EPiC Series in Computing

Volume 91, 2023, Pages 86–97

Proceedings of 38th International Conference on Computers and Their Applications

Design of a New Secured Hierarchical Peer-to-Peer Fog Architecture Based on Linear Diophantine Equation

Saydul Akbar Murad[1], Nick Rahimi[1], Indranil Roy[2], Bidyut Gupta[3]

[1] School of Computing Sciences & Computer Engineering, University of Southern Mississippi, Hattiesburg, MS

saydulakbar.murad@usm.edu

nick.rahimi@usm.edu

[2]Department of Computer Science, Southeast Missouri State University, Cape Girardeau, MO

iroy@semo.edu

[3] School of Computing, Southern Illinois University; Carbondale, IL

bidyut@cs.siu.edu

Abstract

In recent years, there has been massive growth in the usage of IoT devices. Cloud computing architecture is unable to meet the requirements of bandwidth, real-time response, and latency. To overcome these limitations, fog computing architecture is introduced, which responds to requests from IoT devices and only, if necessary, forwards requests to the cloud. Nonetheless, there are still some requests that need to go to the cloud and get affected by the shortcomings of the cloud. In this work, we propose to add a peer-to-peer (P2P) structure to the fog layer. We have considered our recently reported 2-layer non-DHT-based architecture for P2P networks in which at each level of the hierarchy existing networks are all structured and each such network has a diameter of 1 overlay hop. Such low diameters have huge significance in our proposed P2P fog model and improve fog computing by presenting very efficient data lookup algorithms. In this model, fog nodes can work together to complete the client requests. Consequently, fog nodes are able to fulfill the client requests in the fog layer, which ultimately decreased overheads on the cloud. Additionally, to improve the security in communication in the architecture, we have utilized ciphertext policy attribute-based encryption (CP-ABE) and presented a new secure algorithm.

1 Introduction

Devices on the internet produce humongous amounts of data, requiring strong server ends to store and retrieve their computational data. To meet this requirement of robust server ends, cloud computing opts as a solution where all the data outsourcing devices, like IoT devices, use cloud for information storage and retraaieval. As cloud computing is becoming an extensively used approach in today's world to deliver services to end users and provide applications with elastic resources at low cost, there is a need for efficient integration of cloud computing and end devices. But there are some shortcomings in the cloud computing architecture. For example, an autonomous vehicle produces one gigabyte of data per second to accommodate this amount of high-rate data exchange between the cloud and end devices, but due to the long and thin

connection between the cloud and end devices, network bandwidth stands as a challenge, on the other hand, clouds cannot guarantee low latencies which is a top requirement for IoT devices and also the real-time response is a bottleneck for cloud computing as many IoT devices rely on the real-time response time[1][2].

To cover all these shortcomings, the fog computing model was introduced. Cisco first introduced fog computing in 2012 [3]. Fog computing extends the concept of cloud computing by placing a fog layer between the cloud and the end users (mobile or static devices) such that the fog nodes are in proximity to end users. Fog nodes in the fog layer serve the requests of the end users at their respective proximity, and the cloud node is approached by fog nodes only when necessary. This ultimately reduces the bandwidth and overhead on the cloud node. Fog computing overshadowed all the shortcomings of cloud computing by saving bandwidth, providing low response time and real-time interactions due to its short-fat connection with end users, and providing support for mobility and even geographical distribution [4].

P2P fog computing is implemented by establishing a peer-to-peer connection between the fog nodes in the fog computing model [5]. P2P connection is established for routing the communication between the fog nodes, which allows the sharing of computing resources between the systems. Whenever a fog node does not have the resource that has been requested by the client, it first checks with fellow peers (fog nodes) in the P2P overlay before contacting the cloud node. If there is a peer (fog node) with that resource, there is a direct exchange of resources between them, else the cloud is requested. This model, when implemented with appropriate peer-to-peer architecture, will further reduce the bandwidth on the cloud node.

