



PROCEEDINGS OF THE
11th ANNUAL CONFERENCE
ON WORLD WIDE WEB APPLICATIONS

2-4 September 2009
Port Elizabeth
South Africa

Editor:
P.A. van Brakel

Publisher:
Cape Peninsula University of Technology
PO Box 652
Cape Town
8000

Proceedings published at
<http://www.zaw3.co.za>

ISBN: 978-0-620-45215-1

TO WHOM IT MAY CONCERN

The full papers were refereed by a double-blind reviewing process according to South Africa's Department of Education (DoE) refereeing standards. Papers were reviewed according to the following criteria:

- Relevancy of the subject to Web applications
- Explanation of the research problem & investigative questions
- Quality of the literature analysis
- Appropriateness of the research method(s)
- Adequacy of the evidence (findings) presented in the paper
- Standardised referencing style.

The following reviewers took part in the process of evaluating the full papers of the 11th Annual Conference on World Wide Web Applications, held from 2 tot 4 September 2009 in Port Elizabeth:

Prof J Brits
Faculty of Information Sciences (Dean)
University of Wisconsin
Milwaukee
USA

Prof T Carr
Centre for Higher Education Development
University of Cape Town
Cape Town

Prof J Cronjé
Faculty of Informatics and Design (Dean)
Cape Peninsula University of Technology
Cape Town

Mr Craig de Beer
Web Management Services
Massey University
Palmerstone
New Zealand

Prof M Erasmus
Management Information Systems
Erasmus Associates
Lynnwood
Pretoria

Dr AM El-Sobky
Educational Sciences
RITSEC
Cairo
Egypt

Dr MWH Labour
Laboratory of the Sciences of Communication
University of Valenciennes and Hainaut Cambrésis
Valenciennes Cedex
France

Prof S Mutula
Department of Information Science
University of Botswana
Botswana

Dr David Raitt
European Space Agency
The Hague
The Netherlands

Prof A Singh
Department of Business Information Systems
University of KwaZulu-Natal
Durban

Prof P Weimann
Economics and Social Sciences
University of Applied Sciences
Berlin
Germany

Further enquiries:

Prof PA van Brakel
Conference Chair: Annual Conference on WWW Applications
Cape Peninsula University of Technology
Cape Town
+27 21 469 1015 (landline)
+27 82 966 0789 (mobile)

Analyzing DHT broadcast algorithm in location record lookup for web applications

L. Mwansa
Czech Technical University
Prague, Czech Republic
lmmwansa@yahoo.com

J. Janeček
Czech Technical University
Prague, Czech Republic
janecek@cs.felk.cvut.cz

Abstract

Our paper describes an alternative approach to address translations based on DHT usage for Web Application. Identification record is formed of all identifications that the network device has, and all these identifications are used to create keys for storing the record to a DHT. As a result, the record is stored in more copies and each copy provides access to others, allowing us to create efficient update and recovery mechanisms. Storing collections of network identifications has some advantages: straightforward implementation of all possible translations at the same time (provides a way to location services which are unavailable in current systems), and fault-tolerance to DHT's node and network failures due to inherent replication of identification records. The main contribution of this paper entails the illustration of the simplest mechanism that exploits multiplicity of location records in Generic Network Location Service (GNLS). It 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 $O(2 \cdot \log N)$ 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. This service can be exploited by various Web applications designers implementing overlay networks running on top of the existing Internet network topology including lookup services embedded in search engines.

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

1. Introduction

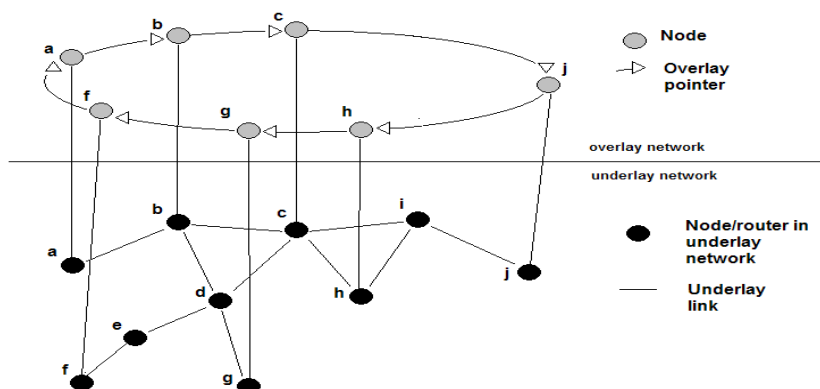
In this section we will briefly describe the Generic Network Location Service (GNLS) according to [13]. The GNLS system consists of the servers, GNLS nodes, which provide a distributed service that allows network devices, GNLS clients, to insert, lookup, and remove location records consisting of several names, using these names as name keys. 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.

