An Efficient Geographical Addressing Scheme for the Internet

Bernd Meijerink(✉), Mitra Baratchi, and Geert Heijenk

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

Abstract. Geocast is a scheme which allows packets to be sent to a destination area instead of an address. This allows the addressing of any device in a specific region without further knowledge. In this paper we present an addressing mechanism that allows efficient referral to areas of arbitrary size. The binary representation of our addressing mechanism fits in an IPv6 address and can be used for route lookup with simple exclusive-or operations. We show that our addressing mechanism can be used to address areas accurately enough to be used as a mechanism to route packets close to their destination.

Keywords: Geocast · Addressing · IPv6

1 Introduction

Geocast is the concept of sending a message to a geographical location instead of a (fixed) address. All devices on a certain network that are in the specified area and are interested in the offered data can subscribe and receive this data. When combined with the use of multicast this can lead to an efficient way of addressing all nodes in a certain area.

An important application area of geocast or geographically addressed multicast, is in the area of vehicular networks [6] or mobile systems in general. There are several situations, such as the transmission of traffic information where it is unimportant to address specific nodes, but it is important that all nodes in a certain area receive a message. For example, cars on a specific road might need to know that an ambulance is approaching so that they can make room, while a car on a parallel road does not need to receive this information.

Although most of the currently proposed geocasting systems focus on ad-hoc networks or vehicular networks more specifically [1], only few have been proposed from a fixed network point of view [2,8]. The main downside of the ad-hoc model is that it only works from inside the ad-hoc network, which limits scalability. It would be beneficial to be able to send a message from anywhere in a fixed network (for example the Internet) to a specific region. One of the main

L. Mamatas et al. (Eds.): WWIC 2016, LNCS 9674, pp. 78–90, 2016.
DOI: 10.1007/978-3-319-33936-8_7

use-cases for such a system would be to target cars on public roads to inform them of locally relevant information.

In this paper, we present an addressing method that can be used to efficiently address arbitrary areas with low computational cost. The resulting address fits in the IPv6 address space and can be used as a destination address for a packet and as a route entry for routers. Route lookup can be performed based on prefix matching already used in the Internet today. We consider the following requirements for such a system based on the requirements for an Internet-wide geo-networking system [4]:

– **Accuracy.** The proposed system needs to be relatively accurate in large as well as small areas. Inaccuracy would cause messages to be delivered to locations that are not within the destination area, or in the opposite case not to be delivered at all.
– **Minimal delay.** Routers will need to be able to quickly forward packets, preferably by a system similar to currently used unicast longest prefix matching. Large tables containing complex forwarding entries should be avoided.
– **Scalability.** Scaling to a worldwide system should be possible without impacting routing performance. It should be possible to aggregate addresses to minimize the growth of route table entries.

The main contribution of this paper is proposing an addressing method in which neighbouring areas can be specified with a single address. This addressing method has a binary representation that can be used to perform route lookups based on prefix matching, avoiding the need to do costly calculation on destination polygons [9].

The structure of the rest of the paper is as follows: In Sect. 2, we present and evaluate related work done on geocasting. Section 3 describes our method to address regions. In Sect. 4, we describe how arbitrary areas can be fitted into a single address. Section 6, explains the applications of our addressing scheme in the Internet. Finally, we conclude and explore possible directions of future work in Sect. 7.

2 Related Work

A number of papers have been published about geocasting protocols in ad-hoc networks, allowing data to be sent to other location in the wireless ad-hoc network based on the location of the destination node(s). Papers describing geocasting systems for large fixed IP networks are however, rare. In this section, several past proposals will be described.

Navas and Imielinski proposed a system that adds an overlay network on top of the existing IP infrastructure to support geocasting [8]. This system requires special routers and modification of the sending and receiving nodes. In their system routers are responsible for a certain area and can forward messages to other routers, if this destination is outside of their coverage area. Addressing in this system is based on GPS coordinates. The same authors proposed a system to use a worldwide geo-network to send and receive messages from geo-networking capable devices [3]. In this paper, the authors propose to divide the world into 3 dimensional partitions with their own individual multicast address. Communication to and from these so called ‘dataspaces’ is based on multicast trees between responsible routers. This addressing approach would require a unique multicast address for each area. While this approach is certainly not impossible with the IPv6 addressing space, it would lead to a large amount of route table entries. Navas and Imielinksi have also proposed a routing algorithm that uses full network knowledge to route geographic packets [9]. In this paper, the cost of calculating the forwarding links for a packet is evaluated. It is concluded that destination areas should be simplified in forwarding routers to reduce routing time at the cost of accuracy.