In this paper, our recently reported, linear Diophantine equation based hierarchical P2P (LDEPTH) architecture has been proposed to be used in a fog computing model [6] [7]. LDEPTH is an interest based, scalable, hierarchal peer-to-peer system that provides efficient data lookup operations and fault tolerance. Most of the structured P2P systems use Distributed Hash Table (DHT) to provide efficient data insertion and lookup services, but LDEPTH employs linear Diophantine equation (LDE) to realize the hierarchal P2P architecture [8]. LDE provides a lightweight mechanism to create and maintain peer-to-peer architecture as compared to DHT. The time complexity of searching a resource is independent of the number of nodes in a network with n number of nodes and instead is bounded by $(1+r/2)$, with r being the number of distinct resource types. As r is typically significantly smaller than the number of nodes. LDE based P2P systems provide highly efficient resource lookup procedures [9]. Our goal in this research is to propose a new architecture by employing LDEPTH as the overlay architecture in P2P fog computing for fog nodes, which ultimately decreased overheads on the cloud.

2 Distributed Hash Table P2P vs LDE- Based P2P

In this section, we discuss the working, advantages, and complexities of DHT-based P2P and LDE-based P2P. The goal of this section is to explain the reasons for choosing LDE-based P2P model over other DHT-based P2P to implement P2P fog computing.

2.1 DHT-Based P2P

All the DHT-based P2P systems come under the structure of P2P systems. A structured peer-to-peer system assigns keys to the data items and builds a graph that maps each key to the node that stores the corresponding data. There are many DHT-based P2P algorithms like Chord, Can, Pastry, Tapestry, etc. [10]. Before looking at the DHT-based P2P, we need to know what a Distributed Hash Table (DHT) is. A Hash Table is a special data structure that can map keys to values. It uses a special function called the hash function that takes the original key as input and outputs a key which is the unique numerical representation of the original key. The numerical key is mapped with its corresponding value. Hence data is stored in the form of (key, value) pairs in the Hash table. A DHT (key, value) pairs over millions of peers (The pairs are evenly distributed among peers). Any peer can query the database with a key, and the database returns the value for the key. In a DHT-based peer-to-peer system, a query is resolved by a small number of message exchanges among peers. Each peer only knows about a small number of other peers, but not all of them. DHT also has some strategies to handle churn. In a circular DHT, all the peers have arranged in the form of a ring (like a Chord). Each peer is assigned a (key, value) pair and a unique id from the id space. And each peer is only aware (has IP addresses) of its immediate successor and predecessor. When a peer inside the circular DHT receives a query message to know about a value associated with a certain key, it first checks if it is responsible for the value of the queried key. If it has the value, then it unicasts the data to the peer that sent the query message. Else it would forward the query message to its successor or predecessor depending on the key. This way, to solve a query message forwarded among the peers in the circular DHT until there is a query hit. Hence, we can say that the time complexity for a search in DHT-based P2P is O ($log\ n$), where n is the number of peers in the network. However, maintaining DHT and handling the problem of churn is a complex task and requires significant effort [11].

2.2 LDE-Based P2P

LDE-based P2P, Two-level Hierarchical (LDEPTH) network architecture is scalable, hierarchical P2P system which uses Linear Diophantine Equation (LDE) as a mathematical basis to realize the hierarchical P2P architecture instead of using DHT. In LDEPTH, a resource is represented as a tuple $< R_i, V >$, where R_i denotes the type of a resource and V indicates the value of the resource. A resource can have many values. For instance, let R_i denote the resource type songs and V_1 denotes a particular artist. Then, $< R_i, V_1 >$ will denote songs by artist V_1. Each peer has its own tuple. Also, no two peers should have the same tuple. **Figure 1** defines the details architecture of LDEPTH network [12].

LDEPTH is a Two-level P2P overlay. At level 2, all the peers having the same resource type are grouped together, and the first one to join the group is the group head. That means each resource type has an individual group at level 2. All the peers are directly connected to each other in the group and are only at a hop distance far from each other, forming an overlay

network with diameter 1. At level 1, all the individual group heads of different resource types form a ring-type network called Transit Ring Network. The diameter of this ring network can be at most $d/2$, where d is the number of distinct resource types. Any communication between peers of different groups happens only via respective group heads. Each peer in the Transit Ring Network is aware of its neighbors and each peer in the transit network maintains a global resource table (GRT) that consists of tuples of the form, <Resource type, Resource code, Group-head logical address> where Group-head Logical Address refers to the logical address assigned to a node by using the solutions of Linear Diophantine equation. To implement this

