Improving Spatial Indexing and Searching for Location-Based DNS Queries

Daniel Moscoviter, Mozhdeh Gholibeigi, Bernd Meijerink, Ruben Kooijman, Paul Krijger, Geert Heijenk

Improving spatial indexing and searching for location-based DNS queries

Daniel Moscoviter[1], Mozhdeh Gholibeigi[1], Bernd Meijerink[1], Ruben Kooijman[2], Paul Krijger[2], and Geert Heijenk[1]

[1] University of Twente
Enschede, The Netherlands
d.moscoviter@student.utwente.nl,
{m.gholibeigi,bernd.meijerink,geert.heijenk}@utwente.nl
[2] Simacan B.V.
Amersfoort, The Netherlands
{ruben.kooijman,paul.krijger}@simacan.com

Abstract In the domain of vehicular networking, it is of significant relevance to be able to address vehicles based on their geographical position rather than the network address. The integration of geocasting (i.e. the dissemination of messages to all nodes within a specific geographical region) into the existing addressing scheme of the Internet is challenging, due to its logical hierarchy. One solution to Internet-based geographical addressing is eDNS, an extension to the DNS protocol. It adds support for querying geographical locations as a supplement to logical domain names. In this work, eDNS is extended with nearest neighbor resolution support, and further, a prototype server is developed that uses bounding box propagation between servers for delegation. Our experiments confirm that distributing location records over multiple servers improves performance.

Keywords: Geocasting, Vehicular networks, DNS, eDNS

1 Introduction

The concept of Intelligent Transportation Systems (ITS) is an emerging area of research [11]. The main objective of such systems is to use vehicular communication to develop novel applications for increasing safety, traffic management, Internet access, or other valuable services. Aside from the opportunities for delivering many novel applications, a significant amount of research has focused on ITS because of the technological difficulties that are involved. Mainly, vehicular networks have to deal with highly dynamic network topologies of vehicles, their high speed, limited communication ranges, and real-time constraints of potential applications. We can differentiate between two different types of communication in vehicular networks: Vehicle-to-Vehicle (V2V) and Vehicle-to-Infrastructure (V2I) communication. In the former type, data packets are exchanged between vehicles using vehicular communication technologies without involvement of an

infrastructure. The latter type extends the vehicular ad hoc networks (VANETs) with a fixed infrastructure (e.g., Roadside Units (RSUs)).

Vehicles in ITS will typically be equipped with localization technologies. This allows vehicles to be addressed based on their geographical positions rather than the network address (i.e. targeting a certain area, not a certain vehicle), utilizing domain-specific forwarding strategies [1]. Messages can be sent to any single node within a target region (geoanycast), or to all nodes within a target region (geocast). Geocasting in particular enables a large number of new applications. Warning about dangerous road conditions, assisting in speed management, and delivery of infotainment are examples of use cases that geocasting can facilitate.

Geocasting requires a routing protocol that delivers messages to the intended targets. It is a challenge to integrate geographically-scoped broadcasting into the existing addressing scheme of the Internet, as IP-based addressing does not support geographical routing. Though solutions to this problem have been proposed, there are bottlenecks and the concept is an open research issue. For instance, GPS-based addressing and routing [9,10] requires a specialized infrastructure; use of a geographical IPv6 prefix format [8] relies on a standardized allocation of IPv6 addresses; and GeoNet [4,12] has Internet-wide scalability limitations [6].

One solution that was proposed to tackle these shortcomings is the Extended DNS (eDNS) [5,6]. It is based on the Internet Domain Name System (DNS) protocol, and extends it to support geographical addressing. While DNS already supports storing locations using location (LOC) resource records, the novelty of eDNS is that the locations can be used as a primary key to return Internet Protocol (IP) addresses that are associated with geographical regions. The appeal of this method is that it does not require specialized hardware or software, nor protocol modifications. Only modification of existing DNS implementations is required, as described later in this document. Support for more efficient indexing, as well as delegation was added at a later stage [17]. However, opportunities for improvement remain. For example, we can consider use cases where one would be interested in entities close to a certain point, rather than entities within a region. More possible improvements can be identified, and will be discussed further in this paper.