GPS based Multicast is based on smallest possible sections (an atom) that can be mapped to a multicast address. Each atom and each possible partition are assigned a multicast address. A sender would have to know the address of its destination area to target it. This system allows for a geocasting system that can work in any IP multicast capable network. The main addressing disadvantage is that the target region must be predefined.

The eDNS platform is a modified DNS system that can perform reverse location queries [2]. The system can be queried for all records that are in a specified area. This system allows the implementation of an application layer geocast solution. A device first looks up all the addresses of the devices in the region it wants to geocast to. When the query returns the data can be sent to all those devices. This approach has the benefit that it can be deployed on top of the existing IP infrastructure and can work with both IPv4 and IPv6. The main downside is that every device in the target area needs to be addressed individually and all devices must periodically update their location in a central database.

To the best of our knowledge, there are no more recent papers concerning geocasting in fixed networks. More recent publications generally focus on the specific ad-hoc wireless use case, specifically on sensor networks.

Georouting protocols for wireless environments are based on several assumptions that are not necessarily true in a wired setting. In a wireless setting, georouting protocols try to be ‘greedy’, routing packets to the neighbour node nearest to the destination [5]. The most fundamental difference is that in wireless ad-hoc networks, there is a strong relation between the physical distance between two nodes and the network distance, e.g., number of hops between nodes, data rate of a link, or error probability of a link. In a fixed network this relation is only very limited. Furthermore, in a fixed network the relation between the direction a packets needs to travel in to reach its destination does not necessarily correspond to the location of the next hop router.

When traffic volume grows efficiency becomes important, routing should preferably be done in a manner resembling the current IP routing system with no routing cost depending on the destination. Another large difference is that fixed infrastructure is generally relatively static compared to an ad-hoc environments. This gives us the possibility to consider the distribution of routing information over multiple hops, or even the entire network.

Geohash [10] is a method for efficiently indexing geographical coordinates with arbitrary precession. While Geohash is not directly related to geocast routing, it is an interesting approach to look at. The geohash system splits the world into halves based on latitude and longitude and represents them in a short form. Locations near to each other share a common prefix although this does not work near the Greenwich meridian, the 180° meridian and the poles (due to the closeness of the longitude coordinates). Due to the nature of Geohash to represent nearby coordinates with identical prefixes and the ability to use arbitrary precision, we feel that Geohash is an interesting concept to explore for routable geographic addresses.

3 Addressing

In this Section, we will describe the method we use to address sections of the planet. To represent a single section on the planet we divide the Earth into four sections based on the latitude and longitude of the wgs84 projection [7]. The latitude ranging from 90 to −90 (North to South) and longitude from −180 to 180 (East to West). The resulting four rectangles (which we refer to as level 1) meeting at (0,0) are themselves divided into four sections. This process continues for the number of steps that are needed for an optimal description of the area that we are interested in. Due to the nature of the wgs84 projection, the rectangles in our system have half the height of their width in terms of degrees. Figure 1, shows an overview of these rectangles.

The major difference between our approach and GeoHash [10] is in the way we number these rectangles. As we will examine in Sect. 3.2, it is possible to combine neighbouring rectangles into a single area description. This allows us to address arbitrary areas on the globe based on these rectangles. With the ability to combine neighbouring rectangles into a single description we can resolve one of the main limitations of addressing single rectangles.

This form of specifying areas allows fast forwarding of geographically addressed packets though the Internet. This system should be accurate enough to get packets relatively close to their destination where a more accurate but slower routing method could take over.

3.1 Size of the Rectangles

Rectangles that are split into four identical sections logically have half the height and width of their parent rectangle. We define the level of a rectangle as the number of times we need to split the initial rectangle to reach it. With this definition there are 4 rectangles at level 1, 16 at level 2 and 64 at level 3 (Fig. 1). The number of rectangles in a certain level is thus given by Eq. 1. In this equation, N is the number of rectangles that are present at a certain *level*.

