Implementation and Evaluation of Distributed Geographical Routing

Bernd Meijerink, Geert Heijenk

To cite this version:

Bernd Meijerink, Geert Heijenk. Implementation and Evaluation of Distributed Geographical Routing. International Conference on Wired/Wireless Internet Communication (WWIC), Jun 2018, Boston, MA, United States. pp.121-133, 10.1007/978-3-030-02931-9_10 . hal-02269727

HAL Id: hal-02269727
https://inria.hal.science/hal-02269727

Submitted on 23 Aug 2019

Implementation and Evaluation of Distributed Geographical Routing

Bernd Meijerink and Geert Heijenk

University of Twente, The Netherlands
{bernd.meijerink,geert.heijenk}@utwente.nl

Abstract. Geocast has the potential to facilitate message delivering for geographically scoped information in many future scenarios such as vehicular networking and crisis control. An efficient geographic routing protocol is needed to enable Internet-wide geocast on the network level. In this paper we evaluate an implementation of a path based geographic routing protocol. We specifically look at the behavior and performance of this protocol during network convergence. We show that our implementation constructs forwarding trees that are close to a shortest path tree in link cost. We also show that our algorithm converges relatively quickly in case the network changes.

Keywords: Geocast, routing, route distribution

1 Introduction

Due to the increase in networked devices, vehicular networks in particular, we believe that location based packet delivery or geocast will become an important method of data distribution in the future. It could provide benefits for vehicular networks in the form of location dependent weather warnings or, on a more local level, information on traffic incidents or road conditions [3].

Geocast, first introduced by Navas and Imielinski [12], is a transmission method where packets are sent to a location or area rather than an IP-address. It can be seen as a one-to-many or many-to-many system like multicast with the main difference that devices receive packets based on their location rather then a subscription model.

While geocast might seem similar to multicast in some ways, multicast-like routing will not be sufficient in a geocast environment for scalability reasons. Multicast routing algorithms are mostly designed to route packets to predefined multicast groups that are relatively static. Geocast packets on the other hand will have to be routed to a set of routers based on an arbitrary destination area that could contain several or even no routers.

Most research around geocast has been centered on mobile and ad-hoc applications [6] and mainly in the area of vehicular networking [1]. We believe that enabling geocast on an Internet-wide scale would make even more applications feasible. To reach such a deployment geocast would need to be enabled in the network layer [3]. Network layer solutions have the benefit that any application can benefit, possibly leading to applications not considered before. Application layer solutions already exist [2] but they have scalability problems [11] that might prove difficult to overcome in a global scenario. Previous proposals for Internet-wide geocast, such as GeoNode [12] relied on modified IP packets, while unicast based concepts [7] rely on partially unicasting a message to a covering router and special servers.

In our previous work we designed and evaluated a path based geographic routing algorithm. In this paper we describe our implementation of this algorithm and evaluate its performance during network convergence. In our previous work we have evaluated the link cost of the algorithm after convergence compared to a perfect shortest path tree. This previous evaluation was done in a static environment. In this paper we answer the question how our routing protocol behaves during convergence, by evaluating the performance of our geographic routing implementation in a simulated network environment.

The main contributions of this paper are: (1) validation of our geographic routing protocol by means of an implementation; (2) evaluation of the convergence speed of the protocol; (3) evaluation of the link usage and packet loss during convergence.

This remainder of this paper is structured as follows: In Section 2 we describe our routing protocol, followed by a description of our implementation in Section 3. In Section 4 we describe our evaluation approach followed by the actual results in Section 5. We conclude the paper in Section 6.

2 Geographic Routing

In this section we will shortly describe the addressing system that is the basis of our routing system followed by a high level overview of the algorithm that makes the routing decision.

2.1 Addressing

The addressing system our routing system is based on divides the world into 4 rectangles. We then divide these rectangles into increasingly smaller nested rectangles. These rectangles are numbered in such a way, that neighboring rectangles that are at the same depth but have different parents share the same number [9]. An example of this scheme applied to the world with rectangles up to a depth of 3 can be seen in Figure 1. In this figure we have highlighted an area encompassing much of northern Europe and the UK. Individually these can be addressed as 3.4.2 and 4.4.2, but taken together we can address them as [3,4].4.2.