As mentioned earlier, the concept of integrating geocasting into the existing infrastructure of the Internet is an important research issue in the field of ITS research. Because of the aforementioned shortcomings in several proposed solutions, this paper focuses on the eDNS protocol as a solution for the problem of addressing all entities within a geographical area. In our work, we have improved and evaluated the eDNS protocol in several ways. This has resulted in the following main contributions.

- We have designed and implemented nearest neighbor resolution by introducing a new DNS resource record containing a location's distance to a queried area.
- We have designed and implemented the propagation of bounding boxes to parent DNS name servers, upon receiving location updates via the Dynamic DNS (DynDNS) protocol.

– We have evaluated the performance of our eDNS implementation in terms of throughput and latency for various input parameters.

The remainder of this paper is structured as follows. We review related work on eDNS in Section 2. Then, we discuss our approach to improve eDNS in Section 3. Section 3 shows the performance evaluation of our developed prototype. Finally, we draw conclusions and discuss future work in Section 5.

2 Background and Related Work

Given the importance of geographical location based addressing in the domain of vehicular networking, various research activities have been carried out over the recent years. In this section we refer to some research relevant to our work.

Fioreze and Heijenk [5] propose the extension of DNS such that clients can resolve IP addresses based on geographical coordinates, rather than domain names. The proposal relies on the existing LOC record specification. The novelty of the proposal is that LOC records are allowed to be used as the primary key for DNS queries, in addition to the methods of using hostnames or IP addresses. The eDNS proposal has various strengths. For one, it is based on the existing DNS architecture, which has proven its high scalability, being used as the addressing scheme in the Internet. Secondly, it does not require specialized hardware or software, or modification of existing protocols.

In [6], a prototype implementation is described based on Name Server Daemon (NSD)[3]. The prototype follows the suggestions made in the proposal by adding support for the use of LOC records as the primary key. Geographical queries have the following format, based on the format of the LOC record: `‘(’dLat mLat sLat ‘N’|‘S’ dLon mLon sLon ‘E’|‘W’ alt[‘m’] size[‘m’]‘)’.domain`. The geographical query format is hybrid, in the sense that logical domain names can be mixed with a geographical location. The geographical part is used on the lowest level. Thanks to this property, top-level domains are not required to support the geographical format. The document also describes a delegation strategy.

Westra [17] extended eDNS based on the previously mentioned implementation. The extended prototype is based on R*-trees [2] as an efficient spatial data indexing and searching method. R*-trees are a variant of R-trees, which are dynamic, hierarchical data indexing structures. To reduce network traffic for delegation, [17] introduces the bounding box (BND) record type which defines the bounding rectangle of child servers, preventing blind delegation of a location request if it falls outside of the known coverage of the child server. Compatibility with existing DNS implementations is preserved. The query format was extended to provide a higher precision.

Previous iterations of eDNS have mainly focused on storing location of nodes that have a fixed location, such as RSUs. In [13], Van Leeuwen extends eDNS with functionality to dynamically manage locations. In the work, it is assumed that a central server exists that tracks dynamics of the environment. Functionality is

[3] https://www.nlnetlabs.nl/projects/nsd/

added to the modified NSD server that retrieves records from this server, rewrites its own zone file, rebuilds the database, and reloads. However, this method does not work if no central server with locations exists.

3 The Approach

The problem of finding nearby nodes in tree data structures is known as k-nearest neighbor (kNN) resolution in the literature, where k is the number of closest results. This can be used to find the nearest RSU that has at least a certain number of vehicles in its coverage range. These vehicles may potentially forward information to the target geocast area in multiple hops. A client could request nearest neighbor resolution in an eDNS query by appending a parameter (`'nn='nn`) to the geographical coordinates. The authoritative name server parses this parameter and performs an algorithm to find nearby neighbors, ordered by distance, rather than finding overlapping LOC records.