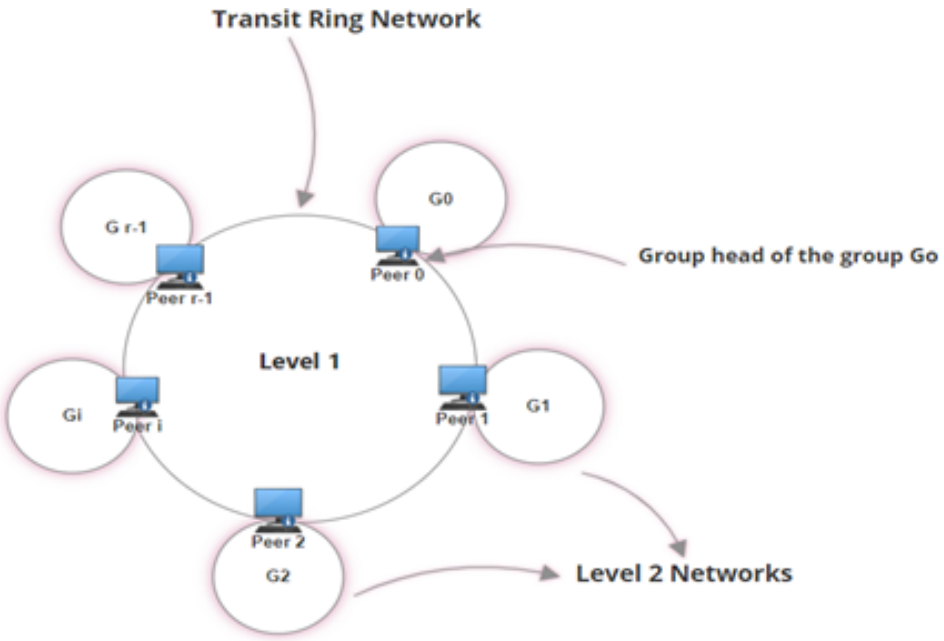

Figure 1: LDEPTH network architecture.

functionality, LDE is used as a mathematical basis. The solutions of LDE are used to assign logical addresses of peers in each group. The mutually incongruent modulo solutions of LDE are in the form n_0, $n_0 + C/d$, $n_0 + 2(C/d)$, . . . $n_0 + (d-1)\,(C/d)$. These solutions are assigned as logical addresses to group heads in Transit (level 1) network such that two peers in the level 1 network are neighbors if the logical address differs by (C/d), with the exception that the first and last peers will be considered as neighbors even if their logical address differ by $(r-1)$ (C/d). The GRT of peer 1(group-head of Resource type R_1) in the LDE network is depicted in **Table 1**. The mutually congruent solutions of LDE are of the form $n_0 + t(C/d) + mc$ *for* $0 \leq t \leq d$, and m is an integer. These solutions are assigned as group membership addresses for peers in the individual group in the level 2 network. As these solutions are mutually congruent, the use of these addresses will be shown to justify that all peers in a group are logically connected to each other to form an overlay network with diameter 1. Logical addresses of group heads are used in identifying peers that are neighbors to each other on the transit ring network, identifying each distinct resource type in the P2P network. From the structure of LDEPTH, it is clearly evident that the time taken for lookup operations is dependent on the number of resource types rather than the number of peers. The time

complexity of LDEPTH is bounded by $(1+r/2)$, with r being the number of distinct resource types, which is typically much smaller than the number of peers. At the same time, the DHT-based P2P has a time complexity of O ($log\ n$), with n being the number of peers. LDEPTH provides an efficient lookup procedure and is a lightweight mechanism that is easier to maintain than complex DHTs.

Table 1: Global Resource table of Peer 1.

Resource type	Resource code	Group-head Logical address
R1	Resource code of R1	$n_0 + C/d$
R2	Resource code of R2	$n_0 + 2C/d$
R3	Resource code of R3	$n_0 + 3C/d$
R4	Resource code of R4	$n_0 + 4C/d$

