Toward Local Overlay-based Mobile Clouds

Lavanya Tammineni
Department of Computer Science
Texas State University
lavanya.tammineni@txstate.edu

Mina Guirguis
Department of Computer Science
Texas State University
msg@txstate.edu

Abstract— **Mobile devices are evolving into powerful systems due to recent advances in their communication, storage and computation technologies. They are posed to play a key role in providing a rich collaborative 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 making them useful in scenarios in which access to infrastructure may not be feasible or expensive. In this paper we present a collaborative computing mobile cloud that enables a group of mobile devices to form an ad-hoc network and advertise their capabilities as cloud services. The mobile cloud capitalizes on locality through the use of a GPS-based hashing technique and a hybrid routing algorithm that combines reliable flooding with Distributed Hash Tables (DHT) to find the closest node(s) that have the desired capabilities for task execution. We evaluate the performance of our proposed mobile cloud through extensive simulations and our results show the feasibility of our proposed platform.**

I. Introduction

Motivation: Recently, there has been a growing interest in Mobile Clouds – clouds that are formed solely by the mobile devices present in an environment without relying on existing infrastructure [1]–[5]. Mobile clouds are envisioned in scenarios where access to the infrastructure may not be possible and/or efficient (e.g., natural disasters areas, heavily crowded regions with limited connectivity, remote areas in military operations, etc). The increase in the numbers, capabilities and diversity of mobile devices are enabling factors for establishing a collaborative computing platform in which devices can share their resources (e.g., capabilities) as cloud services among themselves.

Mobile clouds combine various research areas together such as peer-to-peer networking, mobile ad-hoc routing, and cloud computing to enable them to be realized in practice. Recent efforts have focused on partitioning and distributing the workload among the devices to attain certain properties such as load balancing, collocating tasks for execution on a device and/or sharing a context among nearby devices [2]–[4], [6]. Due to the ad-hoc nature of the underlying connectivity of devices in mobile clouds, nearby devices would be typically interested in a similar computation/context/applicaiton. Consider, for example, a mobile cloud formed by vehicles stuck in a highly congested road segment with limited accessibility to the cellular infrastructure. The majority of them would be interested in finding an alternate route and thus they can share the same "context" among themselves without having each device access the infrastructure individually. The locality in such settings plays an important role that should be exploited by mobile clouds.

In this paper we present a collaborative computing mobile cloud that enables a group of mobile devices to form an ad-hoc network and advertise their capabilities as cloud services. The advertisement of capabilities is done through a structured peer-to-peer network (e.g., Chord [7]) that runs over the ad-hoc physical network. To exploit locality, we use Distributed Hash Tables (DHTs) in which a device's ID in the hash space is obtained using a prefix of interleaved code of its Global Positioning System (GPS) coordinates and a suffix of random bits to reduce collisions. A hybrid routing algorithm is devised that combines reliable flooding (over the ad-hoc network) with routing over DHTs to find the closest node(s) with the desired capabilities. Through finding and requesting capabilities from nearby devices (if present), the number of lookup messages and the number of hops traversed are reduced. Furthermore, to ensure high availability, we study the effect of having multiple responsible nodes to answer queries about the capabilities present. This has the effect of virtually dividing the hash space into smaller regions that can be searched more efficiently. In our assessment, we study the performance of different versions of our hybrid routing algorithm on different metrics such as the average number of lookup messages and the average number of hops traversed until a response is received under different networks loads. Our results show the feasibility of our proposed mobile cloud computing platform.

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

- 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.
- Design a new hashing mechanism that utilizes the latitude and longitude coordinates of a mobile device for node ID assignment to induce congruity between the physical and the overlay topologies.
- Develop a hybrid routing algorithm that balances between reliable flooding and Distributed Hash Tables (DHT) to find device(s) with desired capabilities, giving preferences to nearby ones.
- Evaluate the proposed framework through extensive simulations under various traffic loads and conditions.

Paper organization: The paper is organized as follows: Section II describes the related work. In Section III, we present our system model, GPS-based hashing algorithm and the hybrid routing protocol. We present our evaluation in

978-1-5090-4476-4/16/$31.00 ©2016 IEEE

Section IV followed by our conclusions in Section V.