A DHT infrastructure is 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 Stoica (2003:1) 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 $O(\log N)$ time, using a finger tables of $\log N$ 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 IDspace distance to k . As a result, Chord lookups use $O(\log N)$ messages.

Chord ensures correct lookups despite of 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. 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 $\log N$) make the probability of simultaneous failure vanishingly small.

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 (GHODSI (2006:2))

Figure1. Physical underlay network with 10 nodes and an overlay network formed of 7 nodes



1.1 Related work

In this section we will briefly describe the Generic Network Location Service (GNLS) for web application according to Mwansa(2007a,2007:1-16), 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 names, using these names as name keys.

1.2 Name keys

The GNLS system works with names that are representing network identifications / locations and treats them as byte sequences. These items are converted to DHT keys for the lookup, the most frequently used network identifications /locations are:

- 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.

Figure 2: Location record with 6 name keys and contents

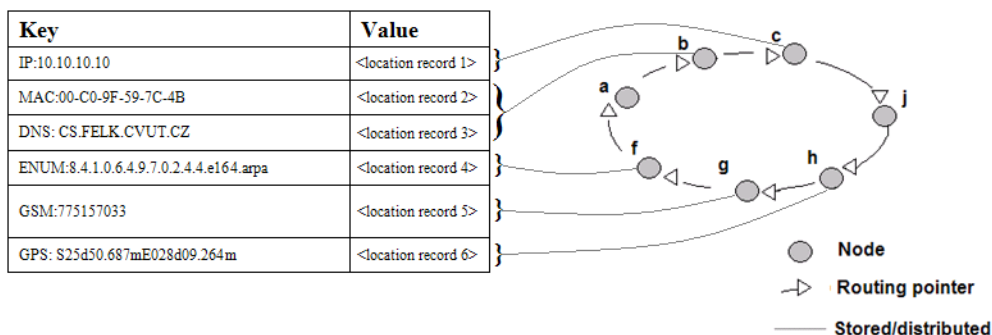
Key Field	Value
IP:10.10.10.10	{IP,MAC,DNS,ENUM,GSM,GPS}
MAC:00-C0-9F-59-7C-4B	{IP,MAC,DNS,ENUM,GSM,GPS}
DNS: CS.FELK.CVUT.CZ	{IP,MAC,DNS,ENUM,GSM,GPS}
ENUM:8.4.1.0.6.4.9.7.0.2.4.4.e164.arpa	{IP,MAC,DNS,ENUM,GSM,GPS}
GSM:284011234567890	{IP,MAC,DNS,ENUM,GSM,GPS}
GPS: S25d50.687mE028d09.264m	{IP,MAC,DNS,ENUM,GSM,GPS}

The remainder of this paper is organized as follows: section 2 describes location records mappings into the Chord DHT network as well as memory requirements. We also show the probability computations. In section 3, we present the In-depth-lookup algorithm, which presents our approach to solving the problem related to node failures and ensuring resilience. Preliminary simulations results are also presented in this section. We conclude the paper by giving a brief analysis of the benefits of the GNLS including issues of fault-tolerance and future work.

2. Location records and Collections

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 3: Distribution of Location Records within a Chord DHT.