The resulting height and width of a rectangle at a specific level depends on the size of the earth and the location of the rectangle on it. Due to the size in meters depending on the latitude, all calculation related to size are done in degrees. Equation 2 gives us the width of a rectangle at a specific *level* where W is the width. Equation 3 gives us the height of a rectangle at a specific *level*, where H is the height.

$$N = 4^{level} \tag{1}$$

$$W(level) = \frac{360}{2^{level}} \tag{2}$$

$$H(level) = \frac{180}{2^{level}} \tag{3}$$

As noted, the actual size of a rectangle depends on the latitude of the rectangle. Assuming the earth has a circumference of 40,075 km this gives us an upper bound on the size of a rectangle as $40,075/360 = 111$ km per degree. In practice, every line of a rectangle will be smaller per degree unless one of the lines is exactly on the equator.

3.2 Numbering the Rectangles

We number the rectangles that result from the process above in a way resembling a horseshoe (see Fig. 1). We start numbering from the centre of the rectangle that the new rectangles appear in with *1*. We then move to the side with *2*. Number *3* is the rectangle below that and we move back to the inside to place number *4*.

This system can be described mathematically by using Eqs. 4–6. We input the *longitude* and *latitude* of a location in Eqs. 4 and 5 respectively, together with the *level* we want to know the number of. We now use the resulting x and y values in Eq. 6 to find the corresponding number.

To find a complete representation of a rectangle with a certain level of precession the equation will have to be used once for each level of accuracy. A formal description of the use of these equations to find a complete description of a point can be found in Eqs. 7 and 8. In Eq. 7, $S_{level}(lat, lon)$ is the number on a specific level, with L the matrix in Eq. 6. Equation 8 represents the entire description for each level with $S(lat, lon)$ being the full description. Algorithm 1 provides the pseudo-code to perform this lookup using Eqs. 2–6.

$$x(lon, level) = \left\lfloor \frac{lon + 180}{W(level)} \right\rfloor \mod 4 \tag{4}$$

$$y(lat, level) = \left\lfloor \frac{90 + lat}{H(level)} \right\rfloor \mod 4 \tag{5}$$

$$L_{x,y} = \begin{bmatrix} 3 & 4 & 4 & 3 \\ 2 & 1 & 1 & 2 \\ 2 & 1 & 1 & 2 \\ 3 & 4 & 4 & 3 \end{bmatrix} \tag{6}$$

$$S_{level}(lat, lon) = L_{x(lon,level),y(lat,level)} \tag{7}$$

$$S(lat, lon) = (S_1(lat, lon), S_2(lat, lon), ..., S_{maxLevel}(lat, lon)) \tag{8}$$

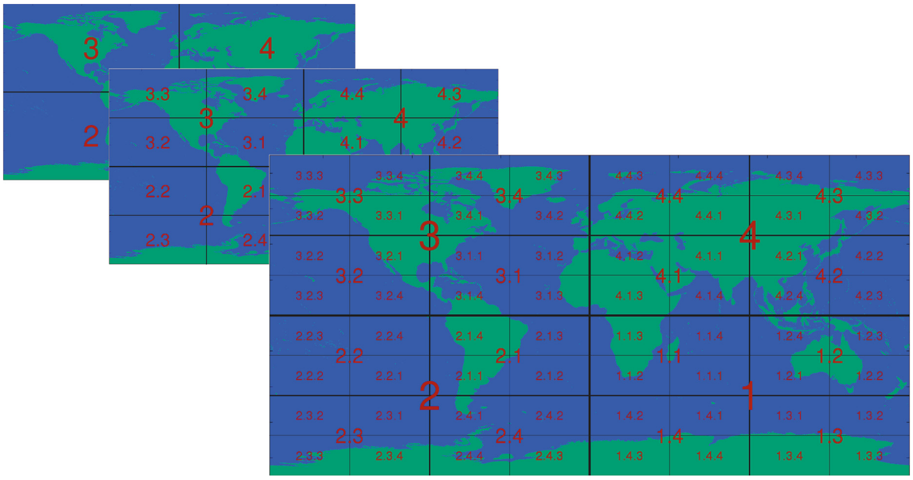


Fig. 1. Worldmap with rectangles up to level 3

The benefit of this numbering schema is that rectangles on the same level will neighbour rectangles with the same number in neighbouring parent rectangles. We will show later that this is helpful in aggregating rectangles to cover arbitrary areas.

