Efficient Forwarding of Symbolically Addressed Geocast Messages

Frank Dürr, Christian Becker, and Kurt Rothermel
Institute of Parallel and Distributed Systems (IPVS), Universität Stuttgart
Universitätsstraße 38, 70569 Stuttgart, Germany
{duerr,becker,rothermel}@informatik.uni-stuttgart.de

***Abstract*— Geocast is used to send messages to all hosts located in a geographic area. This target area can be defined either by geometric figures like polygons or by symbolic addresses like city names or room numbers.**

Geographic routing algorithms, which forward messages based on geographic information, can be used to forward geocast messages. If routing of symbolically addressed messages is based on geometric coordinates, complex mappings between symbolic addresses and their geometric extent as well as complex geometric operations are required. Therefore, we propose a routing algorithm for symbolically addressed geocast messages that operates directly on a symbolic location model. This approach does not require any geometric information for message forwarding, and forwarding decisions can be realized efficiently by comparably simple operations.

I. Introduction

Geocast is a communication mechanism that allows to send messages to all hosts located in a geographic target area. For example, geocast can be used to inform all people close to a fire to keep their windows shut, or to distribute the conference agenda and presentation slides to everyone in the conference room.

Geocast messages can be addressed either by geometric figures or by symbolic names like city names or room numbers. From a user's perspective, symbolic addressing has several advantages over geometric addressing. Since people are using symbolic addresses in everyday life, they are more intuitive to use than abstract geometric figures. Therefore, many geocast applications can be based on symbolic addressing. Consequently, we believe that symbolic addressing is an important alternative to geometric addressing.

Even if symbolic addresses are used at the geocast service interface, geocast routing can still be based on geometric addresses if symbolic addresses are mapped to geometric addresses before a message is handed over to geocast routing. However, to be able to perform this mapping we need a complex location model that associates every symbolic location with its geometric extent. A global geocast service that is also suitable for indoor usage requires a detailed three-dimensional model leading to substantial modeling effort. Moreover, with geometric routing forwarding decisions require the comparison of geometric figures, which may be rather costly operations, even if approximated geometries as proposed in [1] are used.

Therefore, we propose an alternative geocast routing approach for symbolically addressed messages purely based on symbolic location information. The most important advantage of this approach is that symbolic addresses need not be translated to geometric ones, thus, a simple symbolic location model such as proposed in [2] is sufficient. Moreover, message forwarding only requires comparably simple operations rather than geometric operations and therefore can be implemented efficiently. Additionally, our approach can be implemented without modification of the widely deployed IP infrastructure since it is implemented in an overlay network.

The remainder of the paper is structured as follows. We discuss existing geocast routing algorithms in Sec. II. In Sec. III we introduce the symbolic addressing scheme and underlying location model on which we will base our routing algorithm. Then, we propose an overlay network architecture in Sec. IV and present how the overlay network is constructed in Sec. V. The major contribution of our paper, a symbolic routing algorithm, is presented in Sec. VI. In Sec. VII we present the evaluation of our algorithm and then conclude the paper with a short summary and an outlook on future work.

II. Related Work

In this paper, we concentrate on routing algorithms for infrastructure-based systems. In [3], three classes of geocast routing approaches for infrastructure-based systems are proposed: the multicast-based approach, the directory-based approach, and geographic routing.

The *multicast-based approach* subdivides the world into partitions with unique multicast addresses. A host joins the multicast groups of all partitions including the host's position. Messages are sent by mapping target areas to multicast groups. In theory this approach could also be used for symbolically addressed messages by associating every symbolic location with a multicast address. However, for a global and fine-grained symbolic addressing scheme, many multicast addresses are necessary, which are only available in IPv6. Moreover, the set up and maintenance of large numbers of multicast trees can lead to substantial overhead for establishing multicast trees, and can result in large routing tables.

The *directory-based approach* stores the geographic areas covered by subnetworks in a directory. With this directory, the addresses of all subnetworks covering the target area can be determined. However, forwarding messages to these subnetworks using unicast is inefficient, and using multicast leads to the problems described for the multicast-based approach.

