Generic Network Location Service

Laban Mwansa[1], Jan Janeček[2]

Czech Technical University in Prague, Department of Computer Science and Engineering, Karlovo nam. 13, CZ12135 Praha 2, Czech Republic

ABSTRACT

The goal of this work is to present a specific device key translation solution among different identifications of devices based on the DHT (Distributed Hash Table) technology. Whereas nodes in DHTs are obviously used to store data identified by a single identification key and their possible failures are resolved by approaches of multiple data copies. This paper presents the Generic Network Location Service based on the Chord DHT.

The provided algorithms guarantee resilience in the presence of dynamism: they guarantee consistent lookup results in the presence of nodes failing and leaving. Generic Network Location Service provides a Location Service system based on DHT technology, which is storing device location records in nodes within a Chord DHT.

Location records are consisting of network device identification keys as attributes, which are used to create replicas of additional location records through established Chord hashing mechanisms. Storing device location records, in places addressable (using the DHT lookup) by individual location record keys provides a simple way of implementing translation functions similar to well known network services (e.g. ARP, DNS, ENUM).

The generic network location service presented in the paper is not supposed to be a substitution of the existing translation techniques (e.g. ARP, DNS, ENUM), but it is considered as an overlay service that uses data available in existing systems and provides some translations currently unavailable.

ACM Categories & Subject Descriptors

B.8.1 **[Performance and Reliability]:** *Reliability, Testing, and Fault-Tolerance*

C.2.4 **[Computer Communication Networks]:** Distributed Systems – *distributed applications*

H.3.4 **[Information Storage and Retrieval]:** Systems and software – *distributed systems*

GENERAL TERMS

Design, Performance, Reliability

KEYWORDS

Broadcast, DHT, location service, hash table, resilience, location record

1. INTRODUCTION

The main contribution of this work involves the analysis of several algorithms implemented in the Generic Network Location Service (GNLS). We describe several communications algorithms namely: Sequential clockwise denoted as SEQUENTIAL-C, Lookup is initiated by the node using the first alive successor on the Chord DHT ring reaching all nodes in the network in O(n), where n is the number of nodes in the network. Sequential anti-clockwise SEQUENTIAL–CCW, Lookup is initiated by the node using the first alive predecessor on the ring network reaching all nodes in O(n) anti-clockwise, Simultaneous parallel Sequential clock-wise and anti-clockwise SEQUENTIAL-B, here both sequential clockwise and anti-clock are initiated at the same time reaching all nodes in the network in O(n/2) and lastly but not least the Broadcast algorithm. Illustration of the simplest mechanism that exploits multiplicity of location records in Generic Network Location Service (GNLS).

A DHT presents itself as an infrastructure, essentially a hash table which is distributed among a set of cooperating computers, which we refer to as nodes. The main service provided by a DHT is a lookup operation, which return a value associated with a given key (e.g. IP address). Chord [27] assigns ids from the same one dimensional id space $< 0, N - 1 >$ to both names and nodes. The id space wraps around to form a circle. Chord performs lookups in O(logN) time, using a finger tables of logN entries. A node's finger table contains the node id of a node halfway around the id space from it, a quarter-of-the-way, and so forth in powers of two. A node forwards a query for key k to the node in its finger table with the highest id less than k. The power-of-two structure of the finger table ensures that the node can always forward the query at least half of the remaining id space distance to k. As a result, Chord lookups use O(logN) messages.

Chord ensures correct lookups despite node failures using the successor list. Each node keeps track (of the IP addresses) of the next r nodes immediately following it in the id space.

[1] E-Mail: lmmwansa@yahoo.com

[2] janecek@cs.felk.cvut.cz

This allows a query to make incremental progress in id space even if many finger table entries turn out to point to crashed nodes. The only situation in which Chord cannot guarantee to find the current live successor to a key is if all r of a nodes immediate successors fail simultaneously, before the node has a chance to correct its successor list. Since node ids are assigned randomly, the nodes in a successor list are likely to be unrelated, and thus suffer independent failures. In such a case, relatively small values of r (such as logN) make the probability of simultaneous failure vanishingly small.

An overlay network is built on top of an underlying physical network topology. Messages between the nodes in the overlay network logically follow the ring topology of the overlay network, but physically pass through the links and routers that form the underlay network as shown in figure 1.

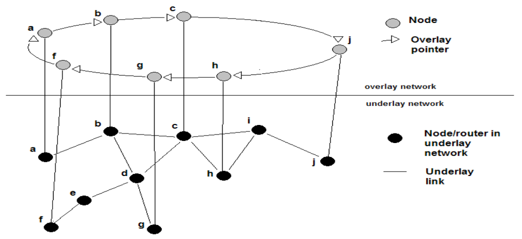


Figure 1: Physical underlay network with 10 nodes and an overlay network formed of 7 nodes

Applications of DHT technology for Internet name service support are numerous. As examples we can present DDNS (Dynamic DNS) [4] based on Kademlia and Microsoft's PNRP (Peer Name Resolution Protocol) [8] based on Pastry and supporting IPv6 name service in Windows XP and Vista.
Another area of DHT applications is device registration and lookup in SIP (Session Initiation Protocol). Original SIP has been based on SIP servers, accessible by DNS. Current development in the frame of IETF is based on DHTs and it resulted in several propositions, an example of the SIP implementation based on OpenDHT was presented in [17].

In [1], the authors describe a DHT overlay network based on Chord [11] supporting SIP service. The authors discuss important issues of security and authentication, as well as adaptations of conventional P2P routing for the social networks typical in personal communications.
The authors of [5] propose a name system that combines DHT and DNS, and describe how to eliminate bottleneck between the two name systems. Using the distributed nature of DHT, cost consuming processes such as protocol translation are distributed. A set of gateways that execute DNS name delegation dynamically is used to bind between a client side DNS resolver and translators running on DHT nodes. In contrast, our work does not propose a new name translation system but seeks to complement the lookup effort of existing technologies.

