Kazaa [4] which use flooding based searching techniques. In loo-hybrid [30], some of the devices are identified as ultrapeers, and they are stored in the overlay DHT. Each ultrapeer is responsible for finding the rare resources from its leaf nodes. Search is performed using conventional flooding technique among the neighbors on the overlay, if enough results are not returned in specified time, the query will be issued in DHT. This approach, however, can incur higher cost in dynamic topologies where the nodes keep joining and leaving as ultrapeers need to maintain rare and popular capabilities matrix from their leaf nodes.

Other works on hybrid routing, Adaptive Peer to Peer Search [31] and GAB[32] propose collecting the global statistics to select the routing technique unlike Loo [30] where the query is flooded first and if it fails, then the query is issued in DHT. The global statistics (i.e. resource is rare or popular) need to be updated when a node joins or leaves the network. Global statistics are maintained by some nodes that are identified as global index nodes [31] or ultrapeers [32]. Nodes that have long uptime and good connectivity are identified as global Index nodes or ultrapeers and mapped to the overlay DHT that maintains a list of nodes having resources. This approach is not feasible in an ad-hoc network as a device's uptime is highly unpredictable, and maintaining the global statistics accurately will consume a significant amount of bandwidth.

In this thesis, we are presenting a hybrid routing algorithm through the combination of reliable flooding and Distributed Hash Tables (DHT) to find the closest nodes with desired capabilities for task execution which does not require maintaining global statistics regarding resources (i.e. rare or popular) or identifying some nodes as ultrapeers to maintain the statistics.

III. PROPOSED FRAMEWORK

3.1 System Model

An ad-hoc network is formed by a group of mobile devices in a region based on their proximity and thus, they form a mobile cloud. The devices can range from personal digital assistants to laptops that have support for wireless communication. We let D denote the set of devices in the network, $D = \{d_1 d_2 \ldots d_i \ldots d_m\}$, where d_i represents each device in the network, and m is the total number of devices. We let λ denote the average arrival rate of devices and β denote the average ON period. Every device, joining the network will have a set of capabilities which are provided as cloud services during its ON period. We let C denote the set of capabilities, $C = \{c_1 c_2 \ldots c_i \ldots c_n\}$, where c_i represents a distinct capability in the set, and n represents the total number of distinct capabilities. Every device will have a set of capabilities that is a subset of C. At any particular time t, the devices whose ON period range includes t, are said to be active devices in the network. We visualize Local Overlay-based Mobile Clouds as shown in Fig 3.1 at an instance with six active devices. Table 3.1 shows the devices in figure 3.1 and their capabilities mapping as a matrix in which distinct capabilities are represented as columns and active devices as rows. The entries in the capability matrix can be read as follows: D_2 is active, and it has capabilities C_3, and C_4 but does not have C_1, C_2, or C_5. D_2 can use D_1 or D_3 or D_6 to execute tasks that require capability C_1 by offloading the task to one of its nearest capable devices.

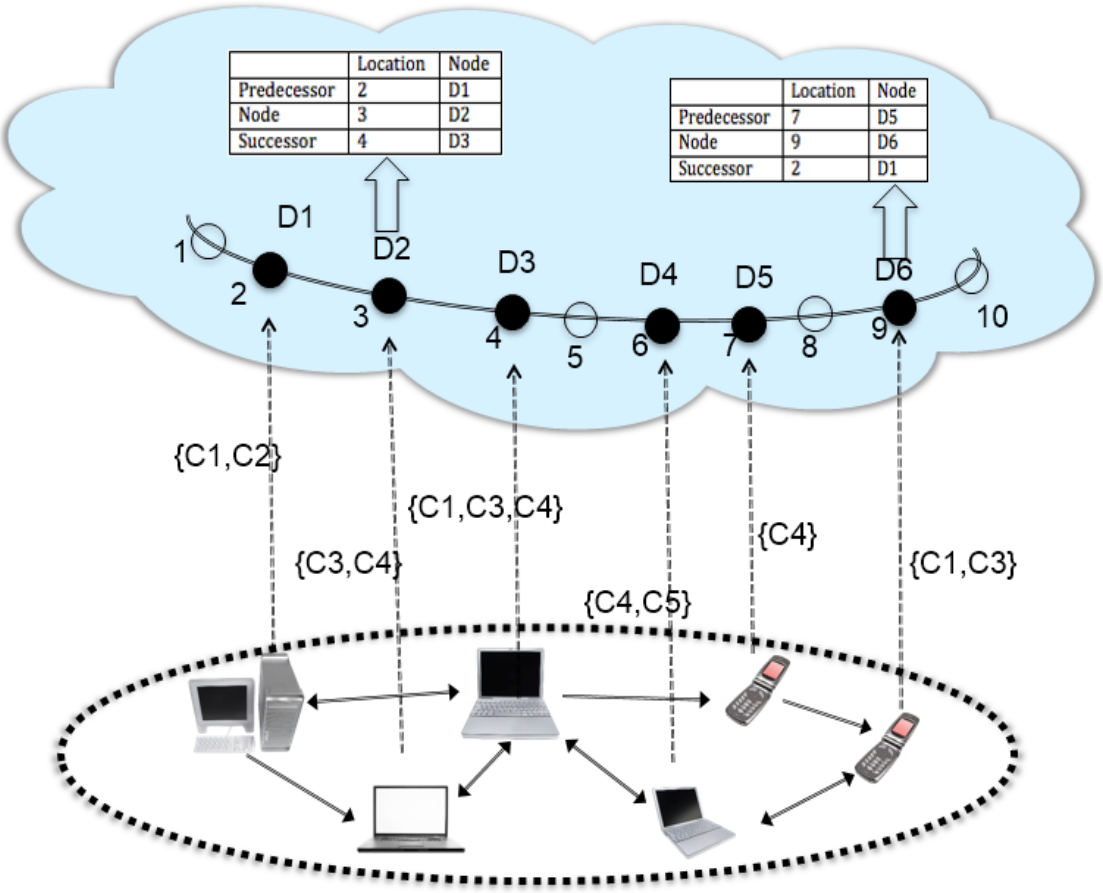


Fig 3.1 Local Overlay-based Mobile Clouds System model

Devices		Capabilities				
		C_1	C_2	C_3	C_4	C_5
	D_1	1	1	0	0	0
	D_2	0	0	1	1	0
	D_3	1	0	1	1	0
	D_4	0	0	0	1	1
	D_5	0	0	0	1	0
	D_6	1	0	1	0	0

Table 3.1 Device Capability Matrix

The device information is stored in a Distributed Hash Table (DHT), which is distributed among all the devices in the network. Each device maintains a part of the DHT as shown

in Fig 3.1 for “Device 2” and “Device 6”. Whenever a device joins, its node ID is assigned based on the locality of the device. The node ID structure is shown in fig. 3.2.

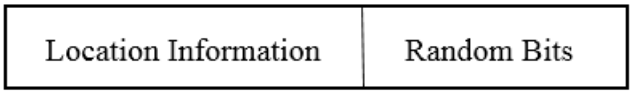


Fig 3.2 Node ID structure

By having the location information as a prefix, the physical topology is efficiently embedded into the hash space. Latitude and Longitude values are interleaved for the location information bits. In section 3.3, we show how node ID prefix is calculated based on location information is explained in detail. The random suffix bits aims to reduce the hashing collisions and spread out the devices evenly over the overlay.

So far we presented how we envision the Local Overlay-based mobile clouds. In the next subsection, we explore the device level architecture. Any device in the network will be able to decide whether to offload the computation or not, find one of the closest devices that is offering the required capability to execute the offloaded task, and provide its own capabilities as cloud services, etc.