Geographic routing uses geographic information for message forwarding. A geographic routing approach based on geometric information is described in [4]. This approach forwards messages along the edges of a tree of special geocast routers, each one responsible for certain geographic areas. As already mentioned in Sec. I, geometric forwarding requires detailed geometric location models and complex operations to make forwarding decisions. A symbolic routing approach using only symbolic location information avoids these problems. [5] describes such an approach but uses a comparably simple tree-based location model. Our symbolic routing approach uses a more powerful lattice-based model as described in Sec. III and an appropriate routing algorithm.

In contrast to the infrastructure-based geocast routing algorithms described above, approaches for mobile ad hoc networks (MANETs) as described in [6] are based on fundamentally different assumptions leading to a different class of algorithms. In contrast to these approaches, we aim at a global geocast service, which is only feasible using a routing infrastructure.

III. Symbolic Addressing Scheme

In this section, we introduce a symbolic addressing scheme based on a generic symbolic location model proposed in [7].

The location model is a hierarchy defined by the spatial containment relationships between symbolic locations. The location l_1 from the set of locations L is a *descendant* of the location l_2 (denoted by $l_1 < l_2$) if l_1 is spatially contained in l_2 where l_2 is called an *ancestor* of l_1. A location is called a *child* of another location if it is a direct descendant of this location, and it is called a *parent* if it is a direct ancestor. $\mathrm{children}(l)$ and $\mathrm{parents}(l)$ denote the set of child and parent locations of l, respectively. The hierarchy forms a *lattice*, i.e., each location pair $\{l_1, l_2\}$ has a unique least upper bound (*supremum*) denoted by $l_1 \sqcup l_2$ and a unique greatest lower bound (*infimum*) $l_1 \sqcap l_2$. $l_1 \sqcap l_2$ defines the spatial intersection of l_1 and l_2, while $l_1 \sqcup l_2$ is the smallest location that contains l_1 and l_2. $\top$ denotes the *top* element "everywhere" of the lattice and $\bot$ the *bottom* element "nowhere". $\mathrm{level}(l)$ of location l is defined as $\mathrm{level}(\bot) = 0$; $\mathrm{level}(l) = \max(\mathrm{level}(l_1), \ldots, \mathrm{level}(l_n)) + 1$ with $l_1, \ldots, l_n \in \mathrm{children}(l)$ (cf. Fig 4).

Figure 1 shows a simple example of a building, where the floors and wings are both part of the building, and therefore are child locations of the location "Building". This example also contains overlapping locations, which cannot be modeled by a simple tree-based model. For example, floor 1 (l_1) and wing 1 (l_2) overlap, and the intersection is $l_1 \sqcap l_2 = l_3$.

Each location has a globally unique address, which is formed by concatenating the addresses of the locations on the path from the top of the lattice to this location. Since multiple paths may exist in a lattice, a location may end up with several addresses. For instance, room l_4 in Fig. 1 has the addresses a_1 = `/de/stuttgart/keplerstreet/5/floor1/wing1/room38` and a_2 = `/de/stuttgart/keplerstreet/5/wing1/floor1/room38`.

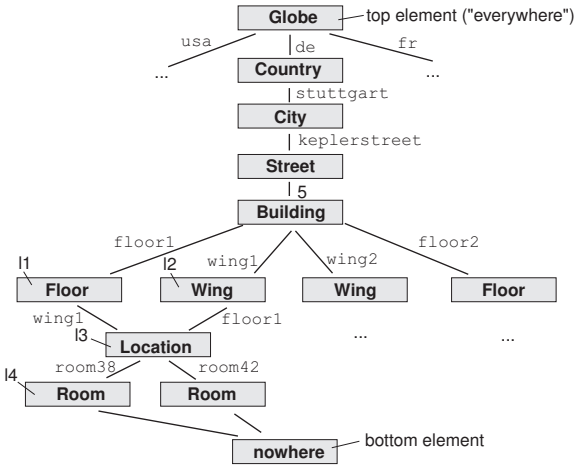

Fig. 1. Symbolic location model

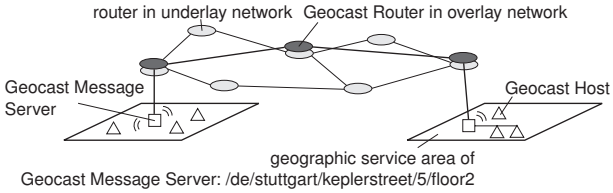

Fig. 2. Architecture

IV. Architecture

Here we present the architecture of our geocast infrastructure.