Nearest neighbor resolution becomes more complicated when delegation is to be considered. Although it is possible for a name server to ask each subdomain about its nearest neighbors, it is not doable to combine results from multiple sources and selecting the nearest neighbors from that set, without knowing the individual distances. One possible solution would be to have the authoritative server request the actual LOC records from the subdomain, in addition to the record type originally requested. The authoritative server could then parse these records, apply its own distance calculations, and compare these results to the distances of its own LOC records. Another solution is to have the authoritative servers for the subdomains report the distances for its results. Every unique record name would require its distance to be reported. There is no standardized way to transfer such a distance value, so we propose the use of a new volatile record for every unique result name, the distance (DST) record. These records are not added to the persistent zone storage of the authoritative server, but instead only generated temporarily for inclusion in the query answer. Its definition is an implementation of the TXT record: `'v=dst1' distance`. The distance is a decimal value in meters. Because each authoritative server already knows the distances of the requested shape to its own LOC records after doing the local nearest neighbor resolution, no additional computation is required. The parent authoritative server uses these temporary records to order the other result records, and trim the number of records to the requested number. Because DST records are essentially metadata, they are returned in the 'additional' section of DNS query answers. This is comparable to how the OPT pseudo-resource record (RR) is returned for the Extension mechanisms for DNS (EDNS(0)) protocol [15,3].

One considerable drawback of the first proposed solution is that it increases the computational burden on the non-leaf authoritative servers, as they would have to perform additional distance calculations. These distance calculations are wasteful, because they were already performed by the subdomains to return the initial nearest neighboring records set. After considering this imposed computational overhead, we chose to implement the second proposed solution in the prototype.

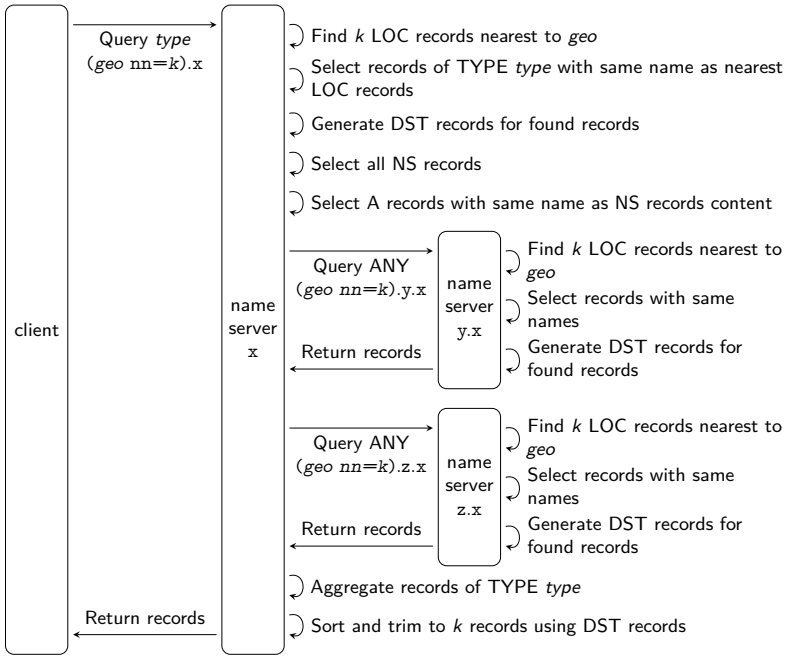

Figure 1: eDNS nearest-neighbor resolution with delegation.

A visualization of the processing done for nearest neighbor resolution in a delegated deployment is shown in Fig. 1 for name server `x` with two subdomains. Note that unlike the process for diameter-based delegation, no overlap with BND records is checked, because all subdomains have to be queried regardless of the result. Requests are sent to subdomains for ANY records, rather than the type requested by the client, as we want to receive DST records in addition to the requested type. It is argued by some that ANY queries should be deprecated to prevent their use in amplification attacks [7], which attempt to overload a victim's bandwidth capacity. DNS ANY queries are well-suited to this attack type, because the response size of a request is significantly larger than the request itself. We therefore note that the subdomain requests can alternatively be performed using two separate queries for the *type* and TXT (the base type of DST) records. Assuming `x` and each of its s subdomains have at least k LOC records, the total number of records of the requested type that are known by `x`, will be $(1 + s) \times k$. They are typically not included in the records returned to the client, but are used to sort the resource record set (RRset) by distance and trim them to k records. These are returned to the client. If less than, or exactly k records of the requested type are known, all these records are returned.