[15] Presents a global location service based on hierarchical DHTs. Authors propose a Location Information Plane (LIP) design based on hierarchical DHT that supports networks based on the Session Initiation Protocol (SIP).

Web based locations services have also received considerable research attention. [12] Proposed decentralized web services discovery mechanism (DWSDM) based on a DHT to solve web originated location problems.

All these efforts are directed towards achieving efficient discovery mechanisms. [6] Shows how P2P technologies can be integrated in load balancing and failover requirements with a centralized VoIP server concept. GNLS was inspired by previous work on location lookup services and name resolution technologies based on DHTs.

In this paper we describe the implementation of the Generic Network Location Service (GNLS). The GNLS system consists of the servers, GNLS nodes, which provide a distributed service that allows network devices, GNLS clients, to insert, lookup, and remove location records consisting of several keys, using these keys as handles.

This paper is organized as follows: first we briefly present the Generic network Service (GNLS); we then describe the GNLS device identification keys. In section 2.2, GNLS resilience to node failures is presented. Further, proactive and reactive strategies are briefly discussed. Experiments based on GNLS reactive algorithms are presented from section 3 to section 4 including algorithm complexity measured in terms of messages exchanged and time consumption. We assume GNLS implemented in the Chord DHT ring using two arbitrary location records. The experiments compare the algorithms with standard Chord lookup algorithm herein referred to as Route. We end with conclusion and a list of references.

2. GENERIC NETWORK LOCATION SERVICE

The GNLS system consists of the servers, GNLS nodes, which provide a distributed service that allows network devices, GNLS clients, to insert, lookup, and remove location records consisting of several keys, using these keys as handles. The GNLS service is implemented on the DHT chord infrastructure providing operations whereby given a key, it maps the key onto a node on an identifier network. Data storage and location can be easily implemented on top of Chord infrastructure by associating a key with each data item, and storing the key/data item pair at the node at which the key maps.

2.1 GNLS Device Identification Keys

The GNLS system works with network identifications / locations and treats them as a byte sequences. These items are keys in lookup service and involve:

- Medium Access Control (MAC) address,
- Internet Protocol (IP) address,
- Domain Name (DNS),
- Electronic Numbering (ENUM),
- GSM IMEI number, and
- GPS position.

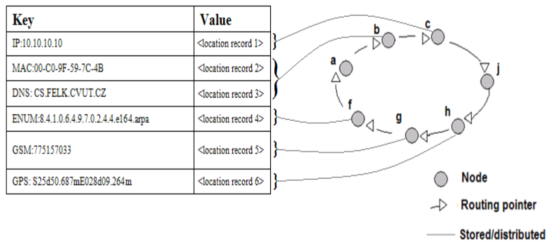
The list can be completed by including others, not so widely identification mechanisms.

Table 1: Location record with 6 name keys and contents

Key Field	Value
IP:10.10.10.10	{IP,MAC,DNS,ENUM,GSM,GPS}
MAC:00-C0-9F-59-7C-4B	{IP,MAC,DNS,ENUM,GSM,GPS}
DNS: CS.FELK.CVUT.CZ	{IP,MAC,DNS,ENUM,GSM,GPS}
ENUM:8.4.1.0.6.4.9.7.0.2.4.4.e164.arpa	{IP,MAC,DNS,ENUM,GSM,GPS}
GSM:284011234567890	{IP,MAC,DNS,ENUM,GSM,GPS}
GPS: S25d50.687mE028d09.264m	{IP,MAC,DNS,ENUM,GSM,GPS}

The location records are data items stored in the GNLS system that consist of the keys identifying network devices: Other information can be added to location records, keys of additional identification / location systems, authentication information, etc. The basic idea of the Generic Network

Location Service (GNLS) is storing location records, which are containing different identifications of the network device instead of purely unstructured data. The location records contain names uniquely identifying network devices mentioned: MAC addresses, IP addresses, GPS waypoint coordinates DNS names, ENUM numbers, SIP names, GSM IMEI numbers, etc. Other information can be added to location records, for example keys for authentication, which should be a vital component of any practical implementation of the location service (the issue of corresponding security protocols is not further discussed in this material). Figure 2 illustrates the logical concept of storing location records in Chord DHT.



Key	Value
IP:10.10.10.10	<location record 1>
MAC:00-C0-9F-59-7C-4B	<location record 2>
DNS: CS.FELK.CVUT.CZ	<location record 3>
ENUM:8.4.1.0.6.4.9.7.0.2.4.4.e164.arpa	<location record 4>
GSM:775157033	<location record 5>
GPS: S25d50.687mE028d09.264m	<location record 6>

Figure 2: Distribution of Location Records within a Chord DHT.

2.2 GNLS Resilience to Node Failures

The GNLS system decreases sensitivity to failures of its nodes by replication of individual location records. Every location record containing n location record keys is stored in n copies in GNLS. Hash function distributes location records evenly over the id space and the only problem of this distribution can be uneven division of the id space among DHT nodes, which results in uneven number of records stored in individual DHT nodes. Although distribution of location records among more nodes itself decreases negative effects of concentration in classical systems (e.g. failure of a DNS server and its secondary replica can disconnect the service area), we need mechanism able to recover from damages resulting from individual node failures. Though more copies of location records are stored in the GNLS system there will be low probable situations when a failure of a single node may result in loss of information, these techniques are improving the situation.