A. Components

Similar to [4] our system consists of three kinds of components (cf. Fig. 2): hosts, message servers, and routers.

A *geocast host* (GH) is a daemon running on a mobile or stationary device such as a personal digital assistant (PDA) or PC, respectively. It is responsible for sending geocast messages handed over by local applications to a geocast router for forwarding. Moreover, it delivers geocast messages received from the local message server to local applications. GHs know their current positions in form of a symbolic address from a positioning system.

Geocast Message Servers (GMS) are responsible for distributing geocast messages to all GHs within certain access networks. The geographic area covered by the access network of a GMS, say n, is called its *service area* $\mathrm{s}(n)$. This service area can be small like a single floor of a building covered by a wireless LAN, or large like the area covered by a GSM cell. Since a certain area may be covered by multiple access networks, multiple GMSs may be responsible for the same or overlapping areas. The focus of this paper is on forwarding messages by geocast routers from the sending GH to the GMSs in the target area rather than on the local distribution within an access network. Therefore, we assume that a GMS distributes received geocast messages to GHs by simply multicasting the

messages to all GHs within its attached access network. For instance, a GMS that is responsible for an access network in form of a wireless LAN covering a building forwards a received geocast message addressed to a floor of this building to all GHs residing in this wireless LAN. This can be done by forwarding the messages to a well-known IP multicast group scoped to the GMS's access network. GHs have to join this group, and mobile GHs have to ensure that they re-join the group whenever they move to another access network. Since GHs know their geographic position, they can filter messages so that only GHs withing the target area actually deliver the message to geocast applications. In the example, only GHs on the addressed floor would deliver the message. Further optimizations of this local delivery within access networks are beyond the scope of this paper. The mobility of GHs may also lead to message loss if a GH changes access networks during message transfer. The algorithms in this paper provide best effort semantics and therefore do not take counter measures to deal with lost or corrupted messages.

Geocast routers (GR) forward messages from the sending GH to all GMSs with service areas overlapping target area $\mathrm{t}(m)$ of message m. That means, to ensure the correctness of our approach, m has to be forwarded to each GMS n with $\mathrm{s}(n) \sqcap \mathrm{t}(m) \neq \bot$ since there are possibly GHs in the intersection of $\mathrm{s}(n)$ and $\mathrm{t}(m)$.

B. Overlay Network Architecture

GRs are organized in a hierarchical overlay network that resembles the lattice structure of the underlying location model as shown by the example in Fig. 3a. Each GR is associated with a service area that corresponds to a location in the location model. The service area of GR r is denoted by $\mathrm{s}(r)$. For example, $\mathrm{s}(r_3) = l_3$ in Fig. 3a. We assume that a GR's service area is configured statically, i.e., GRs are explicitly assigned to some location of the location model. The parent and child GRs of r in the router hierarchy are defined as follows:

$$\begin{aligned} \mathrm{childrouters}(r) &= \{r' \in R \mid \mathrm{s}(r') \in \mathrm{children}(\mathrm{s}(r))\} \\ \mathrm{parentrouters}(r) &= \{r' \in R \mid \mathrm{s}(r') \in \mathrm{parents}(\mathrm{s}(r))\} \end{aligned}$$

In Fig. 3 for instance, the parent GR of r_3 is r_1, and the child GRs of r_3 are r_4, r_5, and r_6.