3.2. Device Architecture

The device level architecture is divided into three layers. This layered approach will allow us to incorporate changes in any layer with ease without affecting other layers as shown in fig. 3.3.

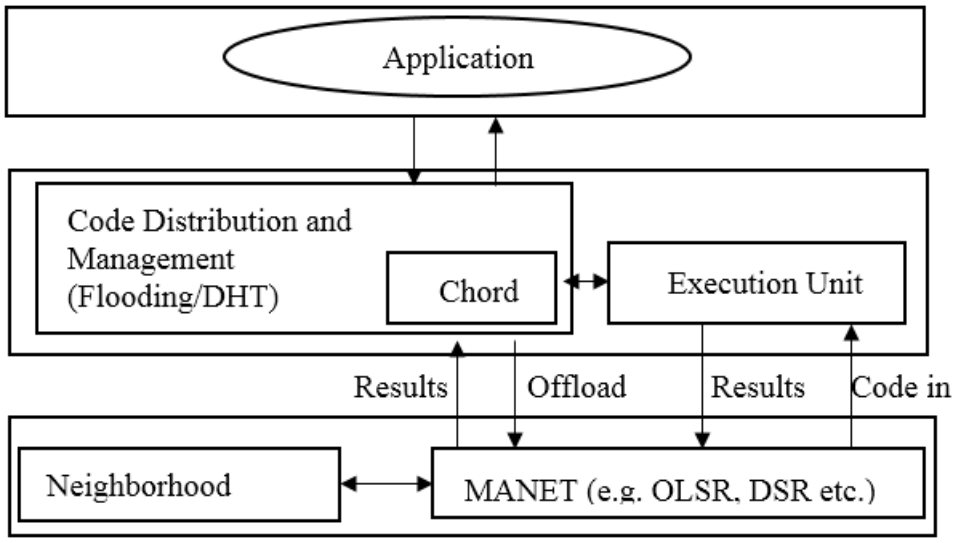


Fig 3.3 Layered device architecture

The top most layer (Layer 1) will be the application layer. Users interact with this layer to turn on the ad-hoc network and to communicate with other users in the network. The distribution module in layer 2 will take the decision to offload the job or to execute it locally on the device. The distribution module decides on how the job is distributed among the devices.

The bottom layer (Layer 3) consists of ad-hoc networking protocol implementation, which is an on-demand routing protocol and a module for discovering adjacent devices and their capabilities. This layer will maintain the routing table, which can be used to route the offloaded tasks, and get the results back. The devices and their capabilities matrices are mapped onto the Chord overlay in layer 2. The devices in the physical topology are mapped to the overlay in such a way that the devices that are neighbors in physical network to be closer on the overlay for routing locally. By routing locally, the context awareness of nearby devices can be utilized to execute the tasks, and the

bandwidth usage for communication with devices that are far can be minimized by identifying closer devices. To achieve that, we are exploiting the location information of the devices in node ID assignment by interleaving the latitude and longitude values.

3.3 Interleaved Hashing

Any point on the earth has a unique global positioning system (GPS) coordinates i.e. longitude and latitude. Latitude is the angular distance of a point from the north or the south of the equator. The latitude values range from -90° to +90°. The values from the equator (0°) to North Pole (+90°) are positive, and the values from the South Pole (-90°) to the equator are negative as shown in fig 3.4. Longitude is the angular distance of a point from east or west of prime meridian and the values range from -180° to +180°. The values from prime meridian towards east are positive (0° -180°) and towards west are negative (-180° to 0°) as shown in fig 3.5. The longitude values are mapped to (0° – 360°) i.e. the negative values are shifted 180° and the latitude values are mapped to (0° – 180°) i.e. the negative values are shifted 90°.

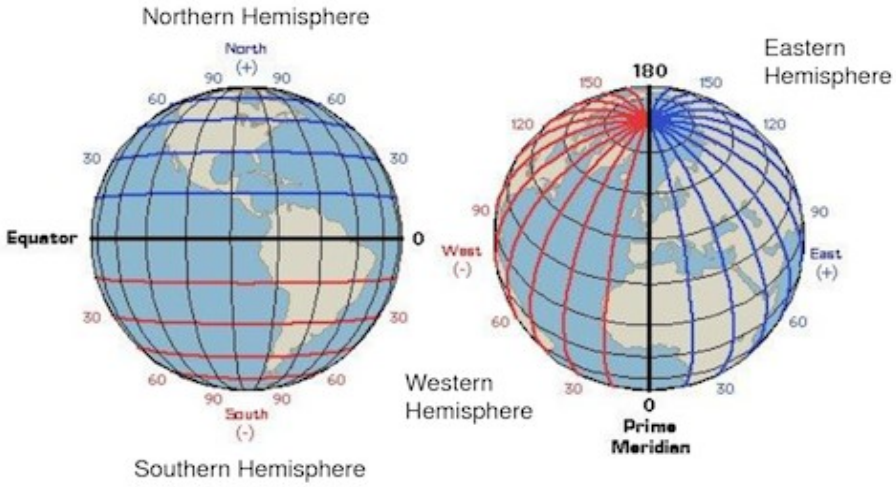
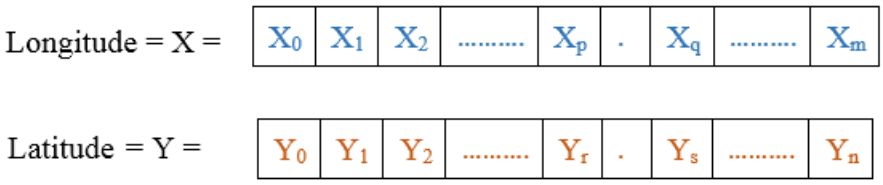


Fig 3.4 Latitude and Longitude [34]

We assume that each device can capture its GPS coordinates. In our hashing method, the latitude and longitude values of the device are converted to binary format and interleaved, so that the proximity relations in the physical layer are reflected in the overlay Chord. The floating point (.) is removed from the binary value as we need an integral value to map to Chord [7]. The floating point (.) is removed by appending the binary value of fractional part at the end of the integral part binary value and interleaved as shown in Fig 3.5. We let X denote longitude value; Y denote the latitude value and their binary representation as shown below.



We remove the floating point (.) and interleave them as shown below.

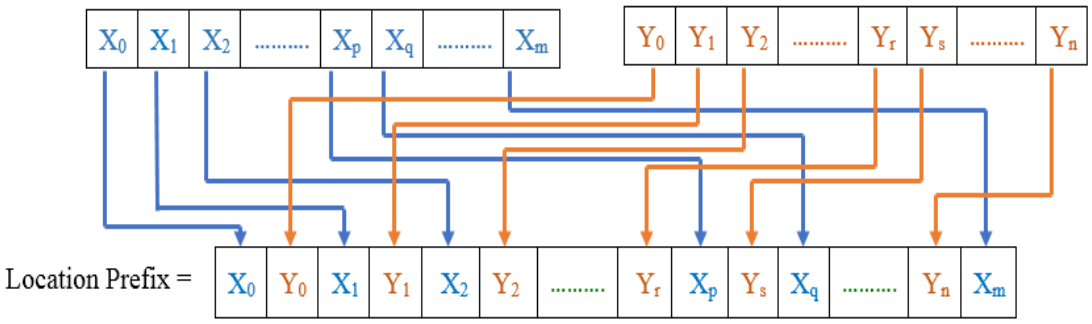


Fig 3.5 Location prefix for node ID

When the nodes are close to each other, their GPS values will be very close too (e.g. if the nodes are on different floors in a building at the same position when compared to earth). As a result, the hash values will be same, and there will be more collisions on the

overlay. To reduce the chance of collision, a suffix of random bits is added such that the devices at the same location also spread out in the hash space in the neighboring locations.