In the following paragraphs, two basic types of such data resilience mechanisms will be described: the pro-active and the reactive one. Though more copies of location records are stored in the GNLS system there will be low probable situations when a failure of a single node may result in loss of information, these techniques are improving the situation.

2.3 Proactive Strategy

The simplest mechanism able to assure high availability of location records in Chord DHT system is to replicate them at some node, which takes on an original node's function after the failure. It corresponds to standard data replication in Chord; where every data record located to the node has a backup at the successor of the node. When a failure of the node is detected, information on the fact that the failing node will be replaced by its successor is broadcast among DHT nodes using the finger tables' tree. Stabilization of finger tables finishes in O(log2n) steps and all data originally stored at the failing node are made available at the backup location.

Replication of data (i.e. location records in the GNLS system) to the next node, which follows, does not influence reliability of the systems assuming the mean time between node failures is much longer then the time required to copy all location records. Advantage of the proactive replication strategy is: all location records are available immediately after the DHT structures (successor links and finger tables) stabilize.

2.4 Reactive Strategy

A drawback of the pro-active strategy is that it almost doubles (for higher number of nodes) memory requirements. The simplest mechanism that exploits multiplicity of location records in GNLS reacts to the detected node failure as follows: Together with starting a stabilization mechanism (i.e. update of node links and finger tables) lookup request is broadcast using the finger tables' tree. For a single mode failure, the worst case analysis shows that in at most O(2.log2n) steps all replicas located out of the failing node will be found. That means, the lookup request can return the result, and, since the failing node is identified, lost replicas can be reproduced at the failing node surrogate. The advantage of the reactive strategy to replication is that it benefits from essence of location records: multiple copies distributed among more nodes, i.e. the mechanism needs no extra memory.

The risk of placing all copies to a single failing node is acceptably low for higher number of DHT nodes. Moreover, such a situation can be simply detected and the additional copy of the location record can be created elsewhere, e.g. at the node's successor. Any node may be used as a place for such an additional copy, linkage of this backup copy (found by lookup broadcast in the case of failure detected during regular lookup) to the true location record can be maintained without difficulties. The influence of multiple failures is lower than in proactive methods assuming the number of fields in location records is greater then number of copies in proactive methods (including the original). We therefore implemented the reactive strategy by incorporating algorithms capable of finding location record replicas and generating the lost replicas. The algorithms include In-depth Broadcast, Sequential clockwise, Sequential Anti-clockwise and Parallel Clockwise and Anti-Clockwise. These algorithms are explained in section 2.5.

3. REACTIVE STRATERGY ALGORITHMS

Reactive algorithms react to the detected node failure as follows: Together with starting stabilization mechanism (i.e. an update of node links and finger tables) lookup request is broadcast using the finger tables' tree. For a single node failure, the worst case analysis shows that in at most the broadcast algorithm will make the worst case time and message complexity logk(n), where k is a configurable constant. Broadcast algorithm requires that nodes use their finger tables as routing tables of size (k − 1) logk(n). Where k refers to the base of the system. O(2.logN) steps all replicas located out of the failing node will be found. That means, the lookup request can return the result, and, since the failing node is identified, lost replicas can be reproduced at the failing node surrogate. The advantage of the reactive strategy to replication is that it benefits from essence of location records: multiple copies distributed among more nodes, i.e. the mechanism needs no extra memory.

The algorithms have been implemented using PlanetSim open source framework as a test bed. PlanetSim[18] implements standard Chord DHT lookup algorithm herein called ROUTE algorithm. Route algorithms utilize the Chord DHT's nodes finger tables to route messages and perform lookups. The implemented lookup mechanism in GNLS works as shown in the figure 3 below.

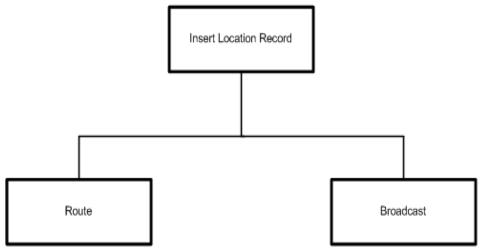
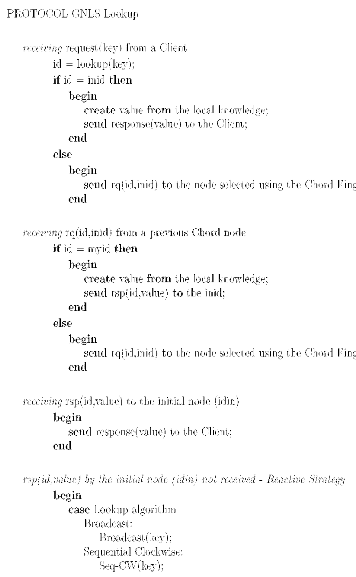


Figure 3: GNLS Location record Insert Algorithms

Figure 4: GNLS Protocol

3.1 Algorithms Complexity



```
PROTOCOL GNLS Lookup

  receiving request(key) from a Client
      id = lookup(key);
      if id = inid then
        begin
          create value from the local knowledge;
          send response(value) to the Client;
        end
      else
        begin
          send rq(id,inid) to the node selected using the Chord Finger Table;
        end

  receiving rq(id,inid) from a previous Chord node
      if id = myid then
        begin
          create value from the local knowledge;
          send rsp(id,value) to the inid;
        end
      else
        begin
          send rq(id,inid) to the node selected using the Chord Finger Table;
        end

  receiving rsp(id,value) to the initial node (idin)
      begin
        send response(value) to the Client;
      end

  rsp(id,value) by the initial node (idin) not received - Reactive Strategy
      begin
        case Lookup algorithm
          Broadcast:
            Broadcast(key);
          Sequential Clockwise:
            Seq-CW(key);
```

The efficiency of our GNLS algorithms will be measured in terms of resource consumption and time consumption. We assume that local node computations use negligible resources and take negligible time compared to the overhead of message passing.