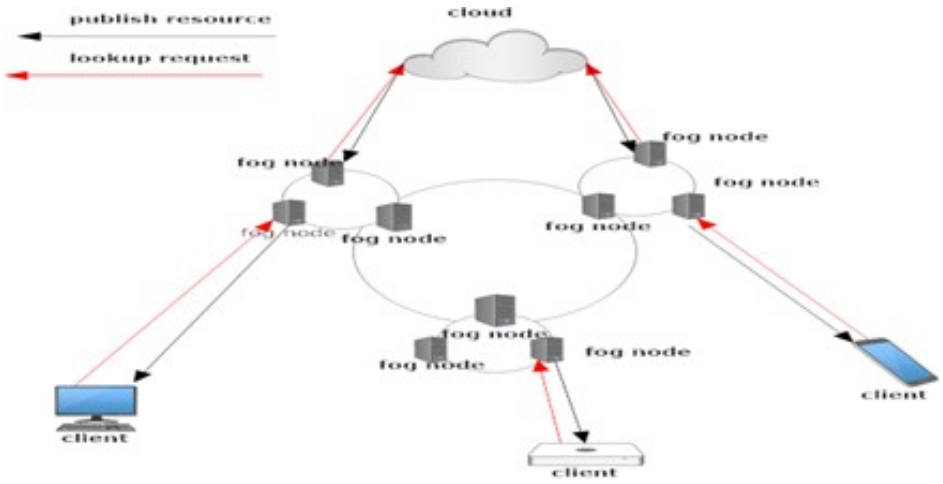

Figure 2: LDE based P2P fog computing architecture.

3 LDE-Based P2P Fog Computing Architecture

LDE-based P2P fog computing is a three-layer architecture consisting of client nodes in one layer, fog nodes in the second layer, and a cloud node in the third layer. LDE-based P2P fog computing architecture implements an LDEPTH network model to establish a connection between fog nodes to facilitate resource lookup and data transfers between them. as fog nodes are now connected together through P2P and use LDEPTH lookup procedures to search for files. **Figure 2** depicts the architecture of LDE-based peer-to-peer fog computing. As the services are stored by implementing LDEPTH when a client node requests its corresponding fog node for some resource or data. In that case, there are four different possibilities.

1. The requested fog node is the group-head of the requested resource type.
2. The requested fog node is the group member of the group of the requested resource type.
3. The requested fog node is the group-head of another resource type.

4. The requested fog node is the group member of the group of another resource type.

Case 1: If the requested fog node (f_i) is the group-head of requested resource type (R_i). If it possesses the resource, then it responds with the data to the client node that requested the resource. Else it broadcasts the request within the group and if any fog node (f_n) within the group possesses the resource, it sends it to the group-head(f_i), and the fog node (f_i), after receiving the data, responds to the client request. If the lookup inside the group fails, then the cloud node is contacted for the data. The pseudocode for Case 1 is shown in **Algorithm 1**.

Algorithm 1 Algorithm for Case 1

```
Node x requests a service (Ri) to its fog node fi;
fi is the group-head of service type (Ri);
if fi possesses Ri then fi unicasts service to x; else
    fi broadcasts the request for service Ri in its group Gi;
    if fn possesses Ri then
        fn unicasts service Ri to fi;
        fi responds to x with service Ri;
    else
        cloud node is contacted by fi for service Ri;
        cloud responds with Ri;
        fi caches the Ri and unicasts it to x;
    end if
end if
```

Case 2: A fog node(f_1) is requested for data of resource type (R_i) (**Algorithm 2**). If the requested fog node(f_1) is the group member of the requested resource type (R_i), it broadcasts the request within the group and if any fog node(f_n) within the group has the resource it sends to the fog node(f_1) and the fog node(f_1) after receiving the data, responds to the client request. If the lookup inside the group fails, then the cloud node is contacted for the data.

Algorithm 2 Algorithm for Case 2

```
Node x requests a service (Ri) to its fog node f1;
f1 is the group member of service type (Ri) which has a group-head fi;
if f1 possesses Ri then f1 unicasts service to x; else
    f1 broadcasts the request for service Ri in its group Gi;
    if fn possesses with Ri then
        fn unicasts service Ri to f1;
        f1 responds to x with service Ri;
    else
        cloud node is contacted by f1 for service Ri;
        cloud responds with Ri;
        fi unicasts the Ri to f1;
        f1 caches the Ri and unicasts it to x;
    end if
```

16: **end if**

Case 3: A fog node (f_1) is requested for data of resource type (R_i). If the requested fog node (f_1) is the group-head of another resource type (R_1) of a level 2 network in LDEPTH, then it checks its global resource table for the requested resource type (R_i) and if there is any group of that resource type then the request is forwarded to the group-head(f_i) of the resource type (R_i). If the group-head(f_i) of resource type (R_i) has the data, it responds, or else it broadcasts the request in its group; if any node in the group has the data, it unicasts it to its group-head(f_i), and f_i forwards the data to f_1 and f_1 forwards it to its client node that requested the data. If any of the cases fail, the cloud node is contacted for the data of that resource type. **Algorithm 3** defines the working procedure for Case 3.