We can map this representation to a binary format by using 4 bits per depth level, where we set the bits in the order 1,2,3,4 with a bit set to 1 indicating that rectangle is included. The example used before would translate to 0011.0001.0100. By using 4 bits per level we can combine any neighboring rectangles into the area we wish to describe. Using this representation we could use this geographic address as a destination IPv6 address. Given that we use 4 bits per level and

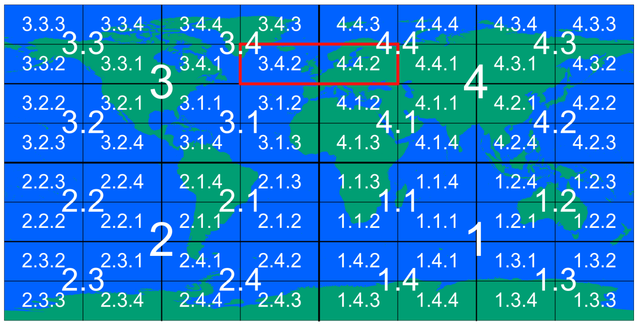


Fig. 1: Geographic addressing up to a depth of 3

assuming the first 2 bytes of the IPv6 address are needed to distinguish geocast from unicast or multicast, we are left with $112/4 = 28$ levels. This gives us a worst case rectangle size of 7 by 3.5 cm on the equator.

2.2 Routing

We have developed a geographic routing protocol on the basis of the following assumptions: All routers know the area (if any) that they service specified in the form of the addressing system described above. Destinations are addressed using the same addressing format. Routers know the (unicast) shortest path to every other router in the network. Links in the network are symmetrical.

Our geographic routing algorithm is based on path information, somewhat like BGP [13]. A complete specification of our algorithm including a discussion on the design choices made can be found in [8]. The main idea is that every router knows the shortest path to every other router and can use this information to find the next hop(s) that will lead to a (close to optimal) shortest path tree from the source to all destination routers that cover part of a packet's destination area. Packets are always forwarded over the least cost path, if there are multiple such paths the path with the lowest next hop id is chosen. This guarantees paths are chosen in a deterministic manner, which leads to shared paths to destination routers located close together.

To perform packet forwarding each router keeps track of 4 types of information essential to the routing process: 1) The two best paths it knows from every other router in the network (this corresponds to the path packets would take coming from those routers). 2) A mapping of destination routers to next hops based on our lowest next hop id rule (this corresponds to the path a packet will take going to that router). 3) The shortest path to all other routers for each of its neighbors. 4) The coverage area(s) of each router in the network.

When a router receives a packet it looks at the destination geocast address and uses its coverage table to lookup the routers who's coverage area overlap with the destination area. On a high level each router will evaluate a forwarding next hop for each destination router a packet has if this packet arrived on the

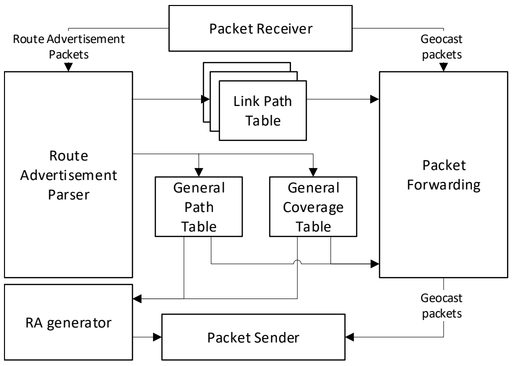


Fig. 2: Global structure of the routing software

link that has the shortest path back to the source. For every destination the router will choose a candidate next hop from its best next hop information. The router will then use the path information from the interface the candidate next hop is on to calculate the shortest path from the source to that destination as seen (advertised) by the candidate next hop. A similar calculation is made for the previous hop. If the path through the router is better or identical to the path as seen from the candidate next hop and previous hop the packet is forwarded to that next hop. If multiple destinations have the same next hop the packet is only forwarded once.