To identify the ratio of the number of bits allocated to the prefix and the suffix, we calculated the maximum number of bits required for the latitude and longitude values based on some precision. We consider up to 5 digits after the decimal point for the latitude and the longitude values to get around one meter precision which is good enough for tracking mobile devices. The maximum value for the latitude is 180 and for which we need 8 bits and the maximum longitude value is 360 and for which we need 9 bits. For the fractional part, we need a maximum of 53 bits when 5 places are considered. If fewer bits are required than allocated for binary representation, most significant bits of the integral parts will be 0's, and least significant bits are 0's in the fractional part. For interleaving, we need a total of 123 bits. For example, consider 23.5 as the latitude value, and 46.875 as the longitude value. The binary representation of the latitude value is 10111.1, and the longitude value is 101110.111. We represent an integral part of the latitude value as 00010111 (8 bits) and the fractional part as 10000… (1+52 0's – total 53 bits) and then we append them. Similarly, we represent the integral part of the longitude value as 000101110 (9 bits) and the fractional part as 11100… (111 + 50 0's – total 53 bits) and then we append them. For location information prefix, a total of 123 bits are required, and the remaining 37 bits are used for random bits.We consider the 37 most significant bits of the hash value obtained using SHA1 based on IP address for random

bits. We let R denote the random bits. Figure 3.6 shows how the 160-bit hash value is calculated.

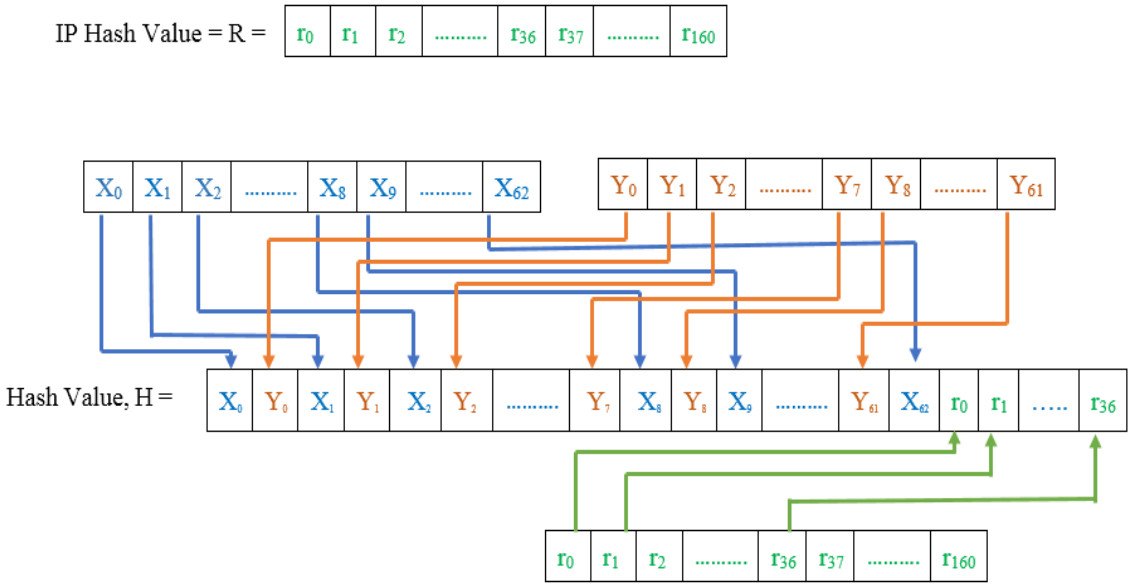


Fig 3.6 Node ID with location information as prefix and random bits as suffix

3.4 Routing Protocols

Every time a node raises a request for a capability, that node will decide whether to execute locally or offload to the cloud. Multiple factors can be considered in taking this decision such as resource availability, battery consumption, etc. We are considering whether the node has the capability or not. If the node doesn't have the capability, it will offload to the cloud. When a node wants to offload a task to its neighbors, it has to check for the devices that have the required capability to execute the task. The conventional way of searching is flooding and is described first, and then an overview of the modified Chord protocol is presented.

In this thesis, the flooding based protocol of interest is sequence number controlled flooding (SNCF) [18]. In SNCF, a source node sends a request packet to its peers. Further, these peers broadcast to their peers and so on until the packet reaches all the nodes in the network. The packets will be dropped by the node only when one of the following occurs:

- time to live of the packet expires
- sequence number in the packet is less than the latest sequence number that node heard from the source node
- node has already seen the packet
- node has only one edge (as the node will not forward to sender)

Nodes in the presented mobile cloud will also drop the packets when the node itself have the required capability. So, until the packet is dropped for any of the above scenarios, the packets will be rebroadcasted by the nodes which consume a considerable amount of bandwidth. In addition to bandwidth consumption, the CPU cycles and the battery power of the devices are wasted. If all the packets are dropped once the source node finds the capable node, it will reduce the usage of resources notably. Moreover, popular capabilities will be available with more peers and more likely to hear a reply from a capable node quickly by flooding but to find a rare capability more resources will be needed.

3.4.1 Overview of modified Chord

Chord [7] is one of the structured P2P networks that uses the DHTs. The devices in the physical topology are mapped onto the virtual Chord ring, and each node's ID is assigned

using SHA1. In the modified Chord, each node's ID is assigned using interleaved hashing for the overlay topology to be congruent with the physical topology. A key associated with a resource is obtained from the same hash space using SHA1 [5] and is mapped onto the node with the smallest ID greater than or equal to the key i.e. *successor node*. In our context, the resource is a capability, and that node is called a responsible node for that capability. For each distinct capability in the network, one node will be identified as the responsible node. A responsible node for a particular capability maintains the list of nodes in the network that can execute the capability, and a single node can be responsible for multiple capabilities as shown in Fig 3.7. Node D4 is responsible for capability C1 and C2. When capability C1 is hashed, the key obtained is 5. The node with highest ID less than or equal to 5 is D4. So, D4 is identified as the responsible node for C1 and the capable nodes list is maintained. Similarly for capability C2.

The capabilities of every joining node fall into two categories. Either the capability is already in the network or the joining node is bringing a new capability to the network. If the capability is already in the network, the joining node should let the responsible node of each of its capabilities know that it can execute those capabilities, and the responsible nodes will add the new node to their list. If the capability is distinct from the set of capabilities in the network, then a responsible node need to be identified by the new node. To identify whether a particular capability is already present in the network, the node will hash (H) the capability ID using SHA1 [5] and a lookup message is sent to the node with highest ID less than or equal to the hashed value (H) in its finger table. If the node hash ID is equal to the hash value (H), or the hash value (H) falls between that

nodes hash ID and its successor hash ID, then the packet reached its destination node. If not, the node will forward the packet to the highest ID less than or equal to the hashed value (H) in its finger table and so on until the packet reaches its destination node. If the destination node is the responsible node for that capability, then its list is updated with the new node and acknowledgment is the source node. If not, the node will forward the lookup packet to its successor node on the overlay and so on. If the initial destination node (i.e. the node with highest ID less than or equal to the hashed value (H)) received the lookup message again, that means the capability does not exist in the network till now, and that node will send the acknowledgment to the newly joined node that raised the lookup. The node having the highest ID less than or equal to the hash value obtained by hashing the capability using the SHA1 will be identified as responsible node by the new node that joined the network. If the node joining the network is the first node, no other nodes are present then the joining node itself will be identified as the responsible node. The newly identified responsible node's list is initialized with the joining node.

