ENSURING FAULT-TOLERANCE IN GENERIC NETWORK LOCATION SERVICE

Laban Mwansa
Czech Technical University in Prague, Department of Computer Science and Engineering
Karlovo nam. 13, CZ12135 Praha 2, Czech Republic
mwansl1@fel.cvut.cz

Jan Janeček
Department of Computer Science and Engineering,
Czech Technical University in Prague
Karlovo nam. 13, CZ12135 Praha 2,
Czech Republic
janecek@cs.felk.cvut.cz

KEYWORDS
Network addressing, replication, name service, location service, DHT.

ABSTRACT

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 hashing mechanisms.

Specifically, we first discuss the design of an alternative approach to address translations based on DHT usage then report how fault-tolerance can be ensured through replication strategies namely proactive and reactive. As a result, we show how 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 device identifications has some advantages: straightforward implementation of all possible translations at the same time (provides efficient discovery mechanisms for 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.

INTRODUCTION

The current miniaturization of computer systems and recent advents in computer storage and memory technologies has brought about new dimensions in research. Notably, the continuing decline in memory prices means that huge amounts of storage is available with considerably high speeds of data transfer from primary to secondary storage. This ensures that data can be replicated synchronously and asynchronously with relatively low overheads concerning acceptable lag time in restoring replicated data. Synchronous replication in this case means that data is 100% synchronized at both the primary node and remote node whereas asynchronous replication will allow for the choice of an acceptable lag time for replicating and restoring data.
Each replication approach requires careful consideration of the infrastructure including architecture framework and intended application software. Some of the factors include the rate of recovery, recovery point objectives as well as the impact on performance of the system. This presents an opportunity for researchers to analyze efficient replication strategies for optimal system performance.
In this paper, we describe reactive and proactive replication strategies in ensuring high availability of data. The scope is limited to GNLS (Mwansa and Janecek 2007) in which the architectural framework is based on Chord DHT system (Stoica et al. 2001), essentially because of its wide acceptance and simplicity; other DHT technology might be used as well.
The contribution of this paper is an attempt to extend the GNLS system described in (Mwansa and Janecek 2007) by describing reactive replication strategies of ensuring fault-tolerance of data at various servers. PlanetSim simulator was used to obtain experimental data presented.
The following section presents some notes on DHT systems, specifically Chord followed by related work, description of the architectural framework and functions of the General Network Location Service (GNLS). Thereafter we briefly describe the importance of replication and mechanisms assuring fault-tolerance to failures. Lastly conclusion including future research perspective

Distributed Hash Table

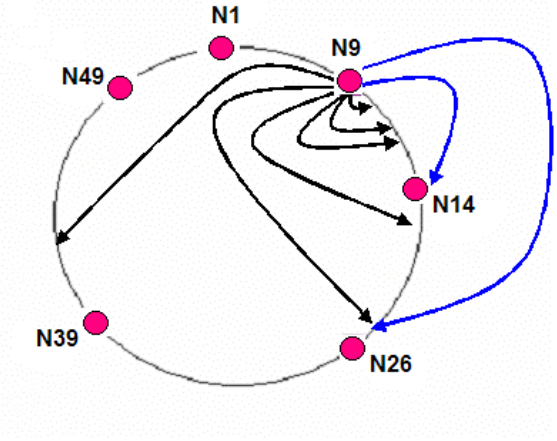
DHT (Distributed Hash Table) systems have received considerable research interest due to their attractive properties. We can refer to the DHT's described in (Stoica et al. 2001) (Ratnasamy et al. 2001) (Balakrishnan et al. 2003). The hash table interface is an attractive foundation for a distributed lookup algorithm because it places few constraints on the structure of DHT keys and the data they refer.