This forwarding operation might seem overly complicated, but using a more simple approach like forwarding packets on the shortest path to all destinations will lead to a form of limited flooding where packets are forwarded over multiple paths to the same destination. In our previous work [8] we have shown that simpler algorithms all show this limited flooding effect to some extent.

3 Implementation

In order to validate the proposed algorithm we have implemented the path based geographic routing algorithm and protocol for advertising paths presented in our previous work [8] and briefly described in Section 2.2. First we will present the general structure of the software followed by our information distribution, information tracking and forwarding approach.

3.1 Software structure

Our software consists of 8 main components: The Packet receiver, Advertisement parser, Packet forwarding, Link path table, General path table, Coverage table, Route advertisement generator and Packet sender. These components and their mutual relationships can be seen in Figure 2

The packet receiver handles all incoming packets. Packets that have a geocast destination address are passed to the packet forwarding system while advertisement packets are passed to the advertisement parser.

The route advertisement (RA) parser parses the advertisement packet into path data and coverage data. Path data consists of a path for each router included in the advertisement. This data is given to the per link path table and the general path table. Coverage information consists of a geographic address of the coverage area and the id of the covering router. This information is passed to the coverage table.

The per link path table is a table of all path advertisements received on a link. It is essentially the best path table of the router on the other side of the link, with the possible exception of paths that include the current router unless no other path was available.

The global path table contains the best and second best known path from all other routers in the network to a router. It also includes the best next hop for each destination. Due to our lowest id next hop choice this is not necessarily the same as the best known incoming path.

The coverage table is simply a mapping of coverage areas to the router id of the routers covering those areas. The Packet forwarding system receives geocast packets and actually runs the forwarding algorithm based on the information it receives from the coverage, path and per link tables. It gives the packet including a list of links to forward it on to the Packet sender.

The RA generator uses the information from the path and coverage table to generate a route advertisement for each link the router has. When a neighboring router is included in a path the second best known path is sent when available.

Finally the Packet sender, as its name implies, sends out all packets on the correct interface.

3.2 Route distribution

Routers periodically send route advertisements consisting of all shortest paths and coverage areas known to the router. They take the following form: Advertising router ID (1 byte), Number of advertisements in packet (1 byte) followed by the actual advertisements. A single route advertisement consists of: the coverage area (16 bytes: an IPv6 address), the path length (1 byte), the covering router id (1 byte) and the advertised path (of path length bytes, with a minimum of 1).

The time needed for every node to have at least one path to each other node is given by $t_c = t_{ra} \cdot d$ where t_c is the time needed to converge (in seconds), t_{ra} the time between route advertisements and d the diameter of the network (the maximum distance, or hops, between any two nodes).

3.3 Information Tracking

To perform its main function a routers needs to keep track of different information, mainly the shortest path to each other router it can reach and the areas these routers cover.

Based on the received coverage area and the advertising router, each router keeps a table that translates a geocast address to a set of destination routers that (partially) cover that area.

As mentioned before, each router keeps a path table for each link it has. This table tracks the best path to all other routers in the network reachable on that link. A router also keeps a global table of the best and second best path it knows to each router in the network (that it can see). The second best route is needed for the routing algorithm to function. The best route table is used to generate the route advertisement packets, unless the path contains the router it is transmitted to. If this happens the second best route is used if it exists, letting the neighbor know that there is another path. If there is no such path the best link is used, this lets the receiving router know there is no other known path except the return path to a certain other router.

3.4 Forwarding

The forwarding procedure is based on the known paths to the source and destination(s). When a geocast packet is received the router checks its destination address against its coverage table leading to a set of destination routers D. If the current router is in D it is removed as the packet already reached that destination.

For each destination d in D we evaluate the forwarding path. We combine our best paths to the source and d into a candidate forwarding path. We compare this path to a similar combination of paths reported by the candidate next hop and the previous hop (this is the main reason we keep a path table for each interface). If our candidate path is better than the other two options the packet is forwarded on this path. This approach ensures that our path is indeed the shortest path from source to destination based on our limited knowledge. Using this method we can construct close to optimal shortest path trees through a network.