II. Related Work

In this section we put our work in context with the following three areas of research:

Mobile Cloud Computing (MCC): A significant amount of work focused on mobile cloud computing as a mean to offload computation to external servers in order to reduce the workload on the mobile devices [8]–[10]. Mobile clouds that only rely on devices present without access to the infrastructure have also been proposed [2]–[4]. In [3], a collaborative computing framework is proposed in which jobs are mapped for execution on nearby devices by considering factors such as the requirements of the job, capabilities of the devices, dependencies of the tasks, and adjacency of the devices. In [2], a probabilistic code distribution model that enables a mobile device to execute code segments on nearby mobile devices has been proposed. In Transient Clouds [4], 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 due to the limitation of Android Wi-Fi direct. This project expands on the above by considering a completely ad-hoc network, taking into account the capabilities of the devices and their physical proximity to each other.

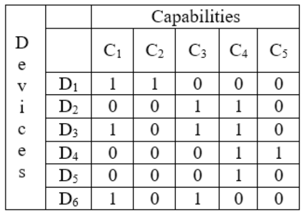
Mobile Ad-hoc Networks and Routing: Routing over a mobile ad-hoc network is an area with a very large body of research (e.g., DSR [11] and AODV [12] with all their variants). Recently, there have been some work on specific implementation of MANETs over Android devices. In Android 4.0 APIs [13], Wi-Fi Direct enables devices to connect among themselves without a wireless access point using a cluster head. SPAN [14] 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 Optimized Link State Routing Protocol (OLSR) [15] is used as the routing protocol. In OLSR, a Multi-Point Relay (MPR) is selected to flood 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. In this project, we also use OLSR as a component in our hybrid routing protocol.

Location-aware Peer-to-Peer Networks: Many of the proposed peer-to-peer networks (e.g., Chord [7]) can be adapted to be location-aware. This is achieved by the careful selection of the hashing function and the node ID structure. Location can be part of the node ID hash space and thus nodes close on the physical space can also be close in the overlay space (low stretch factor). For example, Geo-Chord [16] constructs the Chord ring based on geographical locations in which the nodes are divided into different subregions based on their Euclidean distances. The head node for each subregion will make the main Chord ring. Subregion rings are constructed based on an Euclidean distance matrix and a hop distance matrix of the nodes in the network. Geo-Chord is more suited for static networks than dynamic topologies as the matrices will keep changing and the regions will have to be re-divided accordingly. In this paper, we present a hybrid routing algorithm that combines reliable flooding and Distributed Hash Tables (DHT) to find the closest nodes with the desired capabilities for task execution. Node IDs are assigned using an interleaved hashing function that ensure a low stretch factor.

III. Methodology

A. System Model

We consider an ad-hoc mobile cloud that is composed of m devices in a geographically constrained region. We let D denote the set of devices in the network, $D = \{d_1, d_2, ..., d_m\}$. We let C denote the set of capabilities, $C = c_1, c_2, ..., c_n$, where n represents the total number of distinct capabilities. Any device, d_i, will have a set of capabilities, c_{d_i}, such that $c_{d_i} \subseteq C$. We let λ denote the average arrival rate of devices and β denote the average time devices spend in the network (i.e., ON period). At any particular time t, the devices whose ON period range includes t, are said to be active devices in the network and their capabilities are provided as cloud services. Figure 1 shows an example of a capability matrix in which capabilities are represented as columns and active devices as rows. In this example, 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, d_3 or d_6 to execute tasks that require capability c_1 by offloading the task to one of those devices. Devices and their capabilities are hashed into a structured peer-to-peer network (e.g., Chord) and the capability matrix is stored in a Distributed Hash Table (DHT) that is maintained among all the devices. Figure 2 illustrates our local overlay mobile cloud at an instant with 6 active devices. Each device maintains a part of the DHT as shown in Figure 2 for devices d_2 and d_6. Throughout this paper, we consider a Chord network, but our approach would work with any structured peer-to-peer network.