Obviously, it is not reasonable to assign a different GR to each location. However, the description of our algorithm becomes much simpler if we assume an individual GR for each location. Therefore, we distinguish between physical and virtual GRs. While there is a one-to-one mapping between locations and virtual GRs, a physical GR can implement multiple virtual GRs. For instance, a physical GR may implement a virtual building GR as well as virtual floor GRs of this building, virtual room GRs of this building, etc. In particular, a physical GR that is configured to be the GR of a location, say l, is also implicitly assigned to each descendant location l' with $l' < l$ if no other physical GR is configured to be the GR of l'. In our example, the physical GR assigned to the building by its configuration also implements a virtual floor

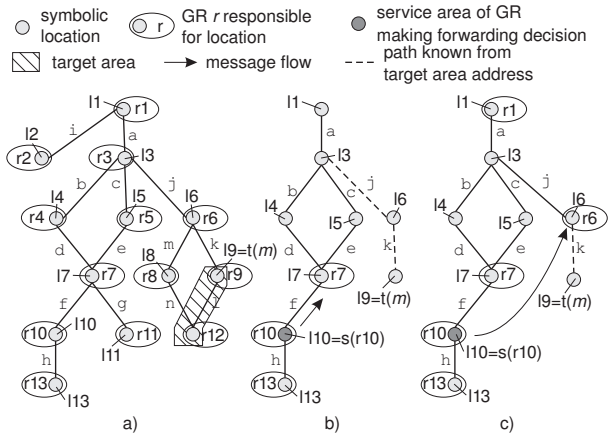

Fig. 3. Geocast Router hierarchy a) Full architectural knowledge b) Minimal architectural knowledge of GR r_{10} c) Extended architectural knowledge of GR r_{10}

GR of this building if no other physical GR is assigned to this floor by its configuration. Clearly, only messages are to be sent over the network if the sending and receiving virtual GRs are implemented by different physical GRs.

In order to make forwarding decisions, GRs must know a portion of the GR hierarchy and location model. The *minimal architectural knowledge* GR r must know is defined as follows:

- $L_r^{\min} = \{l \in L \mid l \geq \mathrm{s}(r)\} \cup \mathrm{children}(\mathrm{s}(r))$ including the symbolic addresses of these locations
- the spatial containment relation $<$ between every pair of known locations: $l_1 < l_2$ for all $l_1, l_2 \in L_r^{\min}$
- $R_r^{\min} = \mathrm{childrouters}(r) \cup \mathrm{parentrouters}(r)$ including the service areas of these GRs

The required knowledge about the location model is either configured statically or queried from a spatial model service such as the one described in [8]. In Fig. 3b for instance, the minimal architectural knowledge of GR r_{10} consists of the locations $L_{r_{10}}^{\min} = \{l_1, l_3, l_4, l_5, l_7, l_{10}, l_{13}\}$, including their symbolic addresses, the containment relationships between these locations, and the GRs $R_{r_{10}}^{\min} = \{r_7, r_{10}, r_{13}\}$ including their service areas.

Since GRs are organized in an overlay network, messages need not traverse the hierarchy, but a GR can send a message directly to any other GR. For instance, a city GR might send a message directly to another city GR without traversing a state GR. Further *extended architectural knowledge*, i.e., additionally known GRs together with their service areas and ancestor locations allows to define such *shortcuts* in the hierarchy. For instance, it might be reasonable to add the (few) country GRs to the architectural knowledge of GRs to reduce the load of the top GR. In Fig. 3c, we added location l_6 and GR r_6 to the minimal architectural knowledge of r_{10} to define a shortcut between r_{10} and r_6.

V. Overlay Network Construction

In this section we present how the overlay network is set up and how it reacts to topological changes caused by GRs

joining or leaving the network whether voluntary or by failure.