To efficiently lookup a key, every node with ID i will build and maintain a finger table to other nodes with up to n fingers from $(i + 2k) \bmod 2n$ where k is an integer ($0 \leq k < n$) as shown in fig 3.7. The node will have most of its fingers to the closer nodes in the forward direction (i.e. clockwise direction) as the difference between the entries in figure table grows as the power of 2. Node Id's and IP addresses are maintained in the finger table. Capability ID and the list of corresponding capable Nodes ID are maintained in the capability responsibility table.

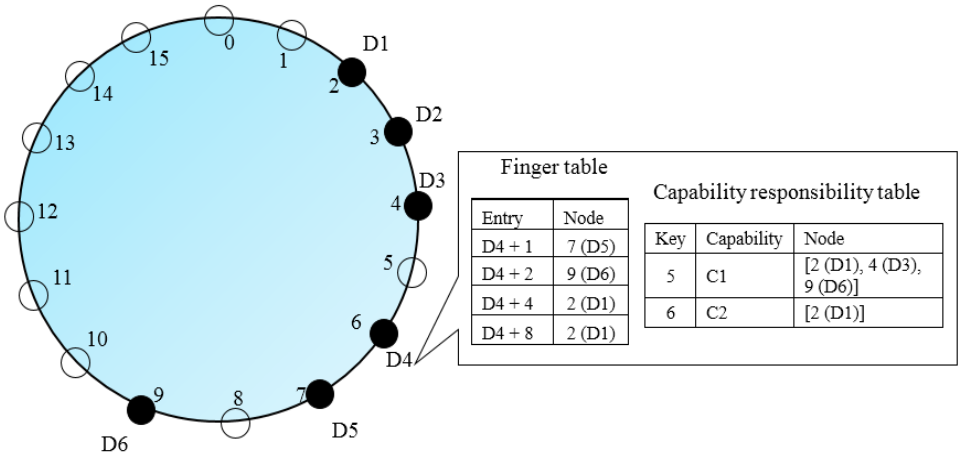


Fig 3.7 Chord ring n = 4 with 6 devices (from Fig 3.1) showing the finger table and the capability responsibility table for D4

Whenever a node is leaving, it will inform the nodes that are responsible for its capabilities it shared with the peers to delete it from their lists. If the leaving node is a responsible node, it will assign its responsibilities to the next node in the ring. The finger table of all the nodes in the network that contain the leaving node will be updated accordingly.

When a node raised a lookup request for a capability to offload a task, the hash value obtained using SHA1 is the key to lookup the responsible node. The lookup request will be handled similar to the responsible node lookup by a new node joining the network. The packet structure and the messages exchanged will be same as Chord [7].

3.4.2 Multiple responsible nodes

As the node density and network size grow (average path of the network is also increased), the number of hops the responsible node is away from a significant number of other nodes in the network can be high. To get the list of capable nodes, the packet sent

by the lookup node for a capability should reach the responsible node which can be many hops away. As a result, the average response time will increase, and the number of lookup messages also increases. Moreover, the Chord lookup is not symmetric as the number of fingers to nodes in the first half of the circle (i.e. to the nodes in the forward direction) is more than the number of fingers to the nodes in the second half (i.e. to the nodes in the backward direction). So, if the responsible node is close to the requested node in the forward direction, it can be reached in less number of intermediate nodes on the overlay than that is close in the backward direction. To decrease the average response time and to ensure high availability, the responsible nodes are replicated by dividing the hash space into regions and picking one responsible node from each region. The number of regions is expected to be a configuration parameter which would be based on the network size. The number of responsible nodes can be decided algorithmically (e.g. dividing the nodes in the network into clusters periodically and picking the responsible nodes from each cluster). We explore this in future.

3.4.3 Hybrid Routing Protocol

Flooding is better to find the highly replicated ones regarding response time, and DHT will be better to find the rare capabilities to minimize the number of messages generated. The number of lookup messages required is reduced, but the response time is increased when compared to flooding by using the DHT for searching popular capabilities. To take advantage of both flooding and DHT, we are presenting hybrid routing protocol.

There are some existing works [30-32] that proposed the use the hybrid routing protocol. In these protocols, the routing protocol selection is based on the statistics of the resources

(i.e. rare or popular) in the network collected by some nodes (ultrapeers or global index nodes). These protocols are not suitable for dynamic topologies as the overhead to maintain the statistics will be high. We are presenting a hybrid routing protocol, which does not require collecting statistics.

Our approach proceeds as follows. When a request for a capability is raised, we flood the query till the specific number of hops to hear the reply quickly if the capability is highly replicated and one of the capable node is within the specified number of hops. If we did not find the capable node by flooding, then the query will be issued in DHT. But with this approach, the lookup resolution time for rare capability is high. So, we present a hybrid parallel protocol (i.e. flood the query and issue the lookup in DHT in parallel). The query packet will be flooded only up to a certain number of hops so that the bandwidth usage will be reduced. If the source node finds the capable node by flooding before hearing back from the responsible node, it will not process the list of nodes obtained from the responsible node which will reduce the computation and the messages involved. By issuing a query in parallel, there will be message overhead from both the protocols but the average lookup time will be better as we are taking advantage of flooding to search for the popular capabilities and DHT for finding the rare capability.

IV. EVALUATION

To evaluate the performance of our presented interleaved hashing technique, and hybrid routing protocol, we implemented a simulator and conducted simulation experiments. We studied the impact of different factors such as the number of lookup messages generated, and the number of lookup hops taken until a response is received under different arrival rates and varying capability distributions.

The descriptions of the performance metrics used are as follows:

Look up hops count is the average number of hops a lookup request traverses in the physical topology. We consider lookup hops count as the response time to resolve a request. For resolved cases, the number of hops the packet traverses till the results are obtained by the source node from the capable node are counted. For the unresolved cases, the hops the lookup packet traverses till the source node hear the acknowledgment that the capability is not available in the network are counted. If the packet reaches the initial destination node for the second time, then that node will acknowledge the source node about the unavailability of a capable node in the network.

Lookup messages are the average number of messages generated in the network for a lookup request.

In the simulations, every device is given an arrival time, an ON time, GPS coordinates, i.e. latitude and longitude values, a capability set and an IP address. The GPS coordinates are picked randomly from a predefined range of latitude and longitude values using a uniform random function. We varied the arrival rate and the capability distribution as we generated the devices. We assume nodes will be joining the network according to a

Poisson distribution. We varied the average arrival rate (λ) to be 1 device/10 min, 1 device/7 min, 1 device/5 min, 1 device/2 min and 1 device/min. Each device is assigned 0, 1, 2, 3, or 4 capabilities and this number is picked uniformly at random. We want to evaluate how the cloud functions when the capabilities are uniformly distributed and when some capabilities are highly replicated with many devices, and few of them are rare. So we conducted experiments by assigning the capabilities to devices following uniform distribution and Zipf distribution from a predefined capabilities set. We used '1' as the exponent in Zipf distribution so that every capability occurs at least once from the predefined capability set. The ON period of the devices follows a Pareto distribution with an exponent of 0.83 which is known as heavy tailed distribution because few devices stay for long periods of time while the majority leave over short intervals. This distribution is usually observed in P2P networks. We held the average ON period any device would stay in the network to be 5 minutes. Table 4.1 gives the details of the simulation parameters, and the following assumptions were considered throughout the simulation experiments.

Assumptions:

We assume that the Wi-Fi chipset range is considered to be 30 meters [6] i.e. devices that are within 30 meters range of any device are considered to be neighbors of that device. We also assume that every device in the network has the complete topology view as obtained by the optimized link state routing (OLSR) [21]. While calculating the average lookup hops, we are not considering the delay on links, we use the count as a proxy for the delay.