4 Evaluation Approach

We want to evaluate our protocol during different convergence scenarios. In this section we will describe the metrics we are interested in, the exact scenario that will be used for the evaluation, and present overview of the tools used and our method of variable selection.

4.1 Evaluation metrics

Our evaluation is focused on protocol convergence times and the behavior of the system during convergence following a link drop or restore. We evaluate our system on the following timing values related to convergence: Initial convergence time, convergence time following a link drop, convergence time following a restored link, and packet delivery restoration time following a link drop.

We further evaluate the number of packets that are lost during network convergence. For this last metric we take into account the number of destinations that did not receive a packet. Using this method we hope to accurately represent the number of deliveries that were missed during convergence.

We will also look at the possible multi path issues during a convergence phase, in which packets are delivered to a router over multiple paths. Ideally this should never occur, but it is likely unavoidable during network convergence due to incomplete or outdated networks knowledge in different routers.

4.2 Evaluation Scenario

We start each simulation run by giving the routing algorithm time to converge. We have set this waiting time to 25 seconds as we have not observed the convergence taking longer than this time. After this initial wait we start transmitting packets from our randomly selected source to a randomly selected area. 10 seconds after the transmissions start we drop a randomly chosen link on the routing tree for those packets. We measure how long it takes the network to converge again and the effect this has on the packets that were in transit during this time. 50 seconds after the link drop we restore the link. We measure how long it takes for the network to converge again and the number of packets lost during the entire run.

4.3 Networks and Variable Selection

We use real world network graphs taken from the Topologyzoo project [4] to evaluate our protocol on a set of realistic networks. We used a total of 162 ranging in size from 6 to 51 routers, with an average of 26 routers. We import these networks into a Mininet virtual network [10] by generating a new node running our routing implementation for each Vertex in the graph and establishing a link (1 gbit/s Ethernet) between nodes if the corresponding graph has an edge.

The latitude and longitude location of the node is used to generate a coverage area of a size corresponding to depth 10 of our addressing scheme (Section 2.1) to ensure routers cover reasonable portions of a network. For each run within a network a destination area is randomly generated. This area can cover between 10% and 95% of the bounding box containing all nodes in the network. We also randomly select a source node for each run, this node can be inside or outside the destination area.

To simulate a link drop we set both ends of a link to have a 100% packet loss rate using the Linux network emulator [5] functionality. The link to drop is chosen from the set of links on the shortest path tree from the source to the destination. With this link selection we try to guarantee that the dropped link will at least have some effect on the forwarding situation. The dropped link is chosen in such a way that is does not lead to a disconnected network, so the algorithm can actually converge to the new situation and still reach all destinations. Later in the run we restore this link by returning the loss rate on both ends to 0%.

Routers are configured to exchange route advertisements every 0.5 seconds. Route entries expire after two seconds, leading to a relatively fast update time.

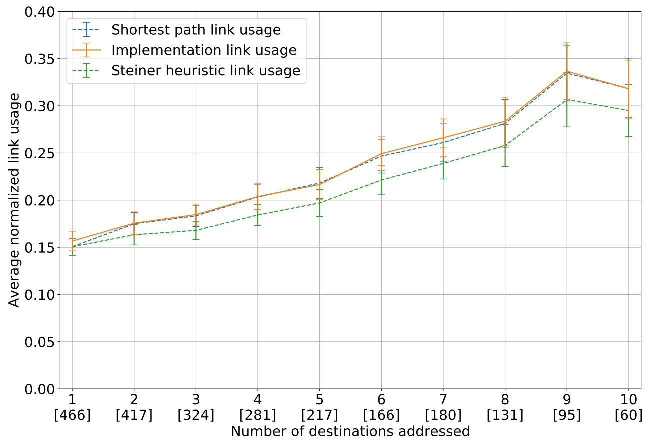


Fig. 3: Normalized link usage in a converged network

5 Evaluation

We evaluate our networks in the way described in the previous section. In this section we will present our results. We start with an overview of our algorithm's performance in a converged state and continue with the convergence time of the protocol in different network states. We end the section with the performance during different convergence situations.