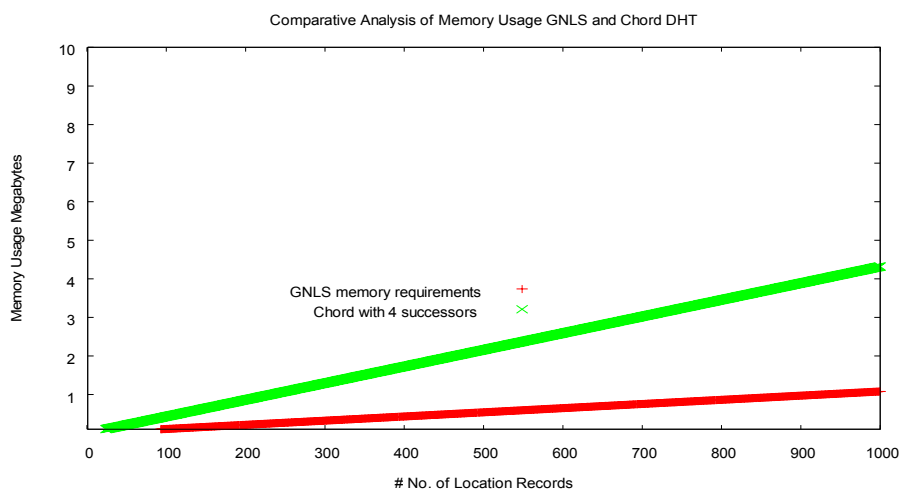
2.1 Location Record Memory Requirements

In GNLS location records are replicated according to the number of key fields present in the record. In the example above we use fixed length records containing 30 characters for each field. The record size will be computed as follows: Location record size = No. of key fields x Fixed field size. The total memory requirements for all location records in the GNLS system will be computed as follows: Total Memory = No. of location records x No. of key fields x Fixed field size.

Chord DHT has an inherent mechanism supporting resilience by storing data on the nodes successor lists Stoica (2003:7). Each node stores its successor nodes following it on the circle. Successor lists are storing key/Data pairs of the node. When a node receives an insert, it propagates a copy of the key/Data pair to those r successors; it also propagates when it notices that its immediate successor changes during stabilization. After a nodes fails, queries for its keys automatically end up at its successor, which will have copies of those keys. The successor list is configurable, but it can be defined in the range $(n + 2^{k-1}) \bmod 2^m, 1 \leq k \leq m$. Stoica (2001:2) Where m is the number of bits used to determine the Chord identifier circle. The circle can have ids/keys ranging from 0 to $2^m - 1$. In our simulations we fix the number of successors to 4 and vary the number of location records with fixed size record lengths.

The following graph give illustrates a comparative analysis of memory requirements of GNLS and classical Chord in storing location records.

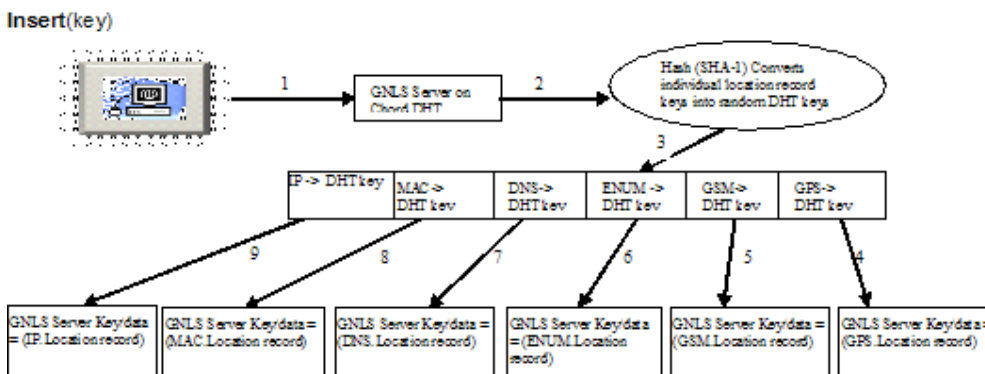
Figure 4: Comparative analysis of memory requirements between GNLS and Chord DHT