We have chosen to represent a rectangle in this quaternary notion by separating each level with a dot. For example, a rectangle on level 2 with number *3* that has a rectangle with number *1* as its parent is represented by 1.3. Figure 1 shows rectangles with their number up to level 3 overlayed on a map of the world. Note that on each level rectangles with similar suffixes neighbour each other when their parent rectangles are neighbours.

3.3 Addressing Multiple Rectangles

Enclosing arbitrary areas in just a single rectangle would lead to a very inefficient system as it would enclose much more than just the requested area. To solve this

```
Input  : $lat$ and $lon$ (Coordinates of point); $level$; $L$ (Lookup matrix)
Output: List $result$ with rectangle representation of length $level$
List $result$;                                         // Initialize list
for $i \leftarrow 1$ to $level + 1$ do
    $x \leftarrow (lon + 180)/(360/2^{level}) \mod 4$;        // Eq. 4 and 2
    $y \leftarrow (90 + lat)/(180/2^{level}) \mod 4$;         // Eq. 5 and 3
    $number_{level} \leftarrow L[x][y]$;                      // Eq. 6 lookup
    $result$.add($number_{level}$);
end
```

Algorithm 1. Coordinate lookup algorithm

problem, it would be beneficial to address multiple lower level rectangles that together better describe the area. To do so we need a method to address multiple rectangles at once. Because the way the rectangles are numbered it is relatively easy to address neighbouring rectangles at once. Imagine an area we want to address stretching from the Netherlands to the centre of Russia. In Fig. 1, we can see that we would likely need the rectangles 4.4.2 and 4.4.1. We can now describe this in a single address as 4.4.[1,2]. An area that crosses parent rectangles, such as the one covering the Netherlands and the UK would be described as [3,4].4.2. This notation means that we want to address both rectangle 3.4.2 and 4.4.2. We describe these rectangles as having level 3. We describe the amount of levels that a rectangle diverges from a single rectangle as its depth: [3,4].4.2 has depth 3, while 4.4.[1,2] has a depth of 1. A formalised expression of combining two rectangles can be seen in Eq. 9, where $\oplus$ denotes the operation of combining two areas.

$$(S_1^A, S_2^A, ...) \oplus (S_1^B, S_2^B, ...) = (S_1^A + S_1^B, S_2^A + S_2^B, ...) \tag{9}$$

Due to the nature of this method to 'mirror' lower level areas into higher levels, the description regularly incorporates more areas than just the components it is constructed of. An example would be extending the area covering the Netherlands and the UK eastward. This would add 3.4.1 to the group leading to the area [3,4].4.[1,2], but this also includes the 4.4.1 area. However, the numbering method has been designed to avoid this as much as possible.

3.4 Binary Representation

To enable the address to be encoded into an IPv6 address and perform route lookups based on our rectangle representation, it would be beneficial to have a binary notation that enables combining rectangles. Each level represents four possible rectangles. Because we want to address destinations that possibly overlap multiple rectangles, we cannot simply represent the numbers of the rectangles in a binary fashion. We represent each level as a 4 bit block: 1 = 1000, 2 = 0100, 3 = 0010 and 4 = 0001. Example: 2.4.2 = 0100.0001.0100. This example can still be represented with two bits per level, but once we need to combine multiple rectangles the four bit system becomes needed. If for example we would want to address the region surrounding (0,0) at level 3, this would require us to address the following rectangles: 1.1.3 (1000.1000.0010), 2.1.3 (0100.1000.0010), 3.1.3 (0010.1000.0010), and 4.1.3 (0001 1000 0010), or [1,2,3,4].1.3 (1111.1000.0010). The binary representation of rectangles that can be combined is a binary OR over the individual rectangles that cover the area.

With this binary representation it becomes possible to map the rectangle addresses to IPv6 multicast addresses. If we take the 16 bit multicast prefix into account, we will be left with 112 bits for addressing. With 4 bits per level this allows us to address a total of 28 levels. At the equator 28 levels corresponds to a rectangle of 7.5 by 3.7 cm. As this is a unrealistic small area to address it is likely that fewer levels can be used, saving space for other information in the address. We will evaluate the number of levels (and thus bits) needed for realistic scenarios in Sect. 5.

4 Area Calculation

Combining neighbouring rectangles that share a parent rectangle is trivial. It is possible to simply combine the addresses of the rectangles with a binary OR as explained in Sect. 3.4. In this section, we will explain how we can find the area between arbitrary points to build a complete representation of any rectangle.