Simulation Area	100 m x 100 m
Wi-Fi Range	30 m
Chord space	2^160
Finger table size	160
Under laying Protocol	OLSR
Number of nodes generated	250
Average Service time	5 min

Table 4.1 Simulation Parameters

Fig. 4.1 shows the network size over time for different arrival rates following Poisson distribution and their ON period following the Pareto distribution. Devices that are in range establish edges with each other. The devices that are not in range to any other device are considered as a single device network. When a new node arrives, and it is in range with the disconnected node, then they establish an edge among themselves. When a node leaves the network and make some other node to fall apart from the network, then the fallen node will be removed from the network, and it is considered as a single node network. Similarly, when a set of nodes falls apart from another set of nodes, each set of nodes will be considered as a separate network. If a new node joins and is in range with the nodes in different broken networks, then all the nodes will form a single network.

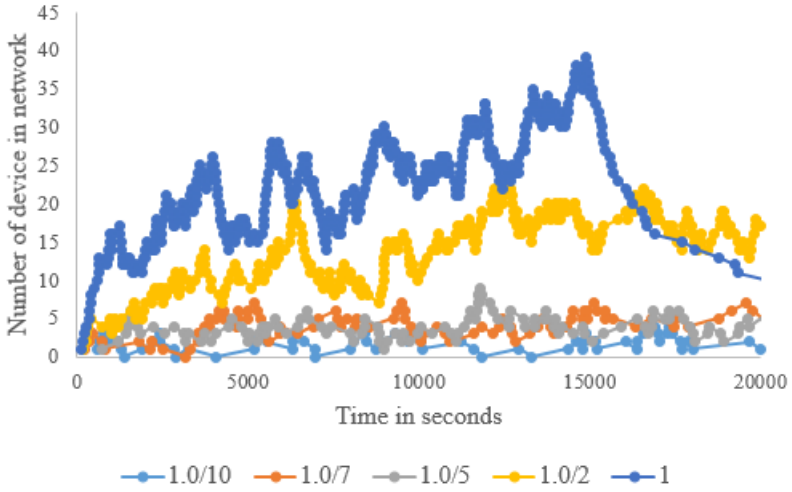


Fig 4.1 Network Size over time for different arrival rates

Table 4.2 show the network topology view when nodes are arriving at different arrival rates. Disconnected nodes are nodes that do not have any neighbors, and the partially disconnected nodes are those that have neighbors but cannot reach a few other nodes in the network as the network is broken. The average longest path is the average of the longest paths at different instances during the simulation. The average path length (i.e. average number of hops between any two nodes in the network) is almost constant as the arrival rate increases and a number of nodes in the network at any instance increases. The average percentage of capabilities is the average of the ratio of distinct capabilities available in the network to the total capabilities in the predefined capability set at different instances. As the arrival rate increases, the average percentage of capabilities available also increases with the increase in the average number of neighbors. By looking at the simulation network statistics, we can say, the network resembles mesh topology.

Each node in the network will raise 0, 1 or 2 lookup requests (i.e. tasks) in their lifetime and this number is picked uniformly at random. The requests that are raised between the first node departure and the last node arrival are considered in all the experiments for the purpose of evaluation, which we consider to be the stable state of the network, and all the experiments are repeated five independent times and the results are averaged.

Arrival Rate	Network Size			Neighbors			Avg % of Capabilities in the network	Avg No.of disconnected nodes	Avg No.of partial discon-nected nodes	Average longest Path	Average Path
	Min	Max	Average	Min	Max	Average					
0.1	1.2	9.8	4.968472	0	4.4	0.96402	34.74794528	1.603320042	2.342437	1.330869	0.980283
0.14286	1.2	12.2	5.912896	0	4.8	0.987104	42.76419226	1.803584025	2.984483	1.59886	1.05855
0.2	2.2	16.4	9.316055	0	6.6	1.707516	51.81047292	1.634641365	5.642477	2.668379	1.437812
0.5	5	24.6	16.79881	0	10.8	3.216175	73.46313393	0.678356463	9.323829	5.3262	2.301127
1	11.2	39.2	27.0325	0	14.8	5.500396	90.62070169	0.194797509	8.762806	7.247736	2.933761

(a) Uniform capability distribution

Arrival Rate	Network Size			Neighbors			Avg % of Capabilities in the network	Avg No.of disconnected nodes	Avg No.of partial discon-nected nodes	Average longest Path	Average Path
	Min	Max	Average	Min	Max	Average					
0.1	1.2	11.8	5.430368	0	5.4	0.9844764	32.30780745	1.737586966	2.6194802	1.5137009	1.0184157
0.14286	1.8	14	7.912524	0	5.8	1.3403984	40.87157994	1.782651774	4.907964	2.17701592	1.3057645
0.2	2.2	13.8	7.939238	0	6.6	1.5039178	41.4651968	1.710918573	4.5376113	2.37247881	1.3738544
0.5	7.4	26	17.12629	0	11	3.2742173	64.23339271	0.754462527	8.9029605	5.86320266	2.4865574
1	7.8	34.8	24.65525	0	14.4	4.8758188	73.38106029	0.39851363	10.514099	7.27052273	2.9238824

(b) Zipf capability distribution

Table 4.2 Network topology statistics (Min – Minimum, Max- Maximum)

Each lookup request is raised by a node in the network for one capability and is picked at random from the predefined capability set so that we can simulate requests that won't get resolved (when the capability is not present in that nodes network) and requests that get resolved. The resolved requests include the cases where the node itself is capable of

executing the task along with the cases that get resolved by finding a capable node in the network. Fig 4.2 gives the percentage comparison of resolved and unresolved requests among the total requests raised. The percentage of unresolved requests decreases as the arrival rate increases because the capability coverage increases as the number of nodes in the network increases. The percentage of unresolved requests is higher when device capabilities follow Zipf distribution when compared to a uniform distribution as the average percentage of capabilities available is less as some capabilities are highly replicated (refer Table 4.2).

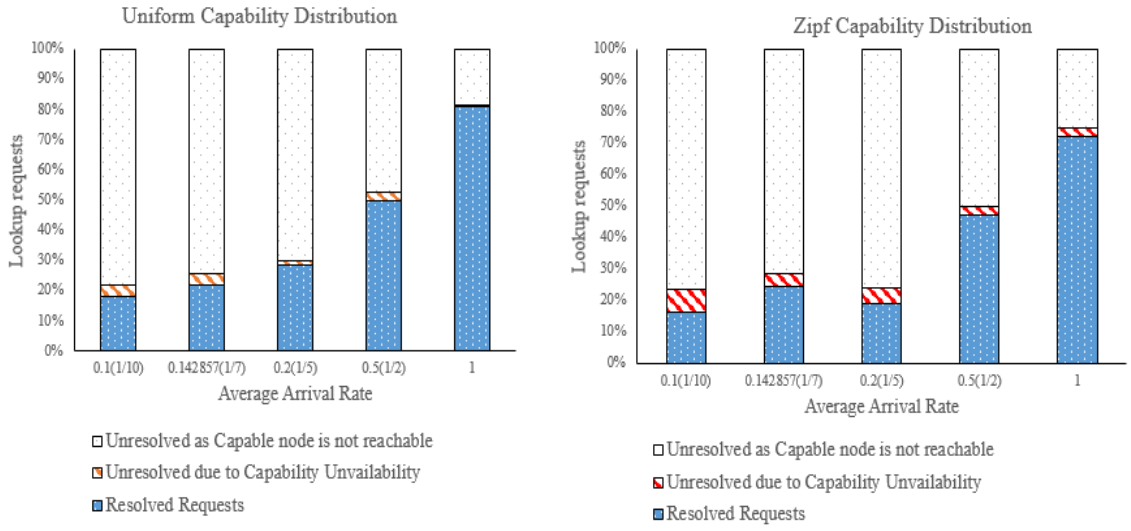


Fig 4.2 Lookup requests – Resolved, and Unresolved

The descriptions of the terminologies used are as follows:

DHTS - Chord DHT with single responsible node for each capability

DHTM (n) - Chord DHT with n responsible nodes for each capability

FDHTS – Hybrid algorithm when flooding and DHTS are run in serial

FDHTS Parallel – Hybrid algorithm when flooding and DHTS are run in parallel

FDHTM – Hybrid algorithm when flooding and DHTM are run in serial

FDHTM Parallel – Hybrid algorithm when flooding and DHTM are run in parallel

Flooding – Total number of messages that will be flooded in the network

FloodingOrg – The number of messages that are flooded till the source hears the first reply

We first examine the lookup efficiency when the interleaved hashing is used to map the devices to the overlay then the effect of Chord with multiple responsible nodes followed by the evaluation of hybrid routing protocol.

4.1 Efficiency of Interleaved Hashing

The devices in the physical topology are mapped to the overlay (Chord DHT) using SHA1 and interleaved hashing. Figure 4.3 shows the comparison of the average number of lookup hops for both hashing techniques. With low arrival rates, the average lookup hops are relatively lower when interleaved hashing is used compared to SHA1 as the average network size will be small. When the arrival rate is just over twice the departure rate, we can see a significant decrease in the number of lookup hops when interleaved hashing is used. When devices are mapped to the overlay using interleaved hashing, there is 40% decrease in the number of lookup hops with uniform capability distribution and 33% decrease when the capability distribution follows a Zipf distribution and their difference increases with the increase in the arrival rate. This is because, when SHA1 is used for mapping devices to the overlay, there is no congruity between the overlay and the physical topologies. Neighbors on the overlay can be many hops away on the physical topology. With interleaved hashing, devices that are neighbors in physical topology are

more likely to be neighbors in the overlay. When a node raising the request receives the capable nodes list from the responsible node, the source will send the request to the nearest node from that list. When devices capabilities follow Zipf distribution, the average lookup hops difference is less compared to when devices capabilities follow Uniform distribution. This is because the probability that the capability is present in the network is less as few capabilities are rare when the capabilities are assigned to devices following the Zipf distribution (Refer Table 4.2).

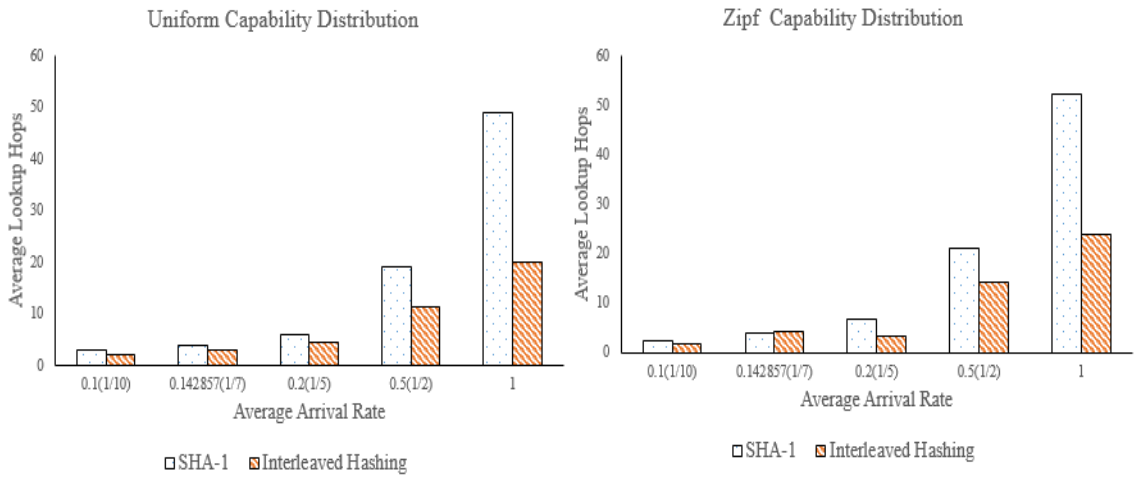


Fig 4.3 Average lookup hops comparison (SHA1 vs. Interleave hashing)

4.2 Chord with multiple responsible nodes

4.2.1 Lookup hops efficiency

We increased the number of nodes responsible for a resource so that any node in the network can find a responsible node nearby and repeated the simulations to see the effect on the average lookup hops.

When the capabilities are assigned to devices in the network following the uniform distribution, and the average arrival rate is lower than the average departure rate, the

number of responsible nodes doesn't make a difference in the average number of lookup hops. As the average arrival rate increases, having more than one responsible node will decrease the number of lookup hops. The increase in the number of responsible nodes beyond two doesn't decrease the average response time further because irrespective of the number of responsible nodes, there will not be a significant difference in the number of hops a responsible node is away from the source node. When the capabilities are assigned to devices following the Zipf distribution, failure cases are relatively higher. Those unresolved cases incur the same cost irrespective of the number of responsible nodes as the capability itself is not in the network in which case we will not have any responsible node for the capability.

To summarize, the average number of lookup hops may decrease by having more than one responsible node when the network is spread out and large enough to uniformly occupy the chord space as we divide the entire hash space into regions.

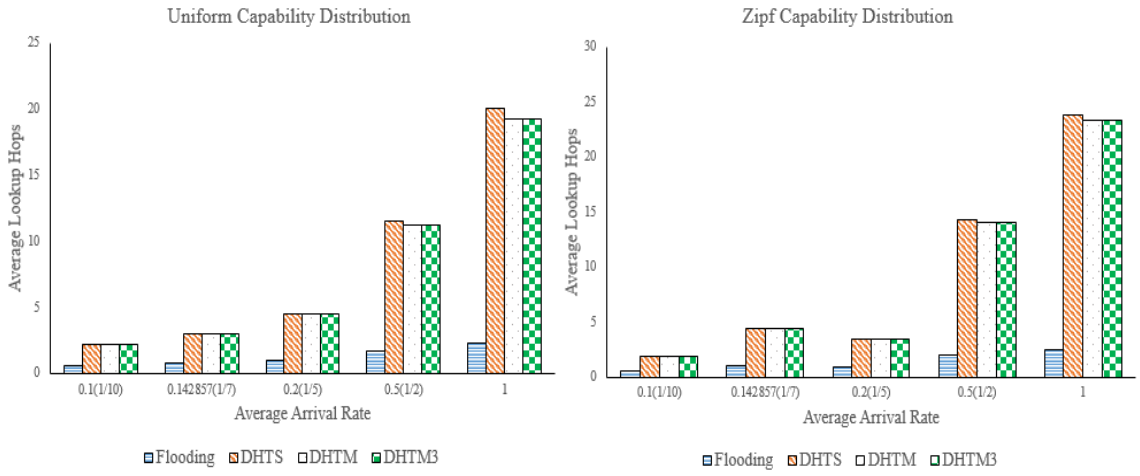


Fig. 4.4 Average lookup hops comparison

Fig. 4.4 shows the average number of lookup hops when the request is flooded are significantly lower than issuing a query in DHT. To decrease the average number of lookup hops, we experimented with the hybrid protocol.