Proceedings 22nd European Conference on Modelling and Simulation ©ECMS Loucas S. Louca, Yiorgos Chrysanthou, Zuzana Oplatková, Khalid Al-Begain (Editors)
ISBN: 978-0-9553018-5-8 / ISBN: 978-0-9553018-6-5 (CD)

DHT implements just one operation: lookup(key) yields the identity (e.g., IP address) of the node currently responsible for the given key. A simple distributed storage application might use this interface as follows: someone who wants to publish a data / file under a particular unique name would convert the name to a numeric key using an ordinary hash function, say SHA1 (Lewin 1998), then call lookup(key). The publisher would send the data / file to be stored at the resulting node. Someone wishing to read that file would obtain its name, convert it to a key, call lookup(key), and ask the resulting node for a copy of the data / file.
A complete storage system base on DHT would have to take care of replication, caching, authentication and other issues; these are outside the immediate scope of the lookup problem. To implement DHT's lookup algorithms, the following issue has to be addressed: mapping keys to nodes, preferably in a load balanced way. All DHT systems do this in essentially the same way.
Both node IDs and keys are represented by numerical identifications DHT IDs. The key is then assigned to the node with the closest node ID using a specific metric (e.g., the one that is the closest numeric successor in Chord, the one with the longest matching prefix in Kademlia, the one having the lowest distance in OpenDHT).

Chord

Chord (Stoica et al. 2001) 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 (Figure 1).



Figures 1: Chord finger table keys and node successors (N =26)

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 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 logN) make the probability of simultaneous failure vanishingly small.
Chord reconfigurations
A new node n finds its place in the Chord ring by asking any existing node to lookup n's ID. All that is required for the new node to participate correctly in lookups is to update its and its predecessor successor lists. Chord does this in a way that ensures correctness even if nodes with similar ID's join concurrently. The new node and the existing nodes, have to update their finger tables; this happens in the background because it is only required for performance, not correctness. The new node must also acquire whatever data is associated with the keys it is responsible for, the successor relationship ensures that all of these keys may be fetched from the new node's successor.

RELATED WORK

GNLS was inspired by previous work on location lookup services and name resolution technologies based on DHTs.
Applications of DHT technology for Internet name service support are numerous. As examples, we can present DDNS (Dynamic DNS) (Cox et al. 2002) based on Kademlia and Microsoft's PNRP (Peer Name Resolution Protocol) (Huitema and Miler 2002) 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 (Singh and Schulzrinne 2006).
In (Bryan et al.2005), the authors describe a DHT overlay network based on Chord (Stoica et al. 2001) 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 (Yusuke 2005) 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.

(Risson et al. 2006) 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. (Quanhao et al. 2006) 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. (Fiedler et al. 2006) shows how peer-to-peer technologies can be integrated in load balancing and failover requirements with a centralized VoIP server concept.

GENERIC NETWORK LOCATION SERVICE (GNLS)

In this section we will briefly describe the Generic Network Location Service (GNLS) according to (Mwansa and Janecek 2007). 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.

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.

Location records and Collections

The basic idea of the generic location system 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).

Suppose we have several disjoin sets S1, S2, ...Sn of names, identifying network devices in different naming systems. As location records we will denote records, consisting of names from different sets Si, e.g. $r = \{s_{1r}, s_{2r}, ..., s_{nr}\}$, where $s_{1r} \in S_1$, $s_{2r} \in S_2$, ..., $s_{nr} \in S_n$, which are identifying a single network device.

For simplicity (and in our simulation studies), we can assume that location record identifies the device in all naming systems involved. However, leaving any of the names undefined presents no problem, an example of such a realworld situation could be a personal computer connected by Ethernet to Internet, its location record will definitely involve the MAC and IP addressees (and apparently the DNS name), whereas a SIP name need not be defined, if the computer does not run SIP VoIP.