Algorithm 3 Algorithm for Case 3

1: Node x requests a service (R_i) to its fog node f_1;
2: f_1 is group-head of another service type (R_i);
3: **if** f_1 has the address code of service type R_i's group-head f_i in its GRT **then**
4: f_1 computes $h = |n_0 + 1(C/d) - n_0 + i(C/d)|$;
5: **if** h > r/2 **then**
6: f_1 forwards the request along with its IP address to its predecessor;
7: **else**
8: f_1 forwards the request along with its IP address to its successor;
9: **end if**
10: All the intermediate group heads pass the request until it reaches f_i;
11: **if** f_1 possesses R_i **then**
12: f_i unicasts the service to f_1;
13: f_1 unicasts service to x;
14: **else**
15: f_1 broadcasts the request for service R_i in its group G_i;
16: **if** f_n with R_i **then**
17: f_n unicasts service R_i to f_1;
18: f_i unicasts service R_i to f_1;
19: f_1 responds to x with service R_i;
20: **else**
21: cloud node is contacted by f_i for service R_i;
22: cloud responds with R_i;
23: f_i caches the R_i and unicasts it to f_1;
24: f_1 unicasts service to x;
25: **end if**
26: **end if**
27: **else**
28: cloud node is contacted for R_i;
29: cloud node responds with R_i to f_1;
30: f_1 unicasts service R_i to x;
31: A new level 2 network is created with service type R_i;
32: A fog node (f_i) is allocated as its group head;
33: All the group heads are intimated to update their GRT;
34: **end if**

Case 4: A fog node (f_2) is requested for data of resource type (R_i) (**Algorithm 4**). If the requested fog node (f_1) is the group member of another resource type (R_1) of a level 2 network in LDEPTH, it forwards the request to its group-head(f_1). The group-head f_1 checks its global resource table for the requested resource type (R_i), and if there is any group of that resource type, then the request is forwarded to the group-head(f_i) of the resource type (R_i).

If the group-head(f_i) of resource type (R_i) has the data, it responds, or else it broadcasts the request in its group; if any node in the group has the data, it unicasts it to its group-head(f_i), and f_i forwards the data to f_1, and f_1 forwards it to f_2 and f_2 sends the data to the client node that requested the data. If any of the above cases fails, the cloud node is contacted for the data of that resource type.

Algorithm 4 Algorithm for Case 4

1: Node x requests a service (R_i) to its fog node f_2; 2: f_2 is a group member of another service type (R_1); 3: f_2 send the request to its group-head f_1;
4: **if** f_1 has the address code of service type R_i's group-head f_i in its GRT **then**
5: f_1 computes $h = |n_0 + 1(C/d) - n_0 + i(C/d)|$;
6: **if** h > r/2 **then**
7: f_1 forwards the request along with its IP address to its predecessor;
8: **else**
9: f_1 forwards the request along with its IP address to its successor;
10: **end if**
11: All the intermediate group heads pass the request until it reaches f_i;
12: **if** f_1 possesses R_i **then**
13: f_i unicasts to f_1;
14: f_1 unicasts service to f_2; 15: f_2 unicasts service to x; 16: **else**
17: f_1 broadcasts the request for service R_i in its group G_i;
18: **if** f_n with R_i **then**
19: f_n unicasts service R_i to f_1;
20: f_i unicasts to f_1;
21: f_1 unicasts to f_2;
22: f_1 responds to x with service R_i;
23: **else**
24: cloud node is contacted by f_i for service R_i;
25: cloud responds with R_i;
26: f_i caches the R_i and unicasts it to f_1;
27: f_1 unicasts R_i to f_2;
28: f_2 unicasts service to x;
29: **end if**
30: **end if**
31: **else**
32: cloud node is contacted for R_i;
33: cloud node responds with R_i to f_1;
34: f_1 unicasts R_i to f_2;
35: f_2 unicasts service R_i to x;
36: A new level 2 network is created with service type R_i;
37: A fog node (f_i) is allocated as its group head;
38: All the group heads are intimated to update their GRT;

39: **end if**

In LDEPTH fog computing, the use of LDE and the same logical address to denote a resource type R_i and the corresponding group-head has made the search process simple and efficient.
This has two significant advantages:

- The maximum number of hops required per any resource search is $r/2$, where r is the number of distinct resource types.
- As an alternative resource lookup process, a group-head can directly unicast a message any other fog node, without having to route through the other group heads in the transit network. This would allow our resource look up process to work with a constant number of message exchanges.