4.3 Hybrid Routing Protocol

4.3.1 Lookup hops efficiency

We conducted experiments with hybrid serial (flooding till the specified number of lookup hops and if the capable node is not found, query is issued in DHT with two responsible nodes) routing protocol and hybrid parallel routing protocol (flooding till the specified number of lookup hops and issuing query in DHT at the same time with two responsible nodes).Fig. 4.5 shows the average lookup hops of hybrid protocols (flooded till two hops) is less than issuing a query in DHT. When the arrival rate is equal or less than the departure rate, the average lookup hops of the hybrid serial is equal to the case of issuing a query in DHT. When the arrival rate is just above the twice of departure rate, the average number of lookup hops by running the FDHTMS is 30% less than the DHTM when capabilities are uniformly distributed and 20% less when devices capabilities follow Zipf distribution.

The average lookup hops with the hybrid parallel protocol is less than a DHT for all the arrival rates. When the arrival rate is just over the twice of departure rate, we can see a significant decrease in lookup hops irrespective of capability distribution. When the average arrival rate of the devices is just over twice the departure rate, and device capabilities are uniformly distributed, we can see a decrease of 36% with FDHTM Parallel compared to DHTM and when capabilities follow Zipf distribution, 28%

decrease. As the arrival rate increases, the difference also grows. When the arrival rate is 1, a 50% decrease is observed when the device capabilities follow uniform distribution and 37% decrease when the devices capabilities follow Zipf distribution.

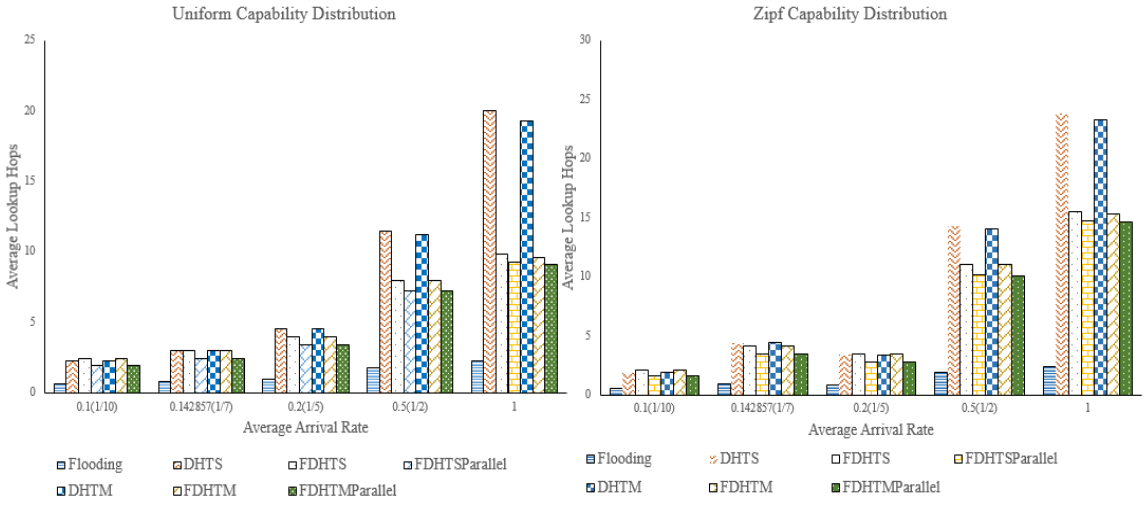


Fig 4.5. Average Hops comparison

4.3.2 Lookup messages efficiency

With hybrid parallel routing protocol, the response time is decreased but there will be an increase in the number of messages generated per request. Fig. 4.6 shows the total number of average lookup messages generated to resolve a request. We can see that the hybrid parallel protocol (FDHTM Parallel) average lookup messages are higher than DHTM (which is less than floodingOrg) and lower than the flooding from fig 4.6. When the arrival rate is just over the twice of departure rate, and devices capabilities follow uniform distribution, there is a 13% decrease in average lookup messages compared to flooding and 19% decrease when the device capabilities follow Zipf distribution. The difference increases as the arrival rate increases. When the arrival rate is 1, there is a 50%

decrease when compared to flooding when devices follow uniform capability distribution and 52% decrease when the device capabilities follow Zipf distribution.

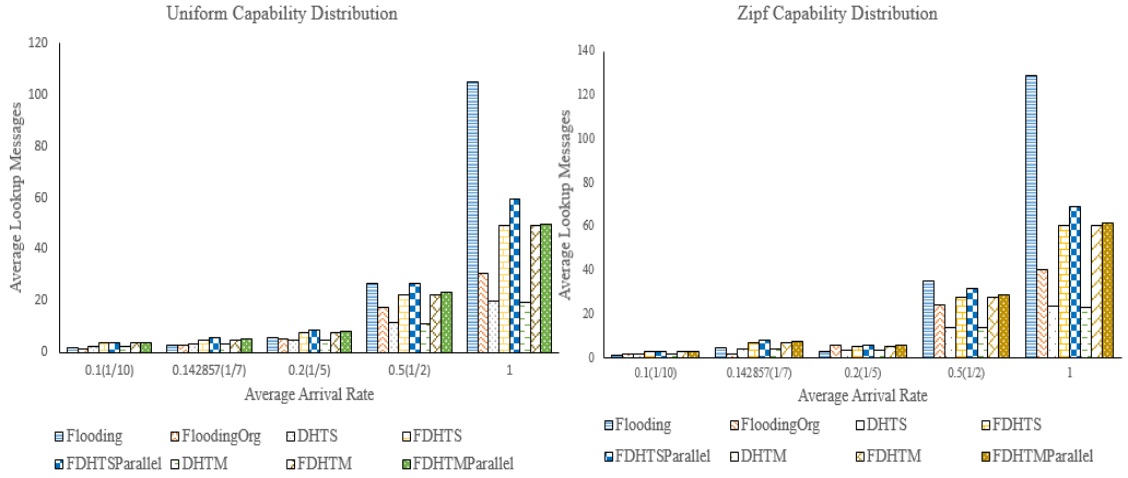


Fig 4.6 Average Lookup Messages comparison

To conclude, by overlay DHT with interleaved hashing, the average number of messages are less than floodingOrg but the overhead in the response time (average number of lookup hops) is increased when the arrival rates are higher than the departures rate. With hybrid protocol, we reduced this overhead. Hybrid protocols serial or parallel consume almost the same number of lookup messages when arrival rate is less than departure rate, and the average number of lookup hops is less with the parallel protocol. When the arrival rate is just over the twice of the departure rate, and the device capabilities follow a uniform distribution, the parallel hybrid protocols take 36% less number of lookup hops when compared to DHTM and the average lookup hops is reduced by 13% when compared to flooding. When the capability distribution follows Zipf distribution, the parallel hybrid protocols take 28% less number of lookup hops when compared to DHTM and the average lookup hops is reduced by 19% when compared to flooding. When the

arrival rate is 1, there is 50% decrease in the average lookup messages when compared to flooding. So, for all the arrival rates, FDHTM parallel outperform other routing protocols.

V. CONCLUSION AND FUTURE WORK

In this thesis, we have presented a mobile cloud that capitalizes on locality through interleaved hashing and provides an efficient collaborative computing framework enabling a group of mobile devices to form an ad-hoc network based on their proximity and advertise their capabilities as cloud services. We also presented a hybrid parallel routing protocol to route requests, which consumes less bandwidth compared to flooding and better response time compared to DHT. We have evaluated the performance of our presented mobile cloud through simulation. The simulations show the better response time with minimum bandwidth utilization using interleaved hashing technique and hybrid routing protocol for task execution even with a small magnitude of the network.