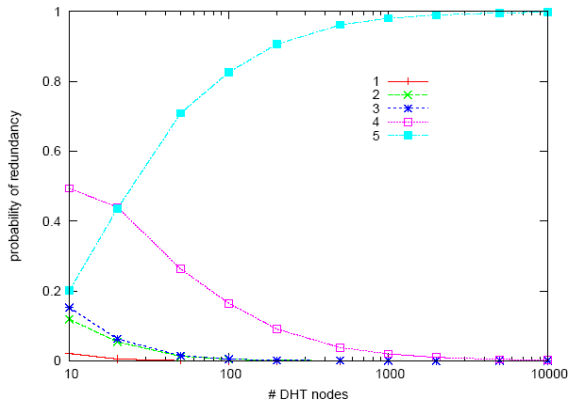
Assignment of names to devices is not static: a network device may get different IP address from DHCP after disconnection, it will definitely get different IP address when moved to different IP subnet (problem solved by Mobile IP), it will get modified GPS coordinates when moving around, a GSM phone changes its IMSI identity when we are replacing SIM. Well, there are many other situations when location information (represented by the location record) changes, the GNLS approach seems to be flexible enough to cover them all. Implementation of the dynamics presents no theoretical problem, modification of the location record can be implemented as a removal of the invalidated followed by an insertion of the modified version. However, the implementation should be optimized to increase effectiveness. More copies of the record (one identifiable by each of names involved) are stored in GNLS nodes, changing one name only in the record will result in replacement of one replica at most, others may be modified at their previous locations.

Devices having more names in any of the sets S_i (e.g. more IPv6 addressees, more DNS names, more SIP URIs) will be given more location records. Though such a solution presents no problem in theory, it could result in the explosion of number of location records in practice. (Number of location records increases with the power of numbers of names in different naming systems involved). To solve this issue we could use more complex structures, collections, in implementation of the location system.

Having disjoin sets S_1, S_2, ...S_n of names, identifying network devices in different naming systems, we will denote as collections sets of names identifying individual network devices, for example $c = \{s_{11c}, ..., s_{1m1\ c}, s_{21c}, ..., s_{nmn\ c}\}$, where $s_{1ic} \in S_1$, $s_{2ic} \in S_2$, ..., $s_{nic} \in S_n$. There is no need to limit number of names in a specific naming system (e.g. the network device with more IP addresses may have them included in a single collection) unless such a situation is, for the specific naming system, forbidden (for example a GSM phone is given a single IMEI, any physical network device will get a single GPS position, and so on). As a result, every network device can be given a single collection

involving all its names. Such a collection is valid for the specific device in a specific time instant, changing any of the names in the collection can be described as collection invalidation (removal of the collection from the location system) followed by insertion of the modified one. Well, since more replicas of the collection are stored in GNLS nodes, more elaborated set of functions will increase effectiveness.

As we mentioned in the previous paragraphs, in both approaches, based on location records and collections, we are storing several copies / replicas in DHT, each of the copies / replicas identified by the DHT key computed from one of the names and placed using this DHT key onto a corresponding DHT node (the successor of the DHT key in Chord). There is an interesting issue to estimate how many nodes will be used to store copies / replicas of the location record (or collection). Although the probability that location record involving n names will be placed in m different nodes (m <= n) can be evaluated analytically, we will present here only the results we have got from the simulation of the Chord DHT consisting from 10 to 10000 nodes (Figure 2). Probability of placement more copies / replicas onto a single node is acceptable even for tens of GNLS servers. Making use of this fact in the GNLS server's implementation provides acceptable fault-tolerance, even without data replication support at the DHT layer.



Figures 2: Probability of redundancy factor

The results indicate that, having sufficient number of GNLS nodes, the intrinsic redundancy is sufficient and probability of extreme concentration of replicas on a one or two nodes (well, excluding binary relations) can be vanishingly low. However, when required, location conflicts, defined as two or more replicas identified by different DHT keys placed at the same DHT node, can be solved by moving "overflowing" replica(s) to subsequent DHT node(s).

GNLS Architecture

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. Whereas servers are forming relatively stable infrastructure and can run DHT mechanisms effectively, clients are generally connected only for shorter time and may change their connection to GNLS servers. Restricting DHT mechanisms to the stable core of GNLS servers decreases overhead of DHT stabilization, which has to manage only node failures and infrequent core reconfigurations.