We use message complexity as a measure of resource consumption. The message complexity of an algorithm is the total number of messages exchanged by the algorithm. Sometimes, the message complexity does not convey the real communication overhead of an algorithm, as the size of the messages is not taken into account. Therefore in some instances, we use bit complexity to measure the total number of bits used in the messages by some algorithm.

Time complexity will be used to measure the time consumption of an algorithm. In our experiments, we assume simulation time and that the transmission time takes at most one time unit and all other operations take zero time units. The worst case time complexity is often the same if we assume that the transmission of a message takes exactly one time unit, but for some algorithms the worst time complexity increases if we assume that the time it takes to send a message takes at most one time unit. Unless specified, we assume that our complexity measures denote the worst-case complexity of a given algorithm

3.2 In-Depth Broadcast Lookup

This method is invoked once the first Chord DHT lookup(key) returns a negative response meaning that the initial node responsible for storing this record has failed. When a node fails or crashes the data kept on it is lost and not available to the system. In this case mechanism that exploits multiplicity of location records in GNLS reacts to the detected node failure as follows: Together with starting stabilization mechanism (i.e. an update of node links and finger tables) In-Depth-Lookup request is broadcast using the finger tables' tree. The broadcast algorithm guarantees that all nodes in the network will be contacted. In a system of N nodes, a broadcast message originating at an arbitrary node reaches all other nodes after exactly N-1 messages.

Once a node receives a broadcast message it checks its local storage of location records for the key (for performance reasons location records will be kept in the nodes local cache). Upon receipt of the broadcast message containing the key, the node will perform a local search into location records kept in its storage. The search will parse all location record's key fields and if the field key is found the location record will be fetched and routed back to the node initiating the broadcast.

In this case the node will terminate the broadcast since the specific location record has been found. If the key is not found amongst the location records available at the node, the node will simply forward the broadcast message to the nodes it is responsible for. The time taken to receive a positive response by the query initiating node from the node containing the key in the network can be represented as DLt = Bt + Ls + Rt . Where DLt is the total response time and Bt is the time the broadcast message arrive at a node; Ls is the time for local cache lookup (dependent on CPU capabilities) and Rt is the time reply containing the location record takes to reach the broadcast initiating or querying node.

The figure 5 below illustrates a high level In-Depth-lookup algorithm utilizing broadcast mechanism implemented in PlanetSim.

```
  In-Depth-lookup(Key)
  --------------------------------------------
------
  In-depth-lookup(key), SERVER SIDE
  --------------------------------------------
------
  Hashtable data= key-record pairs;
  Structure response {
     boolean isNegative := true;
     Record record;
  }
  Structure deepLookup-message {
     String key;
      Node-id origin;
  }
  Structure response-message {
     Node-id target;
     Record record;
  }
  /*In order to run the in-depth-lookup, we have
to search through all records*/
  /*stored in the Chord DHT "data" and search
each of them for the "key" the*/
  /*client is looking for.*/
  Record recordFound := null;
  Set allRecords := data.allRecords;
  /*Go through the set of all records (values)
stored in the hashtable...*/
  for each element (Record) from allRecords do {
      /*Check whether the one of the pairs in
this record*/
      /*contains the "key" the client is looking
for.*/
      if(element.hasKey(key)) {
            recordFound := element;
        }
    }
    /*If the key has been found at the server,
there is no need for broadcast...*/
    if(recordFound            !=            null{
response.isNegative := false;
        response.record := recordFound;
    } else {
    /*If the key has not been found at the server,
there is deep-lookup message creating*/
    /*and broadcasting going on...*/
      message := deepLookup-message;
      message.key := key;
      message.origin := This server;
      broadcast(message);
    }
    responseDelivered(response-message) {
        /*The difference here is that only the
first response is being stored.*/
        /*There will be (might be) more positive
responses for in-depth-lookup from*/
        /*the network, but only the first one is
accepted and being offered to the*/
        /*client*/
    if(response.isNegative) {
                        response.isNegative      =
false;
                        response.record          =
response-message.record;
            }
    }
```

Figure 5: GNLS In-Depth Broadcast Algorithm

3.2.1 Explanatory Example

Step 1: We setup a 30 node Chord DHT network and use to test out our algorithm. The network topology looks like this:

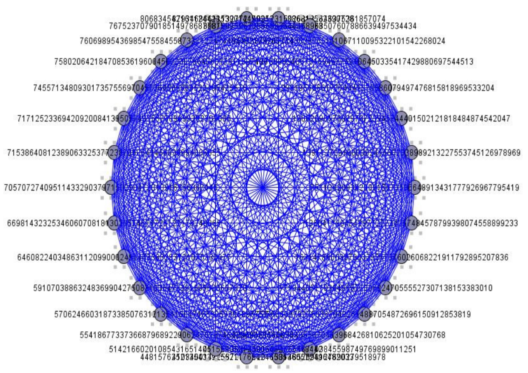


Figure 6: Show the Chord network with 30 nodes

Step 2: We now insert the following location record in the network.

Table 2: Location Record details

LR	Key Field	Location Record Fields
LR C1	IP:10.10.10.10	{IP,MAC,DNS,ENUM,GSM,GPS}
LR C2	MAC:00-C0-9F-59-7C-4B	{IP,MAC,DNS,ENUM,GSM,GPS}
LR C3	DNS: CS.FELK.CVUT.CZ	{IP,MAC,DNS,ENUM,GSM,GPS}
LR C4	ENUM:8.4.1.0.6.4.9.7.0.2.4.4.e164.arpa	{IP,MAC,DNS,ENUM,GSM,GPS}
LR C5	GSM:284011234567890	{IP,MAC,DNS,ENUM,GSM,GPS}
LR C6	GPS: S25d50.687mE028d09.264m	{IP,MAC,DNS,ENUM,GSM,GPS}

The distribution of the location records after insertion is random and depending on the value hashed from the key, the node-id is assigned within the network. The following table 3 illustrates the location record mappings in the DHT network.