In future, we plan to expand on our results through larger networks to see the full potential of the presented mobile cloud and one of the reactive protocol as the underlying protocol. The number of responsible nodes can be decided on demand (e.g. dividing the nodes in the network as clusters periodically and picking a responsible node from each cluster) instead of dividing the hash space and picking a node at random to be responsible in each region based on the configuration parameter. In addition to picking the responsible nodes based on the node density, we will explore adding a new responsible node based on the load on current responsible nodes to reduce it. Future work will also focus on implementing an Android prototype to evaluate the proposed model.

REFERENCES

[1] Open source project – ProjectSPAN / android-manet-manger https://github.com/ProjectSPAN/android-manet-manager.

[2] Google Android Development, “Android 4.0 APIs,” http://developer.android.com/ about/versions/android-4.0.html, 2013.

[3] Gnutella. http://gnutella.wego.com.

[4] Kazaa. http://www.kazza.com.

[5] SHA-1: http://en.wikipedia.org/wiki/SHA.

[6] Schollmeier, R. "A definition of peer-to-peer networking for the classification of peer-to-peer architectures and applications." In Proceedings of the First International Conference on Peer-to-Peer Computing, IEEE 2002.

[7] Stoica, I., Morris, R., Karger, D., Kaashoek, M. F., and Balakrishnan, H., "Chord: A scalable peer-to-peer lookup service for internet applications." ACM SIGCOMM 2001, San Diego, CA, September 2001.

[8] Ratnasamy, S., Francis, P., Handley, M., Karp, R., and Shenker, S., “A scalable content-addressable network,” (Vol. 31, No. 4, pp. 161-172). ACM 2001.

[9] Rowstron, A., and Druschel, P., “Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems,” In Middleware 2001, pp. 329-350. Springer Berlin Heidelberg, 2001.

[10] Zhao, B. Y., Kubiatowicz, J., and Joseph, A. D., "Tapestry: An infrastructure for fault-tolerant wide-area location and routing." (2001): 70.

[11] JOSH, T., JEFF, R., and NICK, M., "Off grid communications with android meshing the Mobile world." Accuvant LABS Research Consulting Group, Bedford, MA, United States (2012): 401-405.

[12] Gordon, M. S., Jamshidi, D. A., Mahlke, S. A., Mao, Z. M., and Chen, X., "COMET: Code Offload by Migrating Execution Transparently," In Proceedings of the 10th USENIX conference on Operating Systems Design and Implementation (OSDI), Hollywood, CA, October 2012.

[13] Cuervo, E., Balasubramanian, A., Cho, D. K., Wolman, A., Saroiu, S., Chandra, R., and Bahl, P., "MAUI: Making Smartphones Last Longer With Code Offload," In Proceedings of the 8th international conference on Mobile systems, applications, and services, San Francisco, California, USA, June 2010.

[14] Zhang, Y., Huang, G., Liu, X., Zhang, W., Mei, H., and Yang, S., "Refactoring Android Java Code For On-Demand Computation Offloading," ACM Special Interest Group on Programming Languages (SIGPLAN) Notices, vol. 47, no. 10, pp. 233-248, 2012.

[15] Guirguis, M., Ogden, R., Song, Z., Thapa, S., and Gu, Q., "Can You Help Me Run These Code Segments on Your Mobile Device," Global Telecommunications Conference (GLOBECOM 2011), 2011 IEEE, Date 5-9 Dec. 2011.

[16] Langford, T., Gu, Q., Rivera-Longoria, A., and Guirguis, M, "Collaborative Computing On-Demand: Harnessing Mobile Devices in Executing On-the-Fly Jobs," Mobile Ad-Hoc and Sensor Systems (MASS), 2013 IEEE 10th International Conference on Date 14-16 Oct. 2013.

[17] Penner, T., Johnson, A., Van Slyke, B., Guirguis, M., and Gu, Q., "Transient Clouds: Assignment and Collaborative Execution of Tasks on Mobile Devices ". In proceedings of IEEE Globecom, Austin, TX, December 2014.

[18] Tanenbaum, A. S., and Wetherall, D. J., “Computer Networks, 5th Edition,” pages 368-370.

[19] Clausen, T., and Jacquet, P., “Optimized link state routing protocol (OLSR),” IETF, Request for Comments (RFC) 3626, 2003.

[20] Perkins, C., Belding-Royer, E., and Das, S., “Ad-hoc on-demand distance vector (AODV) routing,” No. RFC 3561. 2003.

[21] Johnson, D. B., and Maltz, D. A., "Dynamic source routing in ad-hoc wireless networks," Mobile computing, Springer US, pp. 153-181, 1996.

[22] Mbarushimana, C., and Shahrabi, A., “Comparative Study of Reactive and Proactive Routing Protocols Performance in Mobile Ad Hoc Networks,” Advanced Information Networking and Applications Workshops, AINAW '07. 21st International Conference 2, 2007.

[23] Aujla, G. S., and Kang, S. S., "Comprehensive evaluation of AODV, DSR, GRP, OLSR and TORA routing protocols with varying number of nodes and traffic applications over MANETs." IOSR Journal of Computer Engineering 9.2 pp. 54-61, 2013.

[24] Ali, S., and Ali, A., “Performance Analysis of AODV, DSR and OLSR in MANET,” In Proceedings on Seventh International Conference on Wireless Systems, p. 34. 2009.

[25] Burresi, S., Canali, C., Renda, M. E., and Santi, P., "Meshchord: a location-aware, cross-layer specialization of chord for wireless mesh networks (concise contribution)," In Pervasive Computing and Communications, 2008. PerCom 2008. Sixth Annual IEEE International Conference on, pp. 206-212. IEEE, 2008.

[26] Le-Dang, Q., McManis, J., and Muntean, G. M., "Location-aware chord-based overlay for wireless mesh networks." Vehicular Technology, IEEE Transactions on 63, no. 3 (2014): 1378-1387.

[27] Wu, W., Chen, Y., Zhang, X., Shi, X., Cong, L., Deng, B., and Li, X.,"LDHT: locality-aware distributed hash tables." In Information Networking, 2008. ICOIN 2008. International Conference on, pp. 1-5. IEEE, 2008.

[28] Zhou, S., Ganger, G. R., and Steenkiste, P. A., “Location-based Node IDs: Enabling Explicit Locality in DHTs,” September 2003 CMU-CS-03-171.

[29] Pethalakshmi, A., and Jeyabharathi, C., "Geo-Chord: Geographical Location based Chord Protocol in Grid Computing," International Journal of Computer Applications 94, no. 3 (2014).

[30] Loo, B. T., Huebsch, R., Stoica, I., and Hellerstein, J. M., “The Case for a HybridP2P Search Infrastructure,” IPTPS'04 Proceedings of the Third International Conference on Peer-to-Peer Systems on Date 26 Feb. 2004, pp. 141-150.

[31] Tsoumakos, D., and Roussopoulos, N., "Adaptive peer-to-peer search," University of Waterloo Technical Report 55 (2004).

[32] Zaharia, M., and Keshav, S., “Gossip-based search selection in hybrid peer-to-peer networks,” Concurrency and Computation: Practice & Experience - Recent Advances in

Peer-to-Peer Systems and Security (P2P 2006), Volume 20 Issue 2, pp. 139-153 February 2008.

[33] Larsson, T., and Hedman, N., “Routing Protocols in Wireless Ad-hoc Networks - A Simulation Study,” Master Thesis, LuleÅ Tekniska Universitet, 1988.

[34] Dick Eastman, “http://blog.eogn.com/2014/09/16/convert-an-address-to-latitude-and-longitude/”.