Though DHT technology promises effective lookup, enabling servers of the infrastructure and clients with caches would be necessary to get a service quality comparable to DNS. (Remember that GNLS provides basis for name translation unavailable in any current specific translation mechanism). Simple time constrained caches can be used, but more flexible distribution of change reports can be implemented as well at both levels, GNLS servers and clients.

Another technical problem is related to accessibility of GNLS. The simplest mechanism can be based, similarly to SIP, on= the standard DNS service, i.e. the client will need only a single DNS name to get access to one or more GNLS servers.

Generic Network Location Functions

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.)(RFC 3761).

The method is invoked by remote GNLS clients. The GNLS node converts individual name keys of the location record into DHT keys. Then it stores the copies of the location record in GNLS nodes designated by the Chord lookup applied to individual DHT keys.

A counterpart of the insert method is the get method that has the form location record = **get**(name) , where name is a data structure representing type and value of the selected name key. The method is invoked remotely from GNLS clients. The GNLS node converts the name key into the DHT key and finds, using the Chord's lookup function, the node, 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.

Finally, location record can be removed by the method **remove**(name) , which takes any of the name keys as the parameter name. The GNLS node, which is addressed by the remove request gets a corresponding copy of the location record and uses contained name keys to locate and remove all location record replicas from DHT.

The simple interface just described covers functionality of the GNLS system. However, real world implementation should include the modify method, for example in the form **modify**(location record) , which will remove, insert and locally modify individual

replicas of the location record on the basis of the previously stored record and the modification request.
To illustrate function of the GNLS system we will present an example describing insertion of location records for two devices in the Chord DHT which is using 6bit ID space (the real world Chord uses 160bit IDs). Location records of the devices includes MAC address, IP address, fully specified DNS name, phone number in ENUM format and GSM identification, and are presented in Figure 3.
Location record 1

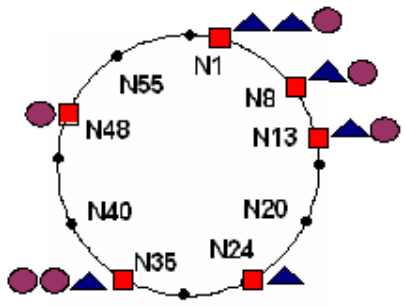
Key	Value	Hash
MAC	00-D0-8F-59-7C-4B	3
IP	10.0.4.6	10
DNS	CS.FELK.CVUT.CZ	15
ENUM	7.4.1.0.6.4.9.7.0.2.4.4.e164.arpa	28
GSM BPS	284011234567890	50

Location record 2

Key	Value	Hash
MAC	00-BB-6E-59-7D-4B	30
IP	128.0.3.5	51
DNS	CS.PUMBA.MSU.ZA	44
ENUM	5.3.1.0.6.4.9.7.0.2.4.4.e164.arpa	30
GSM BPS	386711234567890	11

Figures 3: Location Records

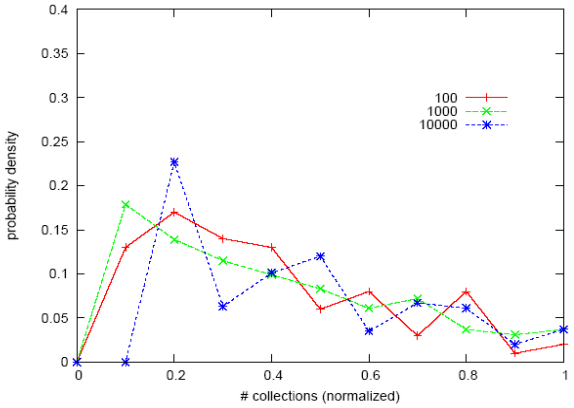
Distribution of location records replicas onto GNLS nodes is presented in Figure 4.