5.1 General link usage

In general our algorithm establishes forwarding trees that use a comparable number of links compared to a shortest path tree. The routing stretch factor for single destinations is on average 1.046.

In Figure 3 we plot the link usage of our implementation against the shortest path tree and a Steiner tree for the same (source, destination) tuples. On the y-axis we show an average of the normalized value of the link usage, meaning the number of links used to reach all destinations divided by the number of links in the network. The x-axis shows the number of destinations addressed and below that, the number of runs with this number of destinations in brackets. The difference in number of runs is a consequence of our destination area generation system, which chooses areas with a small number of routers in geographically large networks. The error bars show the 95% confidence interval for that value. Runs with more than 10 destination were not numerous enough to provide significant results.

In our 2505 runs there were 375 runs where the algorithm established a path with a different number of links than the shortest path tree. In most of these cases this path was one link longer, and in some rare cases one shorter. The former can in some cases be explained by minor inefficiencies while most cases, and the latter difference can be explained by situations where there are multiple

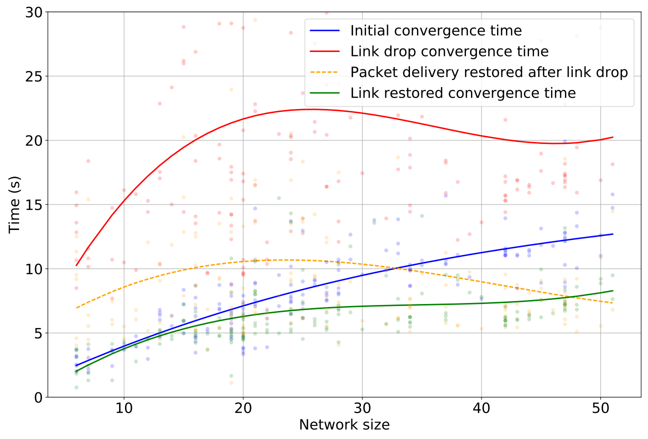


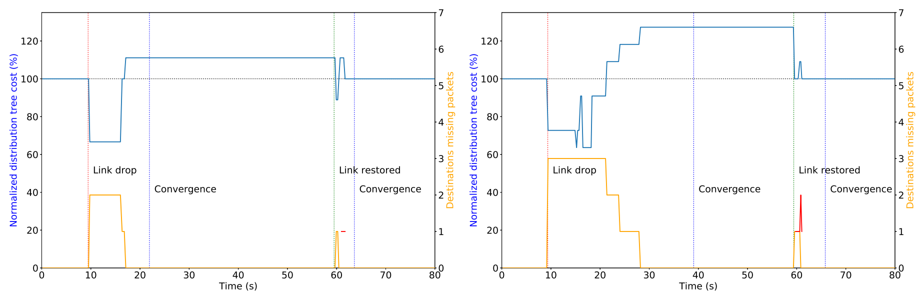
Fig. 4: Convergence times of all runs in the evaluation

shortest paths from the source to a single destination. If the chosen path overlaps with that of another destination the resulting link usage can be lower compared to the situation where the other path was taken.

We can conclude that our protocol performs mostly as expected with respect to the number of links used to construct a forwarding tree.

5.2 Convergence time

One of the more important aspects of any routing protocol is the time it takes to converge following a change in the network. The correlation between the network size (the amount of routers in the network) and the time convergence takes during the evaluation is shown in Figure 4. We can see that the initial convergence time (blue line) of the network shows a strong correlation to the network size. As a larger network mostly corresponds to more possible paths it correlates with the time it takes for the algorithm to converge. On average the convergence time after a link drop (red line) is significantly larger than that of adding or restoring a link (green line) to the network. As noted before this can be explained by the timeout of 2 seconds that needs to occur before link loss is propagated while new links are advertised with at most a 0.5 second delay. The time it takes for full packet delivery to be restored after a link drop (orange line) follows a similar pattern as the convergence time but takes less time overall. This is likely caused by the locality of the destination area, where especially in larger networks the area is contains less routers relative to the number of routers in the entire network. The most interesting thing to note in this graph is the time to convergence after a link restoration is significantly lower than the initial convergence time for larger networks. We expect this is caused by the locality of the change. In larger networks a single link drop is not likely to affect path entries further away in the network, although this does strongly depend on how well connected the network is.