A line can be represented by two points on the same latitude or longitude. As shown in Sect. 3, we can find the representation of these points at any level. To complete our line representation, we will however also need to include the rectangles that are between the rectangles that represent our start and end points. Consider the line between the points (60,−55) 3.4.1 and (60,55) 4.4.1. As we can see in Fig. 1 the rectangles 3.4.2 and 4.4.2 are between them, to accurately represent the line we need a method to find these rectangles. To extend this idea to areas that also differ in latitude we need to also take into account the rectangles in two directions (instead of one).

We can modify Algorithm 1 to accept two coordinates and calculate the area between them. The distance of the two coordinates in the lookup matrix (Eq. 6) is calculated by subtracting the values of Eqs. 4 and 5 for both coordinates from each other per level. We can now simply 'walk' over the matrix within the calculated range for each level and add the found numbers to our value for that level. Algorithm 2 shows the procedure in pseudo code. Note that we make use of Eqs. 2–5 again. It is important that coordinate 1 is the north-western corner of the rectangle and coordinate 2 the south-eastern.

An extra check is added to the algorithm to see if the parent rectangles do not border in the East - West (*line 9*) or North - South (*line 14*) direction. When this is not the case (distance is greater or equal to 4) we need to make at least one 'loop' in that direction over the matrix in Eq. 6 to ensure we cover all rectangles between the two points at that level. If we do not do this, rectangles on that level between the points might not be included in our description.

5 Accuracy

To determine the accuracy of the system we performed an evaluation with random areas on the world map. We generated a set of 10.000 random coordinates $(lat, lon)_N, N \in [1, 10000]$. These coordinates have formed the basis of several sets of rectangles. These rectangles all have one of the generated coordinates as their north-western corner. The south-eastern corner is generated based on a random value between 0 and s. Sets $((lat, lon)_N, (lat + \Delta lat, lon + \Delta lon)_N)$ were created with randomly generated deltas starting at 0 going to $s = 0.1$, 0.5, 1, 2, 5, 10, 15 or 20°. These ranges were chosen as they would cover most realistic scenarios we can envision. As reference: on the equator 0.1° equals 11 km. Additionally, six sets were generated with separate latitude and longitude ranges to

```
Input  : (lat1,lon1),(lat2,lon2) (2 coordinates); level; L (Lookup matrix)
Output: List result with rectangle representation of length level
List $result$;                                      // Initialize list
for $i \leftarrow 1$ to $level + 1$ do
    $x1 \leftarrow (lon1 + 180)/(360/2^{level})$;          // Equation 4 and 2
    $y1 \leftarrow (90 + lat1)/(180/2^{level})$;           // Equation 5 and 3
    $x2 \leftarrow (lon2 + 180)/(360/2^{level})$;          // Equation 4 and 2
    $y2 \leftarrow (90 + lat2)/(180/2^{level})$;           // Equation 5 and 3
    $dX \leftarrow (x2 - x1) \mod 4$;
    $dY \leftarrow (y2 - y1) \mod 4$;
    if $|dX| \geq 4$ then // Are parents East-West neighbours?
        $dX \leftarrow (dX \mod 4) + 4$;
    else
        $dX \leftarrow dX \mod 4$;
    end
    if $|dY| \geq 4$ then // Are parents North-South neighbours?
        $dY \leftarrow (dY \mod 4) + 4$;
    else
        $dY \leftarrow dY \mod 4$;
    end
    $temp \leftarrow 0$;
    for $y \leftarrow y1$ to $y1 + dY + 1$ do
        for $x \leftarrow x1$ to $x1 + dX + 1$ do
            $temp \leftarrow temp \vee L[x \mod 4][y \mod 4]$;
        end
    end
    $result$.add($temp$);
end
```

Algorithm 2. Rectangle lookup algorithm

ensure wide and tall rectangles. These values were as (Δlat,Δlon) in degrees: (1,3), (3,1), (2,5), (5,2), (2,10) and (10,2). The reason is that these areas are difficult to fit into a single rectangle without sacrificing accuracy.

To calculate the accuracy we find the area covered by our generated rectangle and divide it by the area covered by the most accurate rectangle we can find in our numbering scheme. The result is a number between 0 and 1, 1 being exact coverage by the calculated rectangle.