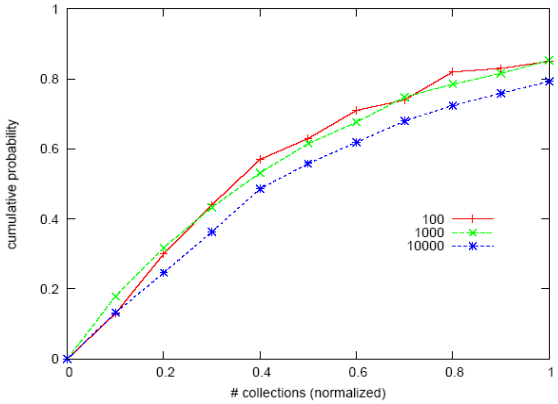
KEY
Location record device 1.
Location record device 2.
Chord DHT node

Figures 4 Distribution of location records

Nodes in Chord are distributed randomly over the DHT ID space, the consequences of this fact are uneven distances between nodes and therefore uneven number of records they store. The distributions we observed on the model of GNLS in PlanetSim are presented in the following graphs. Figure 5 depicts the probability density function and Figure 6 the cumulative probability, both for the number of collections of five names stored at GNLS nodes. Results are presented for sets of 100, 1000 and 10000 randomly distributed GNLS nodes and one million of stored collections. Results are normalized with respect to average number of collections at the nodes.



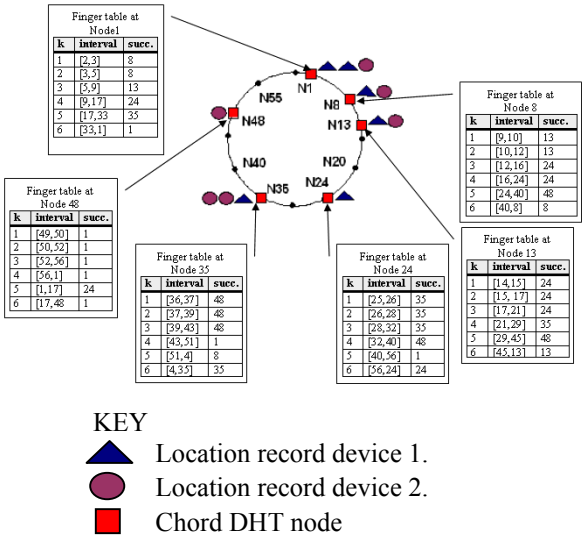
Figures 5 Distribution of collections – density



Figures 6 Distribution of collections - cumulative

FAULT-TOLERANCE

The GNLS system decreases sensitivity to failures by replication of individual location records. Every location record containing n location IDs is stored in n copies in GNLS.
Although intrinsic replication of location records and their distribution 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 a mechanism able to recover from damages resulting from individual node failures.
Moreover, the DHT systems themselves include mechanisms, which assure correct DHT functioning after a node failure occurs.
In the following paragraph we propose our solution for replication maintenance problem in GNLS and possibilities of data reconstruction after a node failure. We use an illustrative example in figure 4 using two location records. The figure below shows the placement of the replicas as well as the Chord fingers tables.



KEY

Location record device 1.

Location record device 2.

Chord DHT node

Figures 7: Distribution of location records

We illustrate distribution of location records onto GNLS nodes together with theirs finger tables (Figure 7).

Replication Strategies

Though DHT systems is equipped by additional mechanisms, which are able to avoid loss of stored data GNLS system provides additional possibilities to fault-tolerance of stored data from node failures. Proactive strategy entails mechanism able to assure high availability of location records in GNLS system is to replicate them at some node, which takes on an original node's function after the failure.

A drawback of the proactive strategy is that it almost doubles (for higher number of nodes) memory requirements. Reactive strategy employs 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.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.

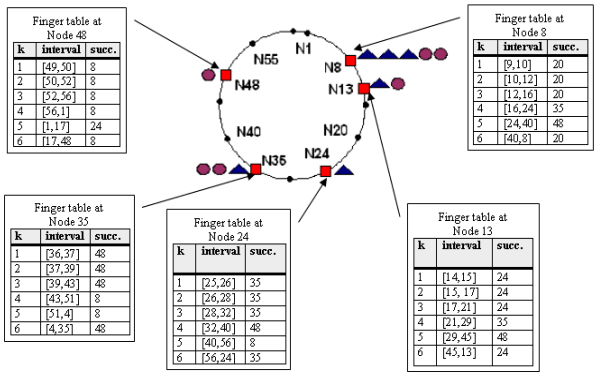
The risk of placing all copies to a single failing node is acceptably low for higher number of DHT nodes. Moreover, such a situation can be simply detected and the additional copy of the location record can be created elsewhere, e.g. at the node's successor. Any node may be used as a place for such an 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).