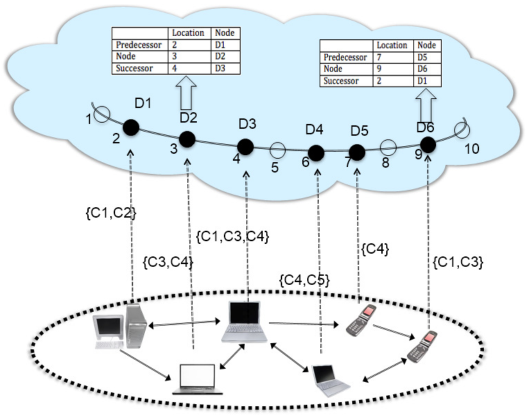


		Capabilities				
Devices		C_1	C_2	C_3	C_4	C_5
	D_1	1	1	0	0	0
	D_2	0	0	1	1	0
	D_3	1	0	1	1	0
	D_4	0	0	0	1	1
	D_5	0	0	0	1	0
	D_6	1	0	1	0	0

Fig. 1. Capability matrix.

B. Interleaved-Hashing

When a device joins the network, its node ID is assigned based on its location. We assume that each device knows its GPS coordinates. Indoor devices can get an estimate of their location using various Wifi localization strategies (e.g., [17]). The node ID structure is composed of two parts: a location prefix and a random bits suffix. The location prefix is obtained through interleaving the binary representation (without the floating point) of the latitude and longitude



	Location	Node
Predecessor	2	D1
Node	3	D2
Successor	4	D3

	Location	Node
Predecessor	7	D5
Node	9	D6
Successor	2	D1

Fig. 2. Local overlay-based mobile cloud.

coordinates of the device. In Figure 3 we illustrate the interleaving where X denotes the longitude value and Y denotes the latitude value . This ensures that the physical topology is efficiently embedded into the hash space. The random suffix reduces hashing collisions in scenarios in which two nodes – say on two different floors in a building – have the same GPS coordinates. These nodes will spread out in the hash space to neighboring locations.

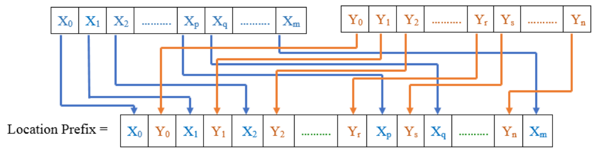

X_0 X_1 X_2 .... X_p X_q .... X_m
Y_0 Y_1 Y_2 .... Y_r Y_s .... Y_n
Location Prefix = X_0 Y_0 X_1 Y_1 X_2 Y_2 .... Y_r X_p Y_s X_q .... Y_n X_m

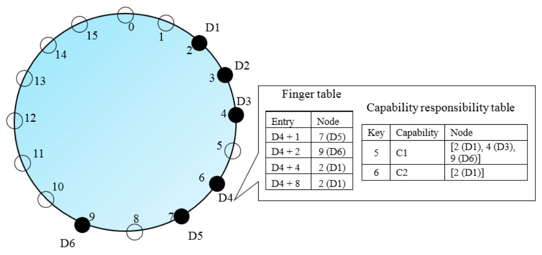
Fig. 3. Interleaved representation of location.

To decide on the number of bits allocated to the prefix and the suffix, we need to consider the maximum number of bits required to represent the latitude and longitude values based on some precision. If we consider 1 meter precision – which is good enough for tracking mobile devices – we will need to consider up to 5 digits after the decimal point for the coordinates. 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 fraction part, we need a maximum of 53 bits when 5 digits are considered (a total of 123 bits to implement the interleaving). If fewer bits are required for the representation, the most significant bits of the integral parts and the least significant bits in the fractional part will be 0's.

C. Routing Requests for Capabilities

A nodes that wishes to execute a capability that it does not have must be able to find a node on the mobile cloud that would provide it. The conventional way of searching on the ad-hoc network is flooding which we describe first, and then we discuss a modified routing scheme over Chord.

Sequence Number Controlled Flooding (SNCF) [18] is a form of reliable flooding in which packets are flooded into the network until they reach all the nodes. The packets are dropped by a node based on sequence numbers, time-to-live (TTL) field and who the sender is (e.g., a node will not forward a packet back to the sender). In our case, a node would also stop the flooding process if it has the required capability. Flooding consumes a considerable amount of bandwidth, as well as wastes CPU cycles and power for the devices. With flooding, popular capabilities will likely be found more quickly whereas rare capability would take more time.