2.2 GNLS insertion of a location record in the Chord DHT nodes

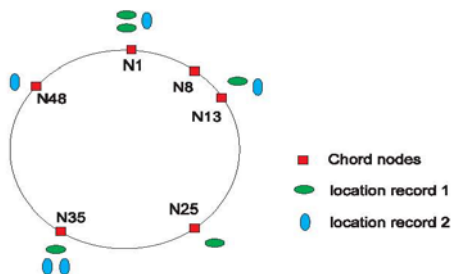
Though GNLS could provide more complex services, the basis for them is the interface to location record storage. Location records are inserted into GNLS system by the insert method that has a form: insert (location record) , where location record is a data structure containing information on types and values of involved name keys (MAC and IP addresses, DNS names, ENUM numbers, IMEI identifiers, etc.). Insert method will be illustrated with the following diagram below:

Figure 5: GNLS insertion of a location key record in the DHT nodes



The GNLS node converts individual name keys of the location record into DHT keys using the standard Chord DHT hash function SHA-1 Stoica (2001:2. Then it stores the copies of the location record in GNLS nodes designated by the Chord lookup applied to individual DHT keys as shown below:

Figure 6: GNLS insertion of a location record in the Chord DHT nodes



In terms of traffic generated by the above operations in figure 2, a single insert method will generate 7 messages excluding the local operations performed at the GNLS server (2, 3 and 10) and acknowledgements. The messages shown in figure 2 are considered only at the network overlay level. The underlying physical network layer may have different hops and latencies depending on the connectivity and protocol used. The cost of a lookup operation in standard Chord DHT is given by $O(\log N)$ Stoica (2001:1) Operation 4 to 9 can be performed in parallel since they do not technically fall in sequence, thereby enhancing performance of insert operations on several keys within the location record. The GNLS local operations time will depend on the specific CPU and memory capacity of the server.

The following algorithm highlights the insert method:

Figure 7: Location record insertion algorithm

```
insert(Location record)
```

```
insert(Location record) -> GNLS server
```

```
Get number of name keys in Location record (n)
```

```
Number of copies = n.
```

```
Do while (i <= n) /* Allocate location record copy(i) within DHT */
```

```
    Insert(key(i),location record) /* create location record copy in DHT */
```

```
    increment ++i
```

```
Endwhile
```

In our example, we have 6 name keys in the location record; the GNLS DHT node will generate 6 copies of the location record. Chord uses consistent hashing to map nodes onto an m -bit circular identifier space. Where m is 160 bit giving $2^{160} = 1.4615016373309029182036848327163e+48$. All nodes maintain an m -entry routing table called the finger table. This table stores information about other nodes in the system: each entry contains a node identifier and its network address (consisting of an IP address and port number). The diagram below shows a simplified illustration of the GNLS mapping of the location record:

Figure 8: Illustration of GNLS server generation of location records

Where key1 = IP, key2 = MAC, key3 = DNS, key4 = ENUM, key5 = GSM, key6 = GPS

Each DHT key generated is mapped to a node on the Chord DHT to store the location record. Considering that the finger tables on the Chord DHT node can be thought of in terms of d identifier intervals corresponding to d entries in the finger table. The order- k interval of a node r is defined as $[(r+2k-1) \bmod 2d, (r+2k) \bmod 2d]$, where $1 \leq k \leq d$. Nodes in a 160-bit typical Chord ring will therefore contain maximum 160 interval entries in the finger table, hence we could calculate probability that no two name keys are hashed into the same interval.

We can compute analytically the probability that two location record name keys will hash in the same interval within the DHT identifier space. Consider the probability $Q_2(n, d)$ that no two location record keys out of a set of n keys will have matching Chord DHT intervals out of d equally possible finger table intervals. (Where n = number of name keys in the location record, and d is the number of the interval entries in the finger table). We assume that the intervals are uniformly and independently distributed over the Chord DHT identifier space given by 2^{160} . If a location record has an assigned interval in m entries of the finger table, then this event has the probability of $1/dn$. Starting with an arbitrary location record's key interval, then note that the probability that the second location record key's interval is different is $(d-1)/d$, that the third location record key's interval is different from the first two is $[(d-1)/d][(d-2)/d]$, and so on, up through the n th location record key field. Explicitly,

$$Q_1(p, d) = \frac{\frac{d-1}{d} \frac{d-2}{d} \dots \frac{d-(p-1)}{d}}{(d-1)(d-2) \dots [d-(p-1)]} \quad (1)$$