Table 3: Distribution of Location records in the 30 node DHT, node 10 is connected to the client

No de No	GNLS Node-id	Key Field
1	26074449377374311736746056743401592241930795503	
2	140738831925442692888321933566609892072254138213	{CS.FELK.CVUT.CZ=[Record 1]}
3	191520614644224828506285644944016166346563282092	
4	198966107340074132805079795960298847766836805652	
5	232219961150593515806704296440369611434548112127	
6	289164740515386390273826447894430660906960853992	
7	316546629438871396714888441082868311468478049425	
8	417780411363740541675836778413245836649507509505	
9	447079944339272582040320945437386070578709058487	
10	518334718266324065298274479769091652668412410465	Node connected to client
11	548512961911599395304697235126315157440886275266	
12	562828613957046539590971840574583014407191388741	
13	590442188132729364486709285384363838639343250578	
14	693967442967512823428669242183336583451760024349	{S25d50.687mE028d09.264m=[Record 1], 10.10.10.10=[Record 1]}
15	729527615060373863377749762536798615361141909151	
16	751206505167380566682189794975958260056024584661	{284011234567890=[Record 1]}
17	836892181796186454640091713881945825448880078502	
18	867774298383195349984194822568125679207212410600	
19	901708681365647825767925092782996879063326244788	
20	906228602344310208008598000184396542598106541871	
21	989735446091124374134144687086495329776759953738	{8.4.1.0.6.4.9.7.0.2.4.4.e164.arpa=[Record 1]}
22	1030194773237856784868426082	

	114933644632370237770	
23	1077670814585334983765854593 667932205423091329464	
24	1187915666977838046787973373 250250412465428483356	
25	1237847457405753999838095309 047746699814413062503	
26	1240965994435693295994585148 152570644146244355490	
27	1338364438744683379744787640 004055126666512479186	{00-C0-9F-59-7C-4B=[Record 1]}
28	1403912127989954990673523333 925115118194073599473	
29	1421846846594109130829302483 810298808451814914875	
30	1436873910024584759551466932 334606598576859280062	

Step 3: We perform lookups using the standard Chord route mechanisms and obtain the number of messages exchanged and the measure the timestamps in simulation milliseconds. Thereafter we fail the nodes containing the location records 2,14,16,21 and 27 and perform lookup on each respective location record key.

- When we fail a node containing a location record and perform a lookup, the following will happen.
- Route lookup will fail on the key and return a negative response.
- Broadcast deep lookup will be triggered
- The Broadcast algorithm will use finger tables and contact all the nodes in the DHT network (n-1)

When the node is contacted, it will search through its local cache of location records matching the key. If the key is matched, it will send a response to the querying node with the location record details. If the key does not match any of the fields of the location records in the cache, the node will simply pass on the broadcast message to the next node according to the finger tables

Once the querying node receives the first positive response (record found) it will deliver the record and discard any other positive responses there on. The location record will be reconstructed with the same key within the DHT network on a surrogate node

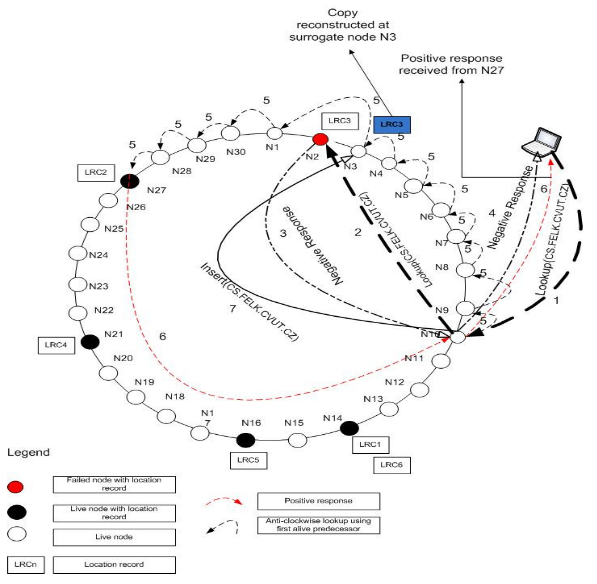

Copy reconstructed at surrogate node N3
Positive response received from N27
Lookup(CS.FELK.CVUT.CZ)
Negative Response
Negative Response
Insert(CS.FELK.CVUT.CZ)
LRC1
LRC2
LRC3
LRC4
LRC5
LRC6
Legend
Failed node with location record
Live node with location record
Live node
Location record
Positive response
Anti-clockwise lookup using first alive predecessor

Figure 7: Deep lookup of location record with key CS.FELK.CVUT.CZ after failure of original node 2.