Finger table

Entry	Node
D4 + 1	7 (D5)
D4 + 2	9 (D6)
D4 + 4	2 (D1)
D4 + 8	2 (D1)

Capability responsibility table

Key	Capability	Node
5	C1	[2 (D1), 4 (D3), 9 (D6)]
6	C2	[2 (D1)]

Fig. 4. Modified Chord.

With Chord, devices in the physical topology are mapped onto the virtual Chord ring, and each node's ID is assigned using a hash value (e.g., SHA-1). In our modified Chord, each node's ID is assigned using the interleaved hashing method such that the overlay topology becomes congruent with the physical topology. A key associated with a capability is obtained from the same hash function and is mapped onto the node with the smallest ID greater than or equal to the key (i.e., the successor node). 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 the nodes that can execute the capability. A single node can be responsible for multiple capabilities.

To identify whether a particular capability is present in the network, the node will hash the capability ID and a lookup message is sent to the node with highest ID less than or equal to the hashed value in its finger table. If the node hash ID is equal to the hash value, or the hash value 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 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 sent to 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) received the lookup message again, that means the capability does not exist in the network, and that node will send a response to the node that raised the lookup.

To efficiently lookup a key, every node with ID i will build and maintain a finger table to other nodes with up to p fingers from $(i+2^k) \mod 2p$ where k is an integer $(0 \leq k \leq p)$ as shown in Figure 4. A node will have most of its fingers in the forward direction (i.e., clockwise direction) and they are spaced by a power of 2 factor. Figure 4 shows the finger table what maintains Node IDs and IP addresses and capability responsible table that maintains the corresponding capable node IDs. Nodes leaving the ring and packet formats are achieved in the same manner as in Chord [7].

D. *Hybrid Routing Protocol*

Flooding is the fastest way to find a highly replicated capability whereas DHTs would preform better in finding rare capabilities. To take advantage of both, we present a hybrid routing protocol. There are some existing works [19]–[22] that proposed the use of hybrid routing protocols. In these protocols, the selection of the routing mechanism is based on the popularity of the resources as collected by some nodes (e.g., ultrapeers). Depending on the popularity of certain capabilities, a node may choose between one of those methods. If, however, this information is not available, we propose a hybrid routing protocol that works as follows: when a request for a capability is raised, we flood the query with a limited TTL to hear the reply quickly if the capability is highly replicated and one of the capable node is within the specified number of hops, and in parallel, we issue the lookup over the DHT. 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 number of messages generated. By issuing a query in parallel, there will be message overhead from both 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 ones.

E. *Multiple responsible nodes*

As the size of the network grows, the number of hops to find a responsible node can be high which increases the average response time and the number of lookup messages. Moreover, the Chord lookup process is not symmetric as the number of fingers to nodes in the first half of the ring (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 closer to the requested node in the forward direction, it can be reached in less number of overlay hops than if the responsible node was 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 plan to explore methods for setting this parameter dynamically in the future, but our results show the impact of this number.

IV. EVALUATION

In this section we present simulation results for assessing the performance of our proposed mobile cloud platform.

A. *The setup*

We consider a mobile cloud in an area that is 100 meters by 100 meters. The GPS coordinates of the devices are generated uniformly at random in that range. The transmission range of each device was set to 30 meters. Devices join 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. The devices' ON period follows a Pareto distribution with an exponent of 0.83. This heavy-tailed distribution is usually observed in P2P networks in which few devices stay for long periods of time while the majority leave over short intervals. We held the average ON period to 5 minutes, corresponding to a departure rate of 1 device every 5 min. Each device is assigned a random number capabilities in the range [0,4]. We assigned capabilities to devices following a uniform distribution and Zipf's distribution. Zipf's distribution is used to reflect that many capabilities are highly replicated and few of them are rare. We used 1 as the exponent in Zipf's distribution so that every capability occurs at least once in the network. Each node in the network will raise a random number of lookup requests (i.e. tasks) in the range [0,2] uniformly at random. We report on results generated between the first node's departure and the last node's arrival to account for a stable network. All experiments are repeated 5 independent runs and the results are averaged.