$$= \frac{d!}{d^{p-1} (d-p)!} \quad (2)$$

But this can be written in terms of factorials as

$$Q_1(p, d) = \frac{d!}{(d-p)! d^{p-1}} \quad (3)$$

From our example we get $Q_1(6, 160) = 160! / (160-6)! 160^5 = 0.90951579780$

so the probability $P_2(p, d)$ that two or more location record keys out of a group of n do have the same Chord DHT interval is therefore

$$P_2(p, d) = 1 - Q_1(p, d) = 1 - \frac{d!}{(d-p)! d^{p-1}} \quad (4)$$

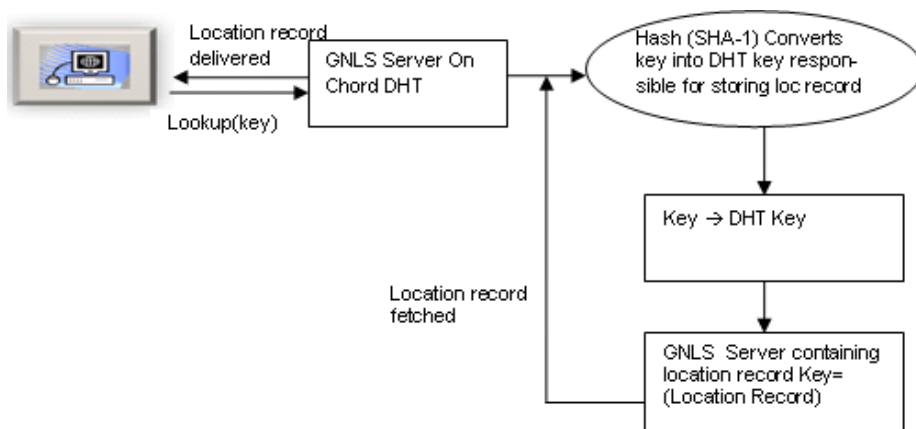
Computed from our example as $P_2 = 1 - 0.90951579780 = 0.0904842022$

A counterpart of the insert method is the lookup method that has the form $\text{lookup}(\text{key})$. The method is invoked remotely from GNLS clients as shown in figure 8 below. The GNLS server node converts the key into the DHT node id and finds the node using the Chord's lookup function, which stores the required location record. It asks DHT for the record and receiving it, the GNLS node passes the record to the GNLS client and stores it into its cache for an eventual future use.

In figure 9, a lookup operation is shown in a normal situation the operation will deliver the queried location record details. If the node containing the location record failed or crashed, the data stored on that particular node will be lost including the location records. Though DHT systems can be equipped by additional mechanisms, which are able to avoid loss of stored data, the essence of GNLS system provides additional possibilities to fault-tolerance of stored data from node failures.

The simplest 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) lookup request is broadcast using the finger tables' tree. For a single node failure, the worst case analysis shows that in at most $O(2 \cdot \log N)$ 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.

Figure 9: GNLS lookup of a location record in the DHT nodes



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.

3. Algorithm: In-Depth-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 $DL_t = B_t + L_s + R_t$, where DL_t is the total response time and B_t is the time the broadcast message arrive at a node; L_s is the time for local cache lookup (dependent on CPU capabilities) and R_t is the time reply containing the location record takes to reach the broadcast initiating or querying node.