Step 4: We repeat the experiment for each of the remaining 5 keys i.e. failing the node and performing a lookup on the respective key. The following results were obtained:

Table 4: Simulation results of lookups on location record keys

Algorithm	Unit	IP	GPS	MAC	ENUM	GSM	DNS	Average
Route	msg	23	23	23	29	29	32	26.5
	Msec	15	15	15	15	16	16	15.333
In-Depth Broadcast	msg	32	47	31	51	62	74	49.5
	Msec	125	32	32	46	219	203	109.5

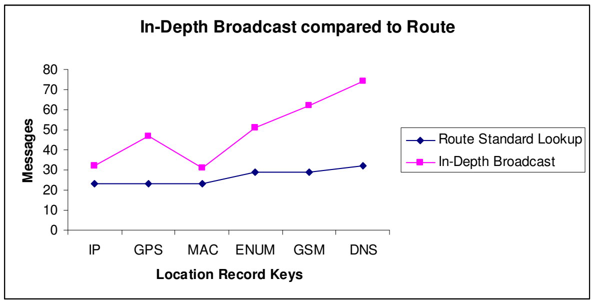


Figure 8: Messages exchanged during location record lookup in a 30 node DHT network

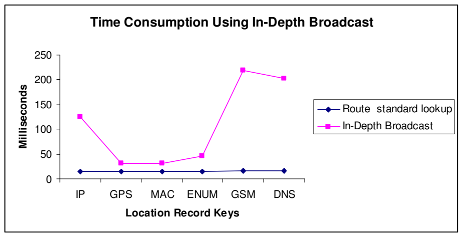


Figure 9: Time utilized by the In-Depth Broadcast Algorithm

3.2.2 Complexity Analysis of In-Depth Broadcast Lookup

In-Depth broadcast algorithm reacts to the detected node failure as follows: Together with starting stabilization mechanism (i.e. an update of node links and finger tables) lookup request is broadcast using the finger tables' tree. For a single node failure, the worst case analysis shows that in at most the broadcast algorithm will make the worst case time and message complexity logk(n), where k is a configurable constant. Broadcast algorithm requires that nodes use their finger tables as routing tables of size (k – 1) logk(n). Where k refers to the base of the system. O(2.logN).

4. SEQUENTIAL CLOCK-WISE LOOKUP ALGORITHM

Lookup is initiated by the node using the first alive successor on the Chord DHT ring reaching all nodes in the network in O(n), where n is the number of nodes in the network. The direction of the lookup is in a clock-wise direction until all the nodes in the ring are contacted. When the node is contacted, its first looks up in its local cache before forwarding the lookup query to the next alive successor. The node's local cache will contain one or more location records from various devices in the network. The node will scan all local records and try to match the key from the lookup query. If the key being queried is found, the node will generate a response and forward it to the query initiator node, otherwise the node will forward the query to the next alive successor on the ring clockwise. If the query reaches the node which initiated the query then this results in

negative response, meaning that the location record with specified key does not exist in the Chord DHT ring network. The pseudo code algorithm is presented below:

4.1 Experimental Explanatory Example:

Similarly we perform simulation of Sequential algorithm using the same DHT network as in section 2.5. The following results were obtained.

Table 5: Simulation results lookup using Sequential clockwise Algorithm

Algorithm	Unit	IP	GPS	MAC	ENUM	GSM	DNS	Average
Route	msg	23	23	23	29	29	32	26.5
	Msec	15	15	15	15	16	16	15.3333
Sequential clockwise	msg	23	23	25	36	36	36	29.8333
	Msec	47	16	47	63	31	31	39.1667

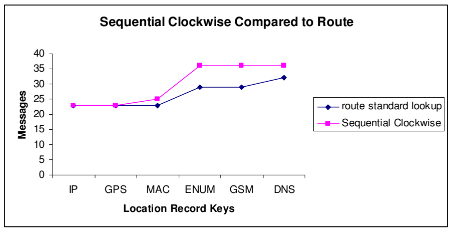


Figure 10: Messages exchanged during location record lookup in a 30 node DHT network

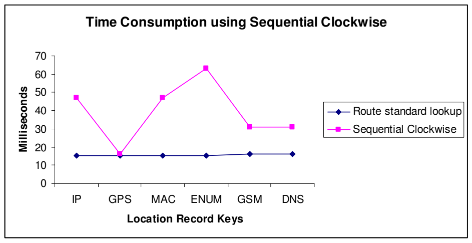


Figure 11: Time utilized by the Sequential Clockwise Algorithm

4.2 Complexity Analysis of Sequential Clock-wise Lookup Algorithm

To analyze the complexity of the proposed Sequential Clock-wise Lookup Algorithm, we consider a Chord DHT with all live nodes in the Network. For simplicity, the complexity analysis will be considered based on the Chord DHT below in Figure 12:

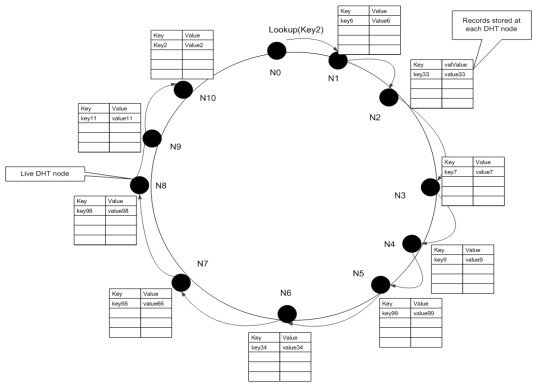


Figure 12: Sequential Clock-wise lookup and distribution of records in a Chord DHT with ten live nodes.

The ring topology has poor performance in terms of worst case message complexity and time complexity. The worst case time complexity and message complexity are n for the ring topology, because in the worst case all of the ring nodes need to be traversed, as depicted in figure 12 above, lookup initiated at node N0 will traverse the ring 10 times. Our extension will make the worst case time and message complexity O(n) , where n is the number of nodes in the system

4.3 Sequential Anti-Clockwise Lookup Algorithm

Lookup is initiated by the node using the first alive predecessor on the ring network reaching all nodes in O(n), The direction of the lookup is in an anti-clock-wise direction until all the nodes in the ring are contacted. When the node is contacted, its first looks up in its local cache before forwarding the lookup query to the next alive successor. The node's local cache will contain one or more location records from various devices in the network. The node will scan all local records and try to match the key from the lookup query. If the key being queried is found, the node will generate a response and forward it to the query initiator node, otherwise the node will forward the query to the next alive successor on the ring anti-clockwise. If the query reaches the node which initiated the query then this results in negative response, meaning that the location record with specified key does not exist in the Chord DHT ring network. The pseudo code algorithm is presented below

4.3.1 Explanatory Example

Similarly we perform simulation of Sequential Anti-clockwise algorithm using the same DHT network as in section 2.5. The following results were obtained.