The problem of managing dynamic nodes, such as vehicles, has been discussed in [13]. The author added support for dynamic nodes by implementing a process in the name server software that periodically reads an updated text file with a

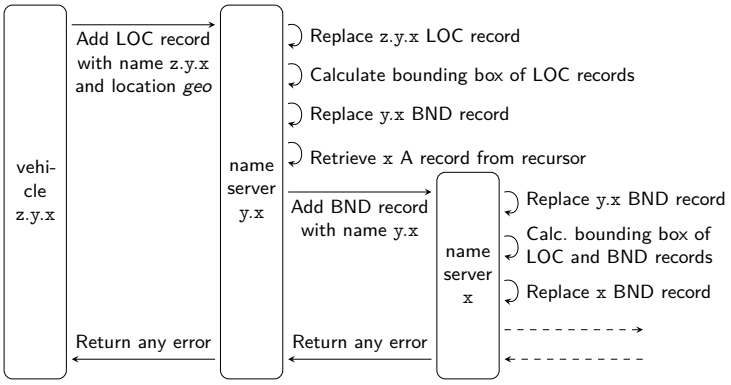

Figure 2: eDNS location updating.

list of nodes, rewrites its zone file, rebuilds the internal database, and reloads the server. It is noted that an alternative approach to this problem is to use the DynDNS Update protocol [16], but this protocol was not supported by the version of the name server software that was used.

We implement dynamicity for vehicles using DynDNS Updates by making use of a DNS name server that supports the DynDNS protocol. A layer of complexity is added when the problem of dynamicity is combined with delegation. An eDNS server needs to know the bounding box coverages of its child servers. If locations in the child servers change, the known bounding boxes may need to be updated as well. The process is visualized in Fig. 2. A similar process involves the deletion of a LOC record. Rather than inserting a new record, only an old entry is removed. Nothing changes from the perspective of `x`, as deletion can also result in the need to send an updated bounding box to the parent server. Finally, one edge case of deletion exists. If the last LOC record in a specific domain is deleted, there is no valid bounding box to be created. The related BND record needs to be removed, both in `y.x` and its parent name server `x`. The latter should therefore be removed with a new DNS Update deletion message.

4 Evaluation and Numerical Results

This section highlights the results of evaluating the implemented functionalities with various configurations and settings to get insight into system behavior in terms of throughput and latency. The evaluation is performed with sets of real historic vehicle location data to simulate the realistic use case of tracking vehicles in an eDNS system.

The prototype is deployed to clusters on Amazon Web Services for reliable performance testing. A PostgreSQL database instance is launched for each server instance, and all database instances are extended with PostGIS functionality. The server instances are of the type *m3.medium*. Experiments that attempt to

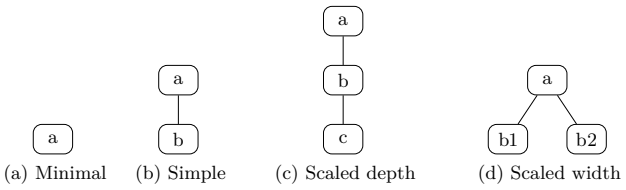

Figure 3: Evaluation server setups.

replicate or compare to our results should therefore be run on instances with equivalent computing performance.

4.1 Test Setups

Given the focus of our work, it is important to evaluate the performance of the system by querying nodes within a specified geographical region; querying nodes with the smallest distance to a specified geographical region; and updating nodes with dynamic locations. Location data is needed to evaluate these perspectives. We have used a historic data set of real vehicle location data. We represent such a location with a LOC record, as well as an A record with a fictional IP address of the vehicle.

We test the influence of server setups by configuring multiple name space tree setups. Various testing setups are visualized in Fig. 3. The setups provide a variety in terms of tree width and depth. Query performance is tested by querying the top node in the name space tree. The LOC records are divided over the edge nodes. The allocation of records to the edge nodes is close to optimal, in the sense that each edge node is responsible for its own geographical region. This causes the bounding boxes of the edge nodes to have no overlap. The performance of location updates is tested by sending DynDNS updates to one or more of the available edge nodes and measuring the performance. Various input parameters of the setup can influence system performance.

4.2 Performance Metrics

The performance of the system is quantified in terms of throughput and latency as the two most relevant performance indicators of the introduced system. For our use case, the throughput determines how many vehicles the system is able to keep track of, in terms of the number of queries and also the number of location record updates the system can handle within one second. The latency influences the responsiveness of the system, and how up-to-date the stored locations are. This metric specifies the time interval between issuing queries and receiving the corresponding reply back. Throughput has a strong inverse correlation to latency. Therefore, we only show throughput in the figures.