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

Figure 10: GNLS In-depth-lookup high level algorithm

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;
  NodeID origin;
}
Structure response-message {
  NodeID 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.containsKey(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;
  }
}
}

```

3.1 Simulation Results

In this section, we present our preliminary simulation results based on the presented In-depth-lookup algorithm. We use open source PlanetSim (http://planet.urv.es/trac/planet_sim) for simulation experiments. We modified the broadcast algorithm in PlanetSim to incorporate the presented to In-depth-lookup algorithm. We are basically interested to see the number of messages exchanged to locate a location record within a network and the time required to deliver the message for networks of varying sizes.

3.2 Experimental Environment

To study the In-depth-lookup algorithm messaging cost, we create serialized Chord DHT networks of 100, 1000, 2000, 5000 and 10000 nodes with an identifier space of size 2^{32} with 4 successor lists ensuring fault-tolerance. For each network, after all the nodes join the system, we initiate an insert(location record) starting at a randomly chosen node. The location record as shown in figure 3 is inserted in the network. We perform lookup's of location records in the network at random using various keys and record messages and time taken to deliver the location record. Thereafter we obtain the node id's of the nodes storing location records and fail or crash those nodes one at a time. Once we have failed the node containing the location record, we initiate a lookup(key) process. The lookup(key) returns a negative response their by triggering the invocation of the In-depth-lookup process. We wait until the In-depth-lookup process ends and analyze the messages and time stamps created during the process. We repeat the same experiment 20 times, failing

the nodes containing the location record at random for each given network size. The results obtained are presented below:

Figure 11: Number of messages created in GNLS In-depth-lookup to successful deliver a location record from the network using In-depth-lookup algorithm.

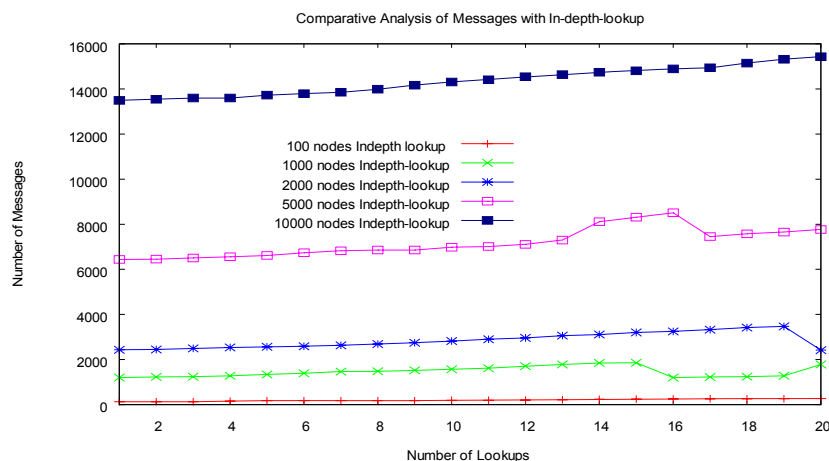
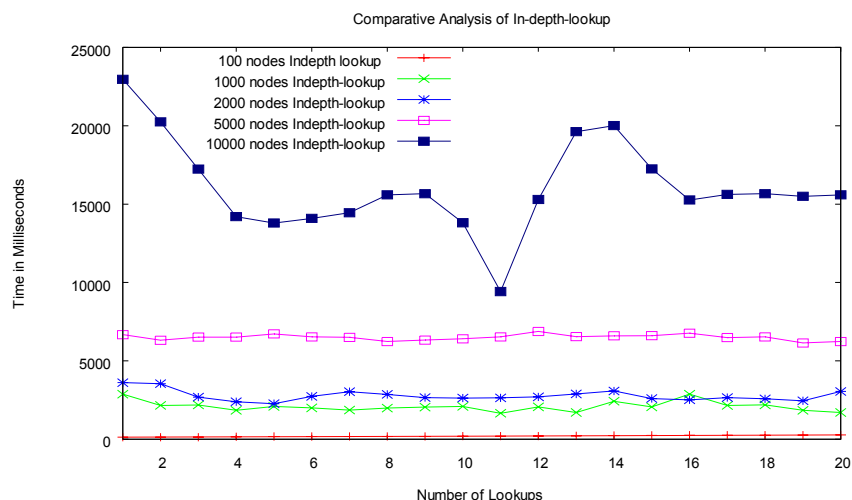


Figure 12: Time in Milliseconds required in GNLS In-depth-lookup to successful deliver a location record from the network using In-depth-lookup algorithm.



The results obtained show that the messages created by In-depth-lookup algorithm follow a consistent pattern as shown in figure 11 for varying number of lookups, the average number of messages 100 is 200.1, 1000 is 1462 and that of 10000 nodes is 14348. The messages indicate the ones created by the broadcast message returning the first positive response from the node in the network; the querying node discards other positive responses after receiving the first one. The time stamps measured in milliseconds indicate also a consistent average time of In-depth-lookup for 100 nodes of 410.1, 1000 nodes of 2091 and that of 10000 nodes is 16067 milliseconds for 20 random lookups as depicted in figure 12.

The experiments show that the location record can be found in the network despite the failure of the original node storing the record. The lost copy is created at another node surrogate. This benefit from the multiplicity of location records in the network ensures resilience to node failures. Reactive strategy employed also reduces the storage

requirements of standard Chord replication using successor lists as shown in figure 4. We are able to react to node failures by employing the second lookup using the In-depth-lookup. We acknowledge that the cost of broadcast of $N - 1$ messages in large Chord DHT networks may be prohibitive for many applications, therefore in future we intend to compare the performance of a token ring sequential lookup with the In-depth-lookup or a combination of both.