Table 6: Simulation results lookup using Sequential Anti-clockwise Algorithm

Algorithm	Unit	IP	GPS	MAC	ENUM	GSM	DNS	Average
Route	msg	23	23	23	29	29	32	26.5
	Msec	15	15	15	15	16	16	15.33333
Sequential Anti-clockwise	msg	23	23	25	23	23	23	23.333333
	Msec	78	47	46	47	46	62	54.333333

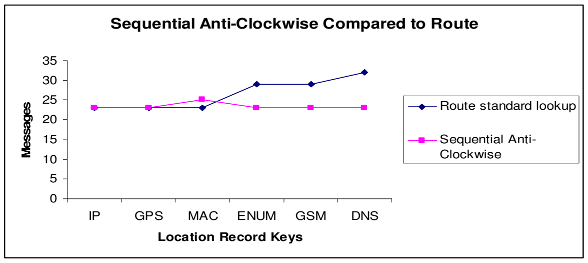


Figure 13: Messages exchanged during location record lookup in a 30 node DHT network

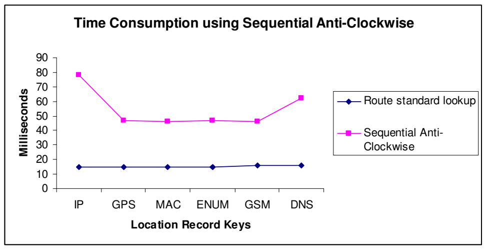


Figure 14: Lookup time utilized by the Sequential Anti-Clockwise Algorithm

4.4 Complexity Analysis of Sequential Anti-Clockwise Lookup Algorithm

This algorithm will yield the same worst case complexity for message and time complexity as stated in the Sequential Clockwise Lookup Algorithm in 4.8.1.1 The worst case time complexity and message complexity are O(n) for the ring topology, where n is the number of nodes in the system.

4.5 Parallel Clockwise and Anti-Clockwise Lookup Algorithm

Lookup is initiated by the node using the first alive predecessor on the ring network reaching all nodes in O(n),), Simultaneous Sequential clock-wise and anti-clockwise SEQUENTIAL-B, here both sequential clockwise and anti-clock are initiated at the same time reaching all nodes in the network in O(n/2) and lastly but not least the Broadcast algorithm. Illustration of the simplest mechanism that exploits multiplicity of location records in Generic Network Location Service (GNLS).

4.5.1 Explanatory Example

Similarly we perform simulation of Parallel Sequential Anti-clockwise algorithm using the same DHT network as in section 2.5. The following results were obtained.

Table 7: Simulation results Lookup using Parallel Clockwise and Anti-clockwise Algorithm

Algorithm	Unit	IP	GPS	MAC	ENUM	GSM	DNS	Average
Route	msg	23	23	23	29	29	32	26.5
	Msec	15	15	15	15	16	16	15.3333
Parallel Sequential -B	msg	24	24	26	34	34	34	29.333
	Msec	125	47	110	125	109	141	109.5

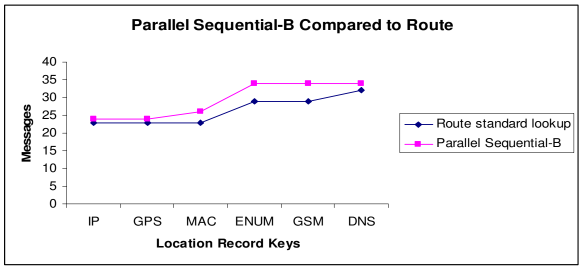


Figure 15: Messages exchanged during location record lookup in a 30 node DHT network

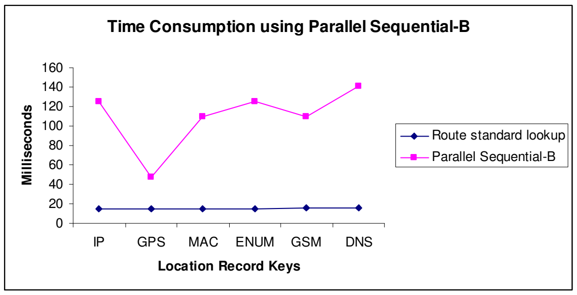


Figure 16: Lookup time utilized by the Sequential Anti-Clockwise Algorithm

4.6 Complexity Analysis of Parallel Clockwise and Anti-Clockwise Lookup Algorithm

If the search can go in both clockwise and anti-clockwise direction, half of the nodes in the ring need to be traversed. In this algorithm both sequential clockwise and anti-clock are initiated at the same time reaching all nodes in the network in O(n/2).

4.7 Comparative Analysis of Reactive Algorithms

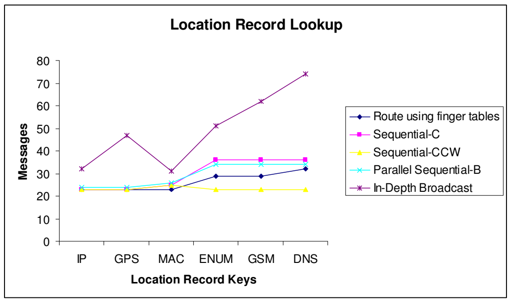


Figure 17: Comparative analysis of messages exchanged during key lookup in a 30 node DHT network.