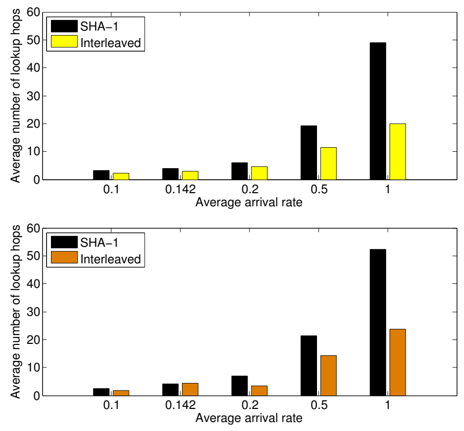


Fig. 5. Average number of lookup hops under uniform (top) and Zipfs (bottom) distributions.

B. *Interleaved Hashing*

Figure 5 shows the effect of our interleaved hashing method in comparison to SHA-1 on the average number of lookup hops traversed under different capability distributions (uniform top and Zipf's bottom). With interleaved hashing, there is 40% decrease in the average number of lookup hops under the uniform distribution and 33% decrease under Zipf's distribution. Interleaved hashing outperforms SHA-1 as the arrival rate increases.

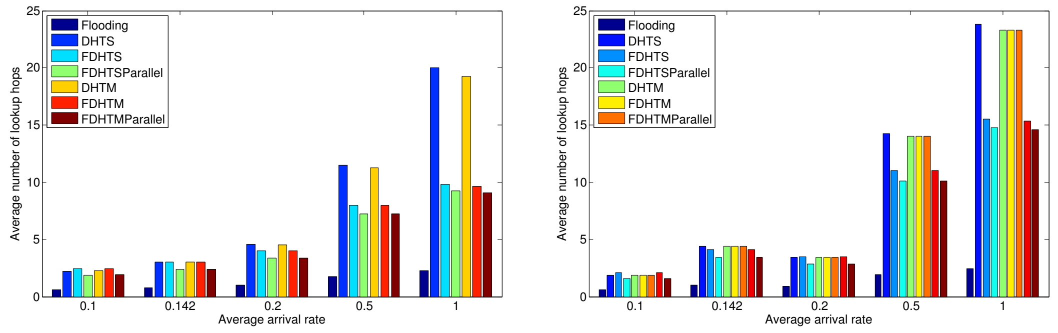


Fig. 6. Average number of lookup hops traversed under uniform (left) and Zipf's (right).

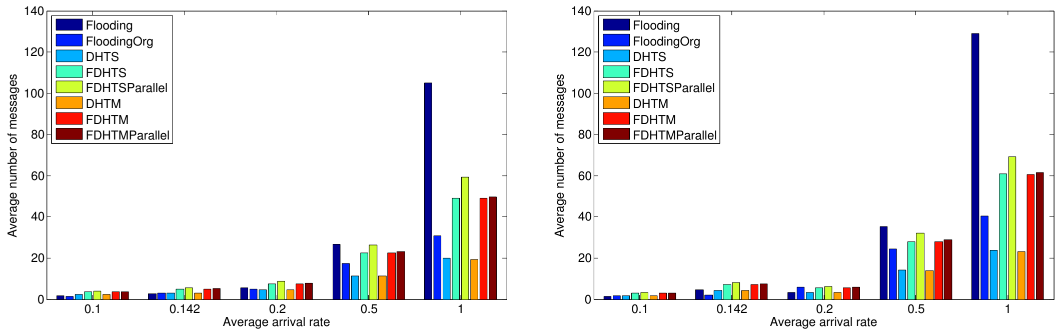


Fig. 7. Average number of messages generated under uniform (left) and Zipf's (right).

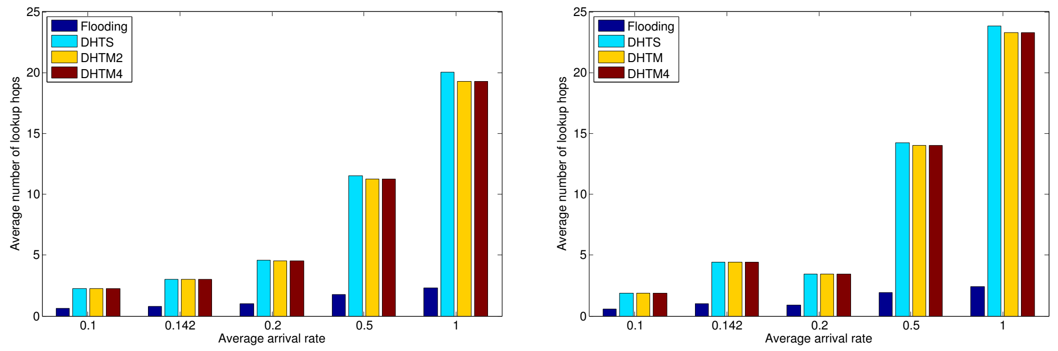


Fig. 8. Average number of lookup hops traversed under uniform (left) and Zipf's (right) with multiple responsible nodes.