Replica reconstruction after node failure

Figure 8 illustrates the distribution of location records in the GNLS after node 1 has been removed. The removal of the node from the DHT ring network maybe as a result of a node failure or voluntary withdraws.

Once a node having a replica placement is detected as having failed, a replica reconstruction mechanism is triggered. If a node does not respond for sometime, a time out situation occurs and it would be concluded that the node is either dead or unavailable.

One technique to solve this problem of replica reconstruction after node failure, is to reconstruct additional copies of location records using hashing mechanism as a replacement copy. Figure 8 shows the contents of the Chord network after reconstruction. The location record replicas will be placed on new node id's after a network stabilization protocol is run.



KEY

Location record device 1.

Location record device 2.

Chord DHT node

Figures 8: Distribution of location records and content of finger tables after node 1 has been removed

CONCLUSION

The paper presents the idea of storing location records, defined as collections 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. Replication strategies are discussed ensuring the availability of data, to clients

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 \(O(logN)\) complexity of the lookup.

Fault-tolerance to failures can be based on stored replication records without creating additional

secondary copies (up to some very infrequent cases mentioned). The GNLS 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. Future research challenges include investigating the cost and latencies of messages exchanges involved in replica reconstruction after node failure.

REFERENCES

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*", AAAIDEA* 2005, 42-49

Cohen E., Shenker S. "Replication strategies in unstructured Peer-to-Peer Networks." In: *Proceedings of the ACM Sigcomm02 Conference*, August 2002.

Cox R., Muthitacharoen A., Morris R."Serving DNS Using a Peer-to-Peer Lookup Service", *Revised Papers from the First International Workshop on PeertoPeer Systems,* LNCS 2429, Springer 2002, 155-165.

Yusuke Doi: "DNS Meets DHT: Treating Massive ID Resolution Using DNS Over DHT", *The 2005 Symposium on Applications and the Internet (SAINT'05)*, 2005.

Fiedler J., Kupka T., Magedanz T, Kleis M. "Reliable VoIP Services Using a PeertoPeer Intranet", *Eighth IEEE International Symposium on Multimedia (ISM'06)*, Dec. 2006, 121-130.

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

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

RFC 3761, "The E.164 to Uniform Resource Identifiers (URI) Dynamic Delegation Discovery System (DDDS) Application (ENUM)".

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

Stoica I., Morris R., Karger D., Kaashoek M.F., Balakrishnan H. "Chord: A Scalable PeertoPeer Lookup Service for Internet Applications", *Proceedings of the 2001 ACM SIGCOMM Conference*, 2001.

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

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

Ratnasamy S., Francis P., Handley M., Karp R., Shenker S. " A scalable content addressable network", *Proc. ACM SIGCOMM*, San Diego, CA, August 2001, 161-172.

Risson J., Qazi S., Moors T., Harwood A. "A Dependable Global Location Service using Rendezvous on Hierarchic Distributed Hash Tables", *International Conference on Networking*, April 2006, 4.

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

AUTHOR BIOGRAPHIES

LABAN MWANSA was born in Zambia, received his Masters degree from Moscow State University of Statistics and Economics, He has worked both in Zambia and South Africa as a Senior Lecturer in Information Technology. He is currently a PhD candidate in Computer Science at the Czech Technical University in Prague. He is a member of the South Africa Computer Society and IEEE. His e-mail address is mwansl1@fel.cvut.cz.

DOC. ING. JAN JANEČEK, CSC. Associate Professor, Academic Board Member, Head of the distributed systems research group, Supervisor, Teacher Czech Technical University in Prague. His e-mail address is janecek@cs.felk.cvut.cz