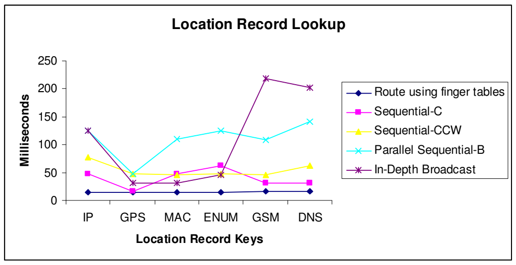


Figure 18: Comparative analysis of messages exchanged during key lookup in a 30 node DHT network.

5. CONCLUSION

This Paper presented five algorithms implemented in GNLS system including their performance and complexity analysis. The efficiency of our GNLS algorithms was measured in terms of resource consumption and time consumption. We assumed that local node computations use negligible resources and take negligible time compared to the overhead of message passing.

The performance of the proposed reactive algorithms was examined in comparison with the standard Chord Route lookup algorithm implemented in PlanetSim simulation framework. The algorithms examined were the Sequential Clockwise, Sequential Anti-Clockwise, Parallel Sequential and the In-Depth Broadcast. The performance of the algorithms was measured in respect of varying network sizes, messages exchanged and time consumed. Mainly the proposed algorithms performed reasonably well for network sizes up to 2000 nodes

A Practical modification to the real network would be to create a live node list configuration file which would keep an updated list of nodes alive in the network. Depending on the number of live nodes a suitable algorithm would then be triggered in case of nodes failure detection.

The generic network location service implemented in this work is not supposed to be a substitution of the existing translation techniques (e.g. ARP, DNS, ENUM), but it is considered as an overlay that uses data available in existing systems and provides some translations currently unavailable.

REFERENCES

[1]. Bryan, D.A., Lowekamp B.B., Jennings C.: SOSIMPLE: A Serverless, Standards based, P2P SIP Communication System. In: Advanced Architectures and Algorithms for Internet Delivery and Applications, 2005, AAAIDEA 2005, 42-49.

[2]. Bryan D.A., Zangrilli M., Lowekamp B.B.: Challenges of DHT Design for a Public Communications System. College of William & Mary Computer Science Department Technical Report WMCS200603, 2006.

[3]. FIPS 1801. Secure Hash Standard. U.S. Department of Commerce/NIST, National Technical Information Service, Springfield, VA, Apr. 1995.

[4]. Cox R., Muthitacharoen A., Morris R.: Serving DNS Using a Peer-to-Peer Lookup Service In: Revised Papers from the First International Workshop on Peer-to-Peer Systems. LNCS 2429, Springer 2002, 155-165.

[5]. Yusuke Doi: DNS Meets DHT: Treating Massive ID Resolution Using DNS Over DHT. In: The 2005 Symposium on Applications and the Internet (SAINT'05), 9-15.

[6]. Fiedler J., Kupka T., Magedanz T, Kleis M.: Reliable VoIP Services Using a Peer-to-Peer Intranet. In: Eighth IEEE International Symposium on Multimedia (ISM'06), Dec. 2006, 121-130.

[7]. Balakrishnan H., Kaashoek M.F., Karger D., Morris R., Stoica I.: Looking up Data in P2P systems. In: Technical and social components of peer-to-peer computing, 2003, 43-48.

[8]. Huitema Ch., Miller J. L.: Peer-to-peer name resolution protocol (pnrp) and multilevel cache for use therewith. EP1248441 Patent, Microsoft 2002.

[9]. RFC 3761: The E.164 to Uniform Resource Identifiers (URI) Dynamic Delegation Discovery System (DDDS) Application (ENUM).

[10]. Lewin, D.: Consistent hashing and random trees: Algorithms for caching in distributed networks. MSc. thesis, MIT, 1998.

[11]. Stoica I., Morris R., Karger D., Kaashoek M.F., Balakrishnan H.: Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications. In: Proceedings of the 2001 ACM SIGCOMM Conference. 2001.

[12]. Quanhao Lin, Ruonan Rao, Minglu Li.: DWSDM: A Web Services Discovery Mechanism Based on a Distributed Hash Table. In: Fifth International Conference on Grid and Cooperative Computing Workshops, Oct. 2006, 176-180.

[13]. Mwansa L., Janeček J.: P2P: Generic Network Location Service for Web Applications. In: 9th Annual Conference on World Wide Web Applications [CDROM], Cape Town Cape Peninsula University of Technology, 2007, ISBN 9780620398374, 1-16.

[14]. Ratnasamy S., Francis P., Handley M., Karp R., Shenker S.: A scalable content addressable network. In: Proc. ACM SIGCOMM, San Diego, CA, August 2001, 161-172.

[15]. Risson J., Qazi S., Moors T., Harwood A.: A Dependable Global Location Service using Rendezvous on Hierarchic Distributed Hash Tables. In: International Conference on Networking. In: International Conference on Systems and International Conference on Mobile Communications and Learning Technologies (ICNICONSMCL'06), April 2006, 4.

[16]. Rowstron, A., Drushel.,P.: Pastry: Scalable, distributed object location and routing for large scale peer-to-peer systems. In: Proceedings of the 18th IFIP/ACM International conference on distributed systems platforms (Middleware 2001). Nov. 2001.

[17]. Singh K., Schulzrinne H.: Using an External DHT as a SIP Location Service. Columbia University Technical Report CUCS00706, New York, NY, Feb 2006.

[18]. PlanetSim, http://projects-deim.urv.cat/trac/planetsim/