All DNS query and update operations are performed with a reasonable timeout of 1 second. If no result is received before this timeout expires, the operation is considered to have failed, and will not contribute to the overall throughput.

In our testing setup, each eDNS server connects to a database backend on another server. The latency between these servers can be measured to determine its influence on the other results. The `ping` command was used from an Elastic Compute Cloud (EC2) instance to determine the communication delay between two servers in the same availability zone. Executing consecutive `ping` requests showed that latency between servers in the same availability zone lies within the range $[0.634, 0.648]$ milliseconds with $\alpha = 0.01$.

For the sake of simplicity, we assume that the result data is normally distributed. This allows us to compute confidence intervals for the data points. All results are displayed with a 95% confidence interval ($\alpha = 0.05$). Note that some graphs contain confidence intervals that are too small to be visible.

4.3 Numerical Results

In order to get reliable results for throughput of the system, it is necessary to have reasonable packet arrival rate, such that the system does not wait for packets to arrive. On the other hand, the system should also not be overloaded to the point where it is unable to reduce its internal buffer or respond within timeout limits. To prevent the system from idling, it is necessary to send multiple packets simultaneously. Initial tests have been performed to evaluate the performance with different numbers of simultaneous packets. This is implemented as a thread pool, with each packet being sent in its own thread (a 'worker'). As expected, employing more workers generally results in higher throughput, although with significant diminishing returns. For every tested configuration, throughput converges to a certain rate where at least one non-buffer related performance bottleneck emerged, such as limited processor power. Even though higher number of workers do not negatively affect throughput performance, we saw that using an arbitrarily large number of workers is not reasonable because of latency. Latency appears to consistently increase when a larger number of workers is used. This happens because the server has to divide its resources over all incoming requests, resulting in longer processing times for each request. Based on these results, we do not consider there to be an optimal number of workers, as even for specific configurations it is a trade-off between throughput and latency. We have opted to perform the rest of the evaluation with 8 concurrent workers. With that number of workers, the majority of the potential performance increase from concurrency has been achieved under most testing configurations while having lower latencies than any higher number of workers.

We now evaluate the performance of both querying and updating of locations.

Querying Throughput and latency have been evaluated for every combination of setup, LOC record count and diameter. The LOC record count represents the number of LOC records present in the system. This can be evaluated with a

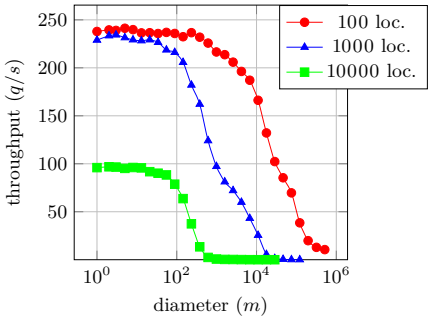

Figure 4: Querying throughput (setup (a)).

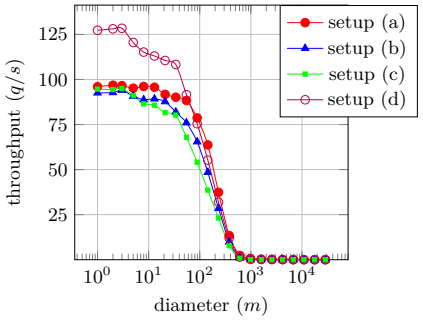

Figure 5: Querying throughput (10000 locations).

varying number of LOC RR inserted in the eDNS servers. We have taken subsets of various sizes from the previously mentioned data set to evaluate the influence of the amount of LOC records in the databases. Three subsets containing 100, 1000 and 10000 locations were created. Since each of the locations represents a vehicle, the diameter of the corresponding LOC record should be small. We have chosen the default LOC diameter of 1 meter as the diameter for each record. Multiple setups are tested, and records are added to one or more edge nodes based on their geographical position. Each edge node is allocated roughly the same number of LOC records. Upon receiving a DynDNS update message that inserts or removes a LOC or BND resource record, the BND record of the node itself will need to be renewed. The total number of LOC and BND servers can therefore influence the performance as they increase the number of spatial database operations that need to be performed.