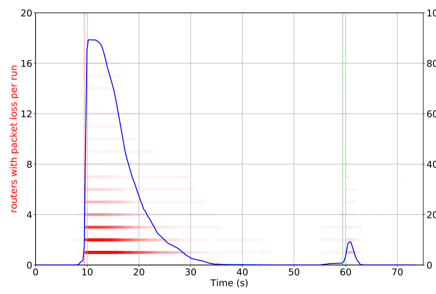
(a) Example run 1 (b) Example run 2

Fig. 5: Examples of convergence behavior on different networks

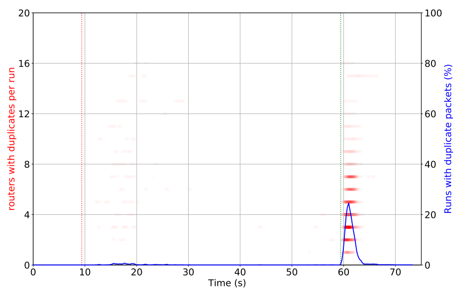
5.3 Behavior during convergence

In Figure 5 we show the convergence behavior in two different networks. On the left y-axis (blue line) we show the normalized distribution tree cost where 100% corresponds to the link usage of the initial distribution tree. On the right y-axis (yellow line) we show packet loss as the number of destination routers that did not receive a packet at a certain time. The x-axis shows the elapsed time from the start of the packet transmissions in seconds. The time of the link drop is marked as a dotted red vertical line and the link restore time with a dotted green vertical line. The dotted blue vertical lines mark the time the network had converged to the new situation. We can see that immediately following the loss of a link there is a period of packet loss, but packet delivery is restored before the network has fully converged. This can be explained by the fact that the link loss occurs on a path that is used, resulting in local convergence before the entire network has had time to converge. A similar pattern can be seen following the link restore, but the packet loss is minimal here. This is likely caused by the fact that information about new link propagates faster compared to information of lost links due to the way that timeouts function. The small red lines that can be seen following the link restoration represent packets that arrived multiple times at a router in the network. Packets are sometimes routed over multiple paths during convergence, temporarily increasing link usage as can be seen in the graphs.

As can be expected, during the convergence directly following a link drop a number of packets is not delivered to the destination routers. We plot these losses in Figure 6a. In this figure the red dots represent the number of routers that did not receive a packet in a certain simulation run at that time. The intensity of the red color corresponds to the number of runs in which this number of packets was lost for that particular time, with the barely visible dots representing a single occurrence. The blue line represents the percentage of runs in which packet loss occurred at that time. Note that while we plot the red and green dotted line to represent the link drop and restore this does not correspond to the exact drop



(a) Packet loss



(b) Duplicate packets

Fig. 6: Loss and duplicate delivery data over all simulation runs

time in all simulation runs, as can be seen by dropped packets that occur earlier than expected. This is caused by varying startup time in the emulation, mostly depending on network size. Note that packet loss mainly occurs after a link drop, and link restoration barely leads to any lost packets.

While our routing system should not route packets in a loop it does in some specific conditions route packets to a single router through multiple paths. This effect can occur during convergence when the network is temporarily in a configuration that allows a multi path situation to occur. This effect can be observed in Figures 5a and 5b as a solid red line. We can see this line corresponds to the small peak in path cost directly following the restoration of a link. We show the overall duplicate packet delivery rate in Figure 6b. In this figure the red dots represent the number of routers in a certain run that received a certain packet twice, or more times. The time represents the time at which the packet was sent from the source router. The blue line represents the percentage of runs in which a packet was received multiple times by any router. As we can see this effect mainly occurs when a link is restored (or added) to the network.