C. Hybrid Routing Protocol

We have experimented with different versions of our hybrid routing algorithm. The following is the list of methods we compare between: (1) DHTS - Chord DHT with a single responsible node for each capability, (2) DHTM (n) - Chord DHT with Multiple (n) responsible nodes for each capability, (3) FDHTS Hybrid algorithm when flooding and DHTS are run in serial (flooding till the specified number of lookup hops and if the capable node is not found, the query is issued in DHT), (4) FDHTS Parallel Hybrid algorithm when flooding and DHTS are run in parallel, (5) FDHTM Hybrid algorithm when flooding and DHTM are run in serial, (6) FDHTM Parallel Hybrid algorithm when flooding and DHTM are run in parallel, and (7) Flooding.

Figure 6 shows the average number of lookup hops under uniform (left) and Zipf's (right) capability distributions. In flooding, we limited the TTL to 2 hops and we considered only 2 responsible nodes whenever multiple responsible nodes are considered. In general, the hybrid versions perform better than issuing the query in DHT. This difference increases as the the load on the network increases. When the arrival rate is just above twice the departure rate, the average number of lookup hops under FDHTMS is 30% less than DHTM when capabilities are uniformly distributed and 20% less when capabilities follow Zipf's distribution. The average number of lookup hops with the hybrid parallel protocol is less than all protocols (clearly except flooding) under all network load. For example, under high load ($\lambda = 1$), we can see a decrease of 50% with FDHTM Parallel compared to DHTM under uniform and 37% decrease under Zipf's distribution.

With hybrid parallel routing protocols, the response time (average number of hops traversed) is decreased but there will be an increase in the number of messages generated per request. Figure 7 shows the average total number of lookup messages generated to resolve a request. In this figure, floodingOrg counts the number of messages generated until a response is received by the requester. We can see that under the hybrid parallel protocol (FDHTM Parallel), the average number of lookup messages is higher than DHTM and lower than flooding. Under high load ($\lambda = 1$), the parallel protocols achieve 50% decrease in the average number of messages generated when compared to flooding with a uniform capability distribution and 52% decrease with Zipf's distribution.

D. Multiple Responsible Nodes

We studied the impact of the number of responsible nodes so that nodes can find closer responsible nodes to reduce the average number of lookup hops. Figure 8 shows the average number of lookup hops with varying number of responsible nodes. In general, 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. When the capabilities are assigned to devices following the uniform distribution, and under low load, the number of responsible nodes does not 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 average number of lookup hops. There is a diminishing return effect in adding more responsible nodes. Due to the relative small size of the network considered, we found that increasing the number of responsible nodes beyond two does not decrease the average response time further. It is worth noting that under Zipf's distribution, there is a higher chance of not finding a certain capability in some cases. These cases would incur the same cost irrespective of the the number of responsible nodes in the network.

V. Conclusions and Future Work

In this paper, we presented a collaborative computing mobile cloud that allows a group of mobile devices to form an ad-hoc network and advertise their capabilities as cloud services over an overlay network. The mobile cloud capitalizes on locality that is achieved through the interleaved hashing method of the devices' coordinates to induce congruity between the physical and the overlay topology. Hybrid protocols are devised that combine the benefits of flooding and hash-based lookups in DHTs. We evaluated the performance of our mobile cloud through simulation experiments in which we considered variants of hybrid protocols. Depending on the parameters used, we showed how these variants tradeoff between bandwidth (captured by the number of messages generated) and time (captured by the number of hops traversed).

Acknowledgment

This research is funded in part by NSF CNS award #1149397 and REU award #1156712 that is co-funded by the Department of Defense.

References

[1] OSGi Alliance Staff, "OSGi Alliance," http://www.osgi.org/Main/HomePage, 2013.