5.1 Lookup Level Accuracy

To find the optimal level at which the system can accurately describe most areas, we calculate the accuracy at different levels for each rectangle in our test set. We start at level 1 (One or multiple rectangles of 180 by 90°), and go up to level 18 (a single level 18 rectangle would measure 152 by 76 m on the equator). From Fig. 2 we can conclude that for each rectangle size tested, level 18 is enough to provide the maximum accuracy possible.

We can also see that larger areas need less levels before the accuracy does not increase any more, compared to smaller areas that require more levels to reach

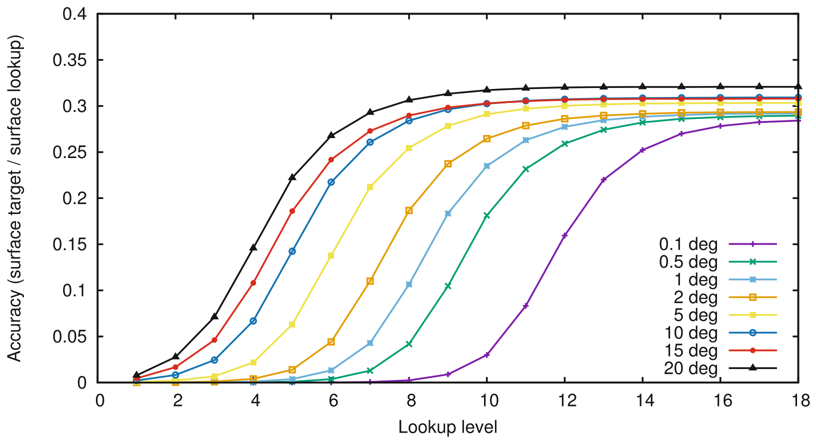
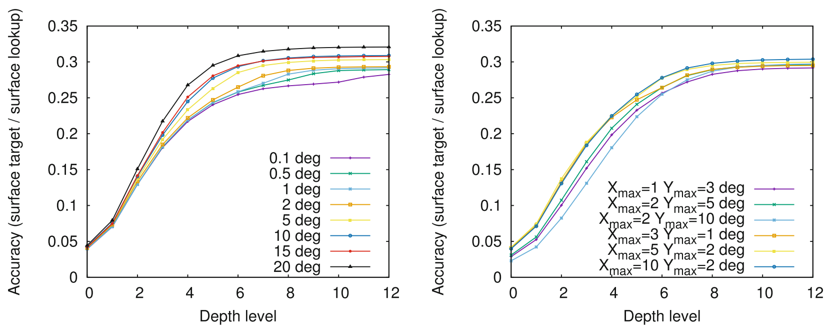


Fig. 2. Accuracy per level of different sized rectangles

the same accuracy. This is a logical result of the amount of space a rectangle of a certain level covers. We can also conclude that the maximum accuracy seems to be 0.3, meaning that the target area covers 30 % of the most accurate rectangle our system could generate.

5.2 Lookup Depth Accuracy

In Fig. 3, we examine the depth needed to accurately represent an area. We define the depth as the number of levels below the point at which the description starts covering multiple rectangles. At depth 0 the address represents a single rectangle, at depth n at most 4^n rectangles appear of that depth. Figure 3a shows that a depth of 12 is sufficient to accurately cover even small areas. In the case of larger areas the lookup can be limited to a depth of 8.



(a) Different sized rectangles (b) Tall and wide rectangles

Fig. 3. Accuracy per lookup depth

As mentioned, we have specifically looked at very tall and wide rectangles to see their accuracy. Figure 3b shows that the performance of the description for these areas is equal to that of fully random areas. It is even possible to achieve better accuracy with less depth in these cases, likely due to the fact the these areas cover many rectangles at the lower levels.

Based on Fig. 3a and b we can conclude that calculating rectangles after depth 10 does not result in much further gain. We can also see that the tall and wide areas need on average a greater depth to gain the same accuracy compared to the completely random ones. This effect is caused by the fact that these areas require more smaller rectangles to accurately represent them. Approaches like Geohash that only represent a single rectangle perform similar to depth 0 in Fig. 3a and b, significant gains can be made by addressing multiple rectangles. Accuracy is increased at least a factor 6 as compared to the single rectangle case.

6 Application to Internet-Wide Geocasting

In this section, we explore the applicability of our addressing approach. As an example we will take a packet addressed to the country of the Netherlands.