This delivers rapid localized services and decreases data movement. Therefore, it fulfills the need for real-time communication and reduces latency among applications of IoT devices. Moreover, fog nodes are able to provide many computational services locally and only send some requests to the cloud which can save bandwidth efficiently.

4 Security in LDE-Based P2P Fog Computing Architecture

Fog computing, in general, is just a cloud that is geographically close to ending devices or clients. So, it inherits all the security challenges that a cloud would face. There is a dire need for secure communication between the fog nodes. If fog is handling different kinds of confidential data such as health related, national privacy, among others. Protecting the data from eavesdropping or any middle-man attack is of the essence. Fog computing has many challenges in terms of security like authentication, trust, intrusion detection, etc.

In this section, we propose an additional layer of security in the level 1- transit network of LDE-based fog computing architecture. Level 2 networks use symmetric keys for encryption and decryption, which in general, are difficult to break. The structure would become much more secure if we combined an additional security layer at level 1. We implement Ciphertext Policy Attribute-based Encryption (CP-ABE) to impose additional security at level 1 (transit network). **Algorithm 5** defines the pseudocode for secure communication in Transit network. In CP-ABE, the identifier of a user is described by some specific attributes, and the authorized attribute sets constitute access policy which can be embedded into cipher text (Ciphertext-policy ABE, CP-ABE). For CP-ABE, a user with attributes satisfying their policy can access the data encrypted under the access policy. Therefore, CP-ABE succeeds in achieving owner-enforced fine-grained access control on outsourced data.

In our LDE-based P2P fog computing, each group-head has its own unique attribute set such as its resource type, resource code, and its logical address. Also, it contains information about other group heads resource types, resource codes, and logical addresses in its Global Resource Table (GRT). Every group-head requesting a service or responding with a service, encrypts the request or response using CP-ABE, where the encrypted request or response is associated with an access structure. This access structure is defined by the sender based on the attribute set of the intended receiver. So that a receiving group-head can decrypt the ciphertext and obtain the shared key only if it possesses the specified attributes in the access structure. In our LDE-based P2P fog, every group-head has the necessary information (resource type, resource code, logical address) of every other group-head to define an access

structure on them. This means each group-head can successfully define an access structure on any other group-head by performing CP-ABE if it wants to communicate (request or response).

Assuming that all the intra-group communication in level 2 networks is implemented using symmetric (shared key) cryptography. Below is the algorithm of how a group-head uses CP-ABE for secure communication in the transit network in LDE-P2P fog computing.

The algorithm explains how f_i requests a service and gets the response from other group heads by using CP-ABE as 1 layer (1st level) encryption and asymmetric encryption as the second layer (2nd level encryption).

Algorithm 5 Algorithm for secure communication in Transit network

Group-head f_i of service type R_1 wants to request for a service of type R_n;
if f_i found the Group-head f_n of resource type R_n in its GRT **then**
    f_i encrypts the request, which also contains a shared
    key to be used by f_n // 1st level CP-ABE by the sender;
    f_i defines an access policy based on attributes of f_n //resource type, resource code, the logical address of f_n;
    f_1 computes $h = |n_0 + 1(C/d) - n_0 + i\,(C/d)|$;
    if h > r/2 **then**
        f_i passes its desired resource type (R_n) along with the 1st-level encrypted request;
        fi performs additional 2nd encryption, using the public key of its predecessor f_0;
        f_i forwards the second encrypted request with its IP address to its predecessor f_0;
        f_0 can only decrypt the second-level encrypted message with its private key;
        f_0 computes 'h' value using the logical address of f_n;
            f_0 uses 'h' to decide to second encrypt the request with a public key of the successor or predecessor;
    else
        f_i passes its desired resource type (R_n) along with the 1st-level encrypted request;
        fi performs additional second encryption using the public key of its predecessor P_g;
        P_1 forwards the second encrypted request with its IP address to its predecessorP_g;
        P_g can only decrypt the second-level encrypted message with its private key;
        P_g computes 'h' value using the logical address of f_n.
            P_g uses 'h' to decide to second encrypt the request with the public key of the successor or predecessor;
    end if
    All the intermediate group heads pass the request until it reaches f_n;
    f_n receives the second layer encrypted request;
    First decrypts the second layer encrypted request with its private key;
    f_n Understands that it is the intended receiver;
        It can now decrypt the 1st layer encryption as it possesses the attributes defined in the access policy;
    It gets access to the shared key and performs decryption;
    if f_n in group n with resource R_n **then**
        f_n performs CP-ABE encryption on the response //defines access based on attributes of f_1;