The query location diameter represents the size of the queried circular area, specified via a diameter. It can be tested on the same data set by querying various area sizes. For this, we can consider querying areas with diameter sizes as low as 1 meter up to 500 kilometer. The larger sizes in this range would encompass our entire data set. The query locations originate from the same data set as the locations described earlier, but represent a different subset. They do share the same characteristics of being more likely to refer to a location on a highway.

The throughput of the system with setup (a) is shown in Fig. 4 . One may note that both the query diameter and data set size have a significant influence on the results. Given that a throughput of around 240 queries per second is achieved for both the data set of 100 and 1000 locations with low diameter sizes, we can conclude that the performance is not limited by spatial computations at this point. Rather, it is likely that network performance or packet handling overhead are responsible for the bottleneck. Additionally, the graph shows that for large diameter sizes and large data set sizes, the throughput approaches 0. This indicates that within the timeout limit, the system is not able to return a result with a large amount of matching locations. The performance for queries applied to the data set of 100 locations also appear to converge, but to a different

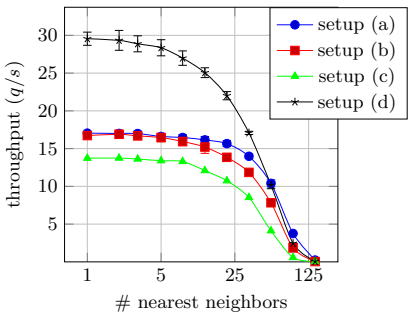

Figure 6: Querying throughput (nearest neighbors, 10000 locations).

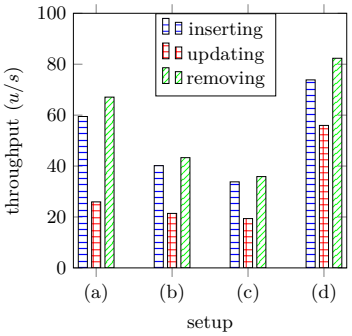

Figure 7: Updating throughput (10000 locations).

value. This can be explained by the fact that queries with diameter sizes larger than roughly 200 kilometers already encompass most of the location points in the data set, so increasing the diameter further will not increase the number of results.

Equivalent comparisons have been made for other setups, but graphs for these are omitted because they show the same pattern of larger data set sizes resulting in slower queries. Instead, we consider the performance of identical data set sizes between different setups. With data sets of 10000 locations, shown in Fig. 5, performance of setup (c) rises above that of the other setups for small diameters. This is likely the result of the system being able to divide its spatial computations over the two edge nodes. At a diameter of around 100 meters, throughput becomes lower than setup (a)'s before converging to the same value.

The same comparison between setups for nearest neighbor queries is shown in Fig. 6. The query nearest neighbor count represents the number of requested nearby results. It can be tested with simple integers. The same location data set as described for the query location diameter is used. Results roughly follow the patterns discovered in the evaluation for diameter-based queries. Queries on a system with this number of locations tend to perform better on setup (d)'s load distribution, up to a point where the portion of packets that does not fit within the specified timeout window becomes large enough that setup (a)'s single server is able to more reliably provide a result.

Updating We have evaluated three aspects of the dynamic location functionality: insertion, updating and removal. The insertion operation involves adding an A record with an IP address, as well as an entity's initial location in a LOC record. For updating a location, we assume that an entity's IP address has not changed. The updating operation therefore involves removing an old LOC record and replacing it with one containing a new location. The removal operation removes both the last know location in the form of a LOC record, but also the associated A record. The three different aspects therefore each involve two database operations.

The aspects are evaluated as follows for a data set with x locations. For each location, a DNS Update message is sent that inserts both a fictional IP address and a location. Then, all x locations are updated with individual DNS Update messages that remove the old LOC record and insert the new one. Finally, x DNS Update messages are sent that remove all entities. As with the query evaluation, all operations are executed with 8 concurrent workers to increase utilization. Storage of locations is also distributed over different edge nodes depending on geographical coordinates, as described in Section 4.1.

In Fig. 7, we show the performance of the three mentioned operations when executed on different setups and using the 10000 locations data set. As expected, introducing more depth to the server tree results in a higher latency, because each parent server needs to be updated with the renewed bounding boxes of its direct child servers. Setup (d) shows that the system scales well in width. Dividing LOC records over multiple servers appears to increase performance, even if it includes the extra operation of updating the parent server compared to setup (a). Updating locations is more expensive than the insertion and removal operations. At least part of this discrepancy can be explained by observing that the updating operation is always performed on a system that contains all x locations in a data set. The insertion and removal operations work on a system with 0 to x entities, depending on how many entities have already been inserted or removed.