4. Conclusion

The paper presents the idea of storing location records, defined as a collection of different location identifications (e.g. MAC address, IP address, DNS name, E.164 number, GSM BTS identification) of a single device, in a DHT overlay system. Storing location records in places addressable (using the DHT lookup) by individual location record fields provides a simple way to implementation of translation functions similar to well known network services (e.g. ARP, DNS, ENUM). Storing records consisting of several location identifications in a DHT space benefits from a $O(\log N)$ complexity of the lookup. We described a reactive strategy to node failures in the network by implementing In-depth-lookup algorithm ensuring fault-tolerance to node failures. The algorithm uses broadcast with feedback benefiting from the stored replicated location records without creating additional secondary copies (up to some very infrequent cases mentioned). That means, the lookup request can return the result, and, since the failing node is identified, lost replicas are 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.

Web services and applications frequently query servers for identification information and GNLS offers a solution to designers implementing search engines. The key fields in the location record are not limited, thereby offering designers flexibility on the type of translation data they want to store in the location record. The generic network location service proposed 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 that uses data available in existing systems and provides some translations currently unavailable.

In the presence of node failures, we consider our algorithm, will delivery the location record from anywhere in the network. However in future we intend to compare In-depth-lookup algorithm with a combination of other strategies.

4.1 List of references

Ali Ghodsi, A Dissertation on Distributed k -ary System: *Algorithms for Distributed Hash Tables*. Available <http://www.sics.se/~ali/thesis/dks>.

El-Ansary, Sameh and Brand, Per and Haridi, Seif, et al. (2003) *Efficient broadcast in structured P2P networks*. In: 2nd International Workshop On Peer-To-Peer Systems (IPTPS'03), 20-21 Feb 2003, Berkeley, CA, USA. Available <http://eprints.sics.se/2811/>

Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan, *Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications*, ACM SIGCOMM 2001, San Diego, CA, August 2001, pp.149-160. Available http://pdos.csail.mit.edu/papers/chord:sigcomm01/chord_sigcomm.pdf

Ion Stoica, Robert Morris, David Liben-Nowell, David R. Karger, M. Frans Kaashoek, Frank Dabek, Hari Balakrishnan, *Chord: A Scalable Peer-to-peer Lookup Protocol for Internet Applications*. To Appear in IEEE/ACM Transactions on Networking. Available <http://pdos.csail.mit.edu/chord/papers/paper-ton.pdf>

Mwansa L., Janeček J. *Investigating generic network location service based on DHT technology*. International Conference on Software Engineering Proceedings of the 2008 international workshop on Software Engineering in East and South Europe, Leipzig, Germany Pages 43-50 Year of Publication: 2008
ISBN:978-1-60558-076-0

Mwansa L., Janeček J *P2P: Generic Network Location Service for Web Applications 2007*, In 9th Annual Conference on World Wide Web Applications [CD-ROM]. Cape Town: Cape Peninsula University of Technology, 2007, p. 1-16. ISBN 978-0-620-39837-4.

Mwansa L., Janeček J, *Ensuring fault tolerance in generic network location service*, 2008, In Proceedings of the 22nd European Conference on Modelling and Simulation. Brusel: European Council for Modelling and Simulation, 2008, p. 254-260. ISBN 978-0-9553018-5-8.

PlanetSim, Available <http://planet.urv.es/trac/planetsim/>