f_n unicasts the encrypted response to P_1;
else
Cloud node is contacted for the service R_n;
f_n, upon receiving, caches the service R_n;
f_n performs CP-ABE encryption on the response R_n; unicasts the response to f_i;
end if
f_1 decrypts the response from f_n;
f_1 can access the service or can unicast it to any member in its group;
end if

5 Conclusion

In this paper, we have presented LDEPTH fog Computing - a new approach to designing a scalable fog computing system which provides highly efficient data lookup operations and therefore reducing the latency and providing faster localized services. Given that LDEPTH fog can handle more computational demands locally, the number of requests that need to be sent to the cloud reduces drastically which saves Internet bandwidth effectively. Different types of cloud services like SaaS, PaaS, and IaaS can be clustered at different level 2 networks in LDEPTH fog computing model. We also considered security in communication in the generalized architecture by using ciphertext policy attribute-based encryption. We intend to further extend the proposed LDEPTH fog architecture to handle multi-layer hierarchical P2P fog structures and compare its performance with existing fog architecture.

References

[1]. Dillon, T., Wu, C., & Chang, E. (2010, April). Cloud computing: issues and challenges. In 2010 24th IEEE international conference on advanced information networking and applications (pp. 27-33). Ieee.
[2]. Sajid, M., & Raza, Z. (2013, December). Cloud computing: Issues & challenges. In International Conference on Cloud, Big Data and Trust (Vol. 20, No. 13, pp. 13-15). sn.
[3]. Yi, S., Li, C., & Li, Q. (2015, June). A survey of fog computing: concepts, applications
and issues. In Proceedings of the 2015 workshop on mobile big data (pp. 37-42).
[4]. Yi, S., Hao, Z., Qin, Z., & Li, Q. (2015, November). Fog computing: Platform and applications. In 2015 Third IEEE workshop on hot topics in web systems and technologies (HotWeb) (pp. 73-78). IEEE.
[5]. Vaquero, L. M., & Rodero-Merino, L. (2014). Finding your way in the fog: Towards a comprehensive definition of fog computing. ACM SIGCOMM computer communication Review, 44(5), 27-32.
[6]. Rahimi, N., Sinha, K., Gupta, B., Rahimi, S., & Debnath, N. C. (2016, July). LDEPTH:
A low diameter hierarchical p2p network architecture. In 2016 IEEE 14th International Conference on Industrial Informatics (INDIN) (pp. 832-837). IEEE.
[7]. Gupta, B., Rahimi, N., Rahimi, S., & Alyanbaawi, A. (2017, July). Efficient data lookup in non-DHT based low diameter structured P2P network. In 2017 IEEE 15th International Conference on Industrial Informatics (INDIN) (pp. 944-950). IEEE.

[8]. Rahimi, N. (2020). Security consideration in peer-to-peer networks with a case study application. International Journal of Network Security & Its Applications (IJNSA) Vol, 12.
[9]. Rahimi, N., Gupta, B., & Rahimi, S. (2018). Secured data lookup in LDE based low diameter structured P2P network. In Proc. CATA.
[10]. Lua, E. K., Crowcroft, J., Pias, M., Sharma, R., & Lim, S. (2005). A survey and comparison of peer-to-peer overlay network schemes. IEEE Communications Surveys & Tutorials, 7(2), 72-93.
[11]. Rhea, S., Geels, D., Roscoe, T., & Kubiatowicz, J. (2004, June). Handling churn in a DHT. In Proceedings of the USENIX annual technical conference (Vol. 6, pp. 127-140).
[12]. Rahimi, N, "A Novel Linear Diophantine Equation-Based Low Diameter Structured Peer-to-Peer Network" (2017). Dissertations.