5 Conclusions and Future Work

In this paper we introduced two main improvements to the eDNS protocol. To solve the problem of finding nearby results, we have described an approach for nearest neighbor resolution functionality. This can be used when vehicles need to be addressed via RSUs, but RSUs provide incomplete coverage of an area.

After previous works have shown that updating individual entities with new locations is possible, we have also added the concept of dynamicity in eDNS using the standardized DynDNS method. This allows us to track the locations of vehicles in LOC resource records by keeping coverage information synchronized over servers in the system.

Performance has been evaluated for queries and updates. The performance is shown to be strongly dependent on server setup, as well as input parameters. The test results show that in most of the evaluated scenarios, horizontal scaling of server setups frequently influences the performance positively, while vertical scaling always influences the performance negatively. Distributing LOC records over multiple servers allows the system to perform its calculations faster, potentially improving both throughput and latency.

In future work, focus can be placed on improving compatibility of the eDNS protocol with the existing DNS protocol. This includes changing the query format to only include characters that are allowed by default, as well as making use of the OPT pseudo resource record for communication of distances between servers. Additionally, the scaling of the protocol may be evaluated more extensively.

Acknowledgments This paper is a result of research performed for a Master thesis [14]. The authors would like to thank the support provided by OVSoftware B.V. and Simacan B.V..

References

1. Baldessari, R., Bödekker, B., Deegener, M., Festag, A., Franz, W., Kellum, C.C., Kosch, T., Kovacs, A., Lenardi, M., Menig, C., et al.: Car 2 car communication consortium manifesto. Tech. rep., CAR 2 CAR Communication Consortium (2007)
2. Beckmann, N., Kriegel, H.P., Schneider, R., Seeger, B.: The R*-tree: an efficient and robust access method for points and rectangles, vol. 19. ACM (1990)
3. Damas, J., Graff, M., Vixie, P.: Extension mechanisms for dns (edns(0)). STD 75, RFC Editor (April 2013)
4. Ernst, T.: Final geonet architecture design (January 2010)
5. Fioreze, T., Heijenk, G.: Extending dns to support geocasting towards vanets: A proposal. In: Vehicular Networking Conference (VNC), 2010 IEEE. pp. 271–277. IEEE (2010)
6. Fioreze, T., Heijenk, G.: Extending the domain name system (dns) to provide geographical addressing towards vehicular ad-hoc networks (vanets). In: Vehicular Networking Conference (VNC), 2011 IEEE. pp. 70–77. IEEE (2011)
7. Gudmundsson, O., Majkowski, M.: Deprecating the dns any meta-query type (March 2015)
8. Hain, T.: An ipv6 geographic global unicast address format. Internet-Draft draft-hain-ipv6-geo-addr-02, IETF Secretariat (July 2010)
9. Imieliński, T., Navas, J.C.: Gps-based addressing and routing. RFC 2009, RFC Editor (November 1996)
10. Imieliński, T., Navas, J.C.: Gps-based geographic addressing, routing, and resource discovery. Communications of the ACM 42(4), 86–92 (1999)
11. Karagiannis, G., Altintas, O., Ekici, E., Heijenk, G., Jarupan, B., Lin, K., Weil, T.: Vehicular networking: A survey and tutorial on requirements, architectures, challenges, standards and solutions. Communications Surveys & Tutorials, IEEE 13(4), 584–616 (2011)
12. Kovacs, A.: Final geonet specification (January 2010)
13. van Leeuwen, J.: Dynamicity Management in Domain Name System Resource Records. Bachelorreferaat, University of Twente (2014)
14. Moscoviter, D.: Improving spatial indexing and searching for location-based DNS queries. Master's thesis, University of Twente (2016)
15. Vixie, P.: Extension mechanisms for dns (edns0). RFC 2671, RFC Editor (August 1999)
16. Vixie, P., Thomson, S., Yakov, R., Bound, J.: Dynamic updates in the domain name system (dns update). RFC 2136, RFC Editor (April 1997)
17. Westra, M.: Extending the Domain Name System with geographically scoped queries. Master's thesis, University of Twente (2013)