Compared to unicast routing we avoid certain issues, like the count to infinity problem, by relying only on path information. Packets are simply not forwarded anymore if a router finds itself on anything but the shortest path from its point of view. On the topic of network change we can conclude that our protocol correctly reestablishes a forwarding tree with only temporary multipath problems in some cases following a link restoration.

6 Conclusion

In this paper we have presented an implementation of a geographic routing protocol. We have shown that the protocol forwards packets along a close to optimal shortest path tree in situations where the network is stable. We have also shown that during periods where the network changes our algorithm can recover in a reasonable time. The time it takes our algorithm to recover depends on factors like network size and how well connected the network is.

During network convergence due to link failure we lose packets as can be expected, the algorithm does however recover from this state in a reasonable time. Following a link restoration there is only minimal packet loss. The protocol does deliver packets over multiple routes to a destination in this situation. While this effect is not desirable it seems limited to certain network topologies and does to a some degree prevent packet loss from occurring.

For future work we are planning to improve our information distribution method. In the current implementation coverage area and the path to the covering router are tightly linked. We would like to completely decouple these things to increase scalability. Coverage area information does not need to be advertised as often as path information, and this change could thus decrease overhead. In general our routing protocol, especially with these improvements, can provide another step in enabling Internet-wide geocast in the future.

References

1. Di Felice, M., Bedogni, L., Bononi, L.: Group communication on highways: An evaluation study of geocast protocols and applications. Ad Hoc Networks **11**(3), 818–832 (2013)
2. Fioreze, T., Heijenk, G.J.: Extending DNS to support geocasting towards VANETs: A proposal. In: VNC. pp. 271–277. IEEE (2010)
3. Karagiannis, G., Heijenk, G., Festag, A., Petrescu, A., Chaiken, A.: Internet-wide geo-networking problem statement (2013), `https://tools.ietf.org/html/draft-karagiannis-problem-statement-geonetworking-01`
4. Knight, S., Nguyen, H., Falkner, N., Bowden, R., Roughan, M.: The internet topology zoo. Selected Areas in Communications, IEEE Journal on **29**(9), 1765 –1775 (october 2011). https://doi.org/10.1109/JSAC.2011.111002
5. Linux Foundation: netem (2009), `http://www.linuxfoundation.org/collaborate/workgroups/networking/netem`
6. Liu, J., Wan, J., Wang, Q., Deng, P., Zhou, K., Qiao, Y.: A survey on position-based routing for vehicular ad hoc networks. Telecommunication Systems **62**(1), 15–30 (2016)
7. Maihöfer, C., Franz, W., Eberhardt, R.: Stored geocast. In: Kommunikation in Verteilten Systemen (KiVS). pp. 257–268. Springer (2003)
8. Meijerink, B., Baratchi, M., Heijenk, G.: A Distributed Routing Algorithm for Internet-wide Geocast (May 2018), `https://arxiv.org/abs/1805.01690`, under submission, pre-publication: arXiv:1805.01690
9. Meijerink, B., Baratchi, M., Heijenk, G.: An efficient geographical addressing scheme for the internet. In: International Conference on Wired/Wireless Internet Communication. pp. 78–90. Springer (2016)
10. Mininet Project: Mininet, `http://mininet.org`
11. Moscoviter, D., Gholibeigi, M., Meijerink, B., Kooijman, R., Krijger, P., Heijenk, G.: Improving spatial indexing and searching for location-based dns queries. In: International Conference on Wired/Wireless Internet Communication. pp. 187–198. Springer (2016)
12. Navas, J.C., Imielinski, T.: GeoCast - Geographic Addressing and Routing. In: Pap, L., Sohraby, K., Johnson, D.B., Rose, C. (eds.) MOBICOM. pp. 66–76. ACM (1997)
13. Rekhter, Y., Li, T., Hares, S.: A border gateway protocol 4 (bgp-4). Tech. rep. (2005)