[2] M. Guirguis, R. Ogden, Z. Song, S. Thapa, and Q. Gu, "Can you help me run these code segments on your mobile device?" in *Global Telecommunications Conference (GLOBECOM 2011), 2011 IEEE*. IEEE, 2011, pp. 1–5.

[3] T. Langford, Q. Gu, A. Rivera-Longoria, and M. Guirguis, "Collaborative computing on-demand: Harnessing mobile devices in executing on-the-fly jobs," in *Mobile Ad-Hoc and Sensor Systems (MASS), 2013 IEEE 10th International Conference on*. IEEE, 2013, pp. 342–350.

[4] T. Penner, A. Johnson, B. Van Slyke, M. Guirguis, and Q. Gu, "Transient clouds: Assignment and collaborative execution of tasks on mobile devices," in *Global Communications Conference (GLOBECOM), 2014 IEEE*. IEEE, 2014, pp. 2801–2806.

[5] E. Miluzzo, R. Cáceres, and Y. Chen, "Vision: mClouds - Computing On Clouds of Mobile Devices," in *Proceedings of the third ACM workshop on Mobile cloud computing and services*, UK, June 2012.

[6] A. Sciarrone, I. Bisio, F. Lavagetto, T. Penner, and M. Guirguis, "Context awareness over transient clouds," in *2015 IEEE Global Communications Conference (GLOBECOM)*, 2015, pp. 1–5.

[7] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," *ACM SIGCOMM Computer Communication Review*, vol. 31, no. 4, pp. 149–160, 2001.

[8] E. Cuervo, A. Balasubramanian, D.-k. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl, "Maui: making smartphones last longer with code offload," in *Proceedings of the 8th international conference on Mobile systems, applications, and services*. ACM, 2010, pp. 49–62.

[9] M. S. Gordon, D. A. Jamshidi, S. Mahlke, Z. M. Mao, and X. Chen, "Comet: Code offload by migrating execution transparently," in *Presented as part of the 10th USENIX Symposium on Operating Systems Design and Implementation (OSDI 12)*, 2012, pp. 93–106.

[10] Y. Zhang, G. Huang, X. Liu, W. Zhang, H. Mei, and S. Yang, "Refactoring android java code for on-demand computation offloading," in *ACM SIGPLAN Notices*, vol. 47, no. 10. ACM, 2012, pp. 233–248.

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

[12] C. Perkins, E. Belding-Royer, and S. Das, "Ad hoc on-demand distance vector (aodv) routing," Tech. Rep., 2003.

[13] (2013) Google android development. [Online]. Available: "Android4.0APIs",http://developer.android.com/about/versions/android-4.0.html

[14] J. Thomas and J. Robble, "Off grid communications with android," *The MITRE Corporation*, 2012.

[15] T. Clausen, P. Jacquet, C. Adjih, A. Laouiti, P. Minet, P. Muhlethaler, A. Qayyum, and L. Viennot, "Optimized link state routing protocol (olsr)," 2003.

[16] A. Pethalakshmi and C. Jeyabharathi, "Geo-chord: Geographical location based chord protocol in grid computing," *International Journal of Computer Applications*, vol. 94, no. 3, 2014.

[17] K. Chintalapudi, A. Padmanabha Iyer, and V. N. Padmanabhan, "Indoor localization without the pain," in *Proceedings of the sixteenth annual international conference on Mobile computing and networking*. ACM, 2010, pp. 173–184.

[18] A. Tanenbaum and D. Wetherall, "Computer networks," 2011.

[19] B. Leong, B. Liskov, and E. D. Demaine, "Epichord: parallelizing the chord lookup algorithm with reactive routing state management," *Computer Communications*, vol. 29, no. 9, pp. 1243–1259, 2006.

[20] B. T. Loo, R. Huebsch, I. Stoica, and J. M. Hellerstein, "The case for a hybrid p2p search infrastructure," in *Peer-to-Peer Systems III*. Springer, 2004, pp. 141–150.

[21] M. Zaharia and S. Keshav, "Adaptive peer-to-peer search," *University of Waterloo Technical Report*, vol. 55, 2004.

[22] ——, "Gossip-based search selection in hybrid peer-to-peer networks," *Concurrency and Computation: Practice and Experience*, vol. 20, no. 2, pp. 139–153, 2008.