In its binary representation, our addressing scheme could be used to lookup routes for packets in the Internet. Routing could be performed based on the four bit groups that specify one or more rectangles on a given level. A router can perform a bitwise exclusive or on an address and its routing entries to find entries that have overlap.

A route entry is represented in the same way as a destination address. We take the country of The Netherlands as an example: The country covered by the rectangle with the North-West corner (3.3750933,53.6724828) and South-East corner (7.2230957,50.6266868). It is contained in the level 7 rectangle [4].[4].[2].[3].[2].[1,2,3,4].[1,4]. At level 7 these 8 rectangles represent an area 5.625° wide and tall. The binary representation of this rectangle is 0001.0001.0100.0010.0100.1111.1001.

Now consider a router that has a route entry to this exact area. Also consider a packet addressed to the northern part of the country with the following destination address: [4].[4].[2].[3].[2].[1,2,3,4].[1] or 0001.0001.0100.0010.0100.1111.1000. The router can now perform the exclusive-or operation on the address and entry to obtain the overlap:
0001.0001.0100.0010.0100.1111.1001 $\oplus$ 0001.0001.0100.0010.0100.1111.1000 = 0001.0001.0100.0010.0100.1111.1000. The resulting value has at least a single 1 in each group of four bits, so the entry is a valid forwarding route for this address.

This approach puts the burden of most calculations at the sending system. This system will have to calculate the destination address of its packets based on Algorithm 2. Routers must initially calculate (or be provided with) the rectangle description of the area they cover, but no computationally intensive operations are necessary during forwarding.

We determined in the previous section that a description of level 18 can accurately cover most areas. A rectangle of level 18 can be encoded into 72 bits.

This means that it can be used in a IPv6 multicast address while leaving 56 bits unused.

7 Conclusion and Future Work

In this paper, we have presented a method for addressing geographic regions using rectangles. Multiple rectangular regions can be addressed with a single address. This system allows areas of any size to be specified with relative accuracy. We show our approach is in some cases a factor 6 more accurate than conventional methods such as Geohash. The binary representation of our addressing method can fit within an IPv6 address with space to spare, while maintaining good accuracy. The representation can also be used for route lookup in an IP-based network. We show that our approach has potential as a system to transmit geocast packets close to their destination, where a more accurate but computationally expensive routing method can take over.

For future work we will focus on more accurately representing areas by using multiple addresses, finding a balance between multiple packets and more accuracy. For this paper, we focussed on rectangular destinations. In practice most areas will not be rectangles but other shapes, such as polygons and circles. We will extend our work to use these other shaped which will allow greater flexibility. Furthermore, we would like to focus on routing based on the presented addressing method.

Acknowledgements. The authors would like to thank the financial support provided by the SALUS project, co-funded by the EU under the 7th Framework Programme for research (grant agreement no. 313296).

References

1. Di Felice, M., Bedogni, L., Bononi, L.: Group communication on highways: an evaluation study of geocast protocols and applications. Ad Hoc Netw. **11**(3), 818–832 (2013)
2. Fioreze, T., Heijenk, G.J.: Extending the domain name system (DNS) to provide geographical addressing towards vehicular ad-hoc networks (VANETs). In: Altintas, O., Chen, W., Heijenk, G.J. (eds.) VNC, pp. 70–77. IEEE (2011)
3. Imielinski, T., Goel, S.: DataSpace - querying and monitoring deeply networked collections in physical space. In: MobiDE, pp. 44–51. ACM (1999)
4. 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
5. Karp, B., Kung, H.T.: GPSR: greedy perimeter stateless routing for wireless networks. In: Proceedings of the 6th Annual International Conference on Mobile Computing and Networking, pp. 243–254. ACM (2000)
6. Khaled, Y., Ben Jemaa, I., Tsukada, M., Ernst, T.: Application of ipv6 multicast to vanet. In: 9th International Conference on Intelligent Transport Systems Telecommunications (ITST 2009), pp. 198–202. IEEE (2009)

7. National Geospatial-Intelligence Agency: World Geodetic System (2015). https://www.nga.mil/ProductsServices/GeodesyandGeophysics/Pages/WorldGeodeticSystem.aspx. Accessed 14 January 2016
8. 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)
9. Navas, J.C., Imielinski, T.: On reducing the computational cost of Geographic Routing. Rutgers University, Department of Computer Science, Technical report DCS-TR-408 (2000)
10. Niemeyer, G.: Geohash (2008). http://geohash.org/