The overlay is constructed top down. That means for instance first the links from the earth GR to the country GRs are established, then links from country GRs to state GRs, etc. In order to join the overlay network, a GR, say r_{new}, must know some other bootstrap GR, say $r_{\mathrm{bootstrap}}$, that is already part of the overlay network. This GR can be configured manually. For instance, the earth or country GRs should be well-known. Alternatively, an expanding ring search or anycast on the IP layer can be used to find a bootstrap GR without knowing it a priori. With anycast, for instance, all GRs join a well-known anycast group. If r_{new} sends a message to this group, it is delivered to the topologically closest GR with respect to the IP network, and this GR is used as bootstrap GR. Via $r_{\mathrm{bootstrap}}$, r_{new} can contact its parent GRs in $\mathrm{parentrouters}(r_{\mathrm{new}})$ by sending a special registration message including $\mathrm{s}(r_{\mathrm{new}})$ to the locations in $\mathrm{parents}(\mathrm{s}(r_{\mathrm{new}}))$ using the routing algorithm described in Sec. VI. The addresses of these locations can be determined from the addresses of r_{new}'s service area by cutting of of the last part. When a GR in $\mathrm{parentrouters}(r_{\mathrm{new}})$ receives a registration message, it sends its IP address back to r_{new}. For instance, a city GR sends a registration message including its service area address to the GR of the state the city is located in, and this GR returns its IP addressed to r_{new}. Now, r_{new} knows its parent GRs and these GRs know about their new child GR and its service area. That means, these GRs have completed their minimal architectural knowledge, and r_{new} is integrated into the GR hierarchy.

In order to let the overlay network survive node and link failures and to adapt to changes of the overlay network topology caused by changes of service areas, we use a soft state approach. Every GR continuously monitors its parent and child GRs using periodically sent "heartbeat" messages from child GR, say r_c, to parent GR, say r_p, and vice versa. A successfully received heartbeat messages renews the overlay network link. If r_p has not received a heartbeat message from r_c for some time, it assumes that r_c is unavailable or has changed its service area. Therefore, r_p removes r_c from its architectural knowledge. Since now no GR is associated with r_c's service area anymore, r_p assumes to be assigned to this area as it was before r_c had joined the overlay network. If, for instance, a state GR fails and does not send heartbeat messages to its country GR anymore, the country GR also takes the role of the state GR. When r_c is up again, it reclaims to be assigned to this area and reestablishes an overlay network link between r_p and r_c as described previously.

If r_c has not received a heartbeat message from r_p for some time, it assumes that r_p is unavailable or has changed its service area. Therefore, r_c tries to find a new parent GR using the process described above. Finally, r_c registers at some available GR, say r_a, assigned to an ancestor location of its service area. If for instance a state GR fails, its city GRs eventually register at the country GR that takes the role of the failed state GR. When r_p recovers and renews its parent link with r_a by sending a registration message to r_a as described above, r_a informs r_c that it should register with r_p again. In our example, the country GR informs all city GRs that they should register with the state GR that has recovered.

Since heartbeat messages are small and can be often piggybacked onto forwarded traffic, the overhead of maintaining the hierarchy can be kept small.

Links to GRs of the extended architectural knowledge, i.e., shortcuts to non-parent and non-child GRs, can be established similarly by sending a special message to a geographic area, say l. GR r with $\mathrm{s}(r) = l$ responds with its IP address, and the sending GR can add r to its extended architectural knowledge.

In order to deliver geocast messages to access networks via GRs, GMSs have to register with GRs. This registration is described in detail in Sec. VI-C.

VI. Symbolic Routing Algorithm

Now, we present our geographic routing algorithm for symbolically addressed messages.

The routing algorithm has to guarantee that geocast message m addressed to target area $\mathrm{t}(m)$ reaches each GR $r \in R_{\mathrm{t}(m)}$ with $R_{\mathrm{t}(m)} = \{r \mid \mathrm{s}(r) \leq \mathrm{t}(m)\}$, i.e., every GR whose service area is within the target area. The GRs in $R_{\mathrm{t}(m)}$ forward m to the GMSs in the target area, which registered with these GRs.

The routing algorithm consists of two phases:

1) *Forwarding to the target area*: geocast message m is forwarded to a GR $r \in R_{\mathrm{t}(m)}$.
2) *Distribution within the target area*: starting from r, m is forwarded to all GRs in $R_{\mathrm{t}(m)}$.

Algorithm 1 shows the algorithm executed by each GR r. On receiving a geocast message, r first checks if message forwarding is in phase 2 by checking a flag in the header of the geocast message indicating the current phase (line 2). If the flag indicates phase 1, then r switches to phase 2 if its service area is completely within the target area (line 3). r then forwards the message according to the current phase (line 4,7) and additionally forwards the message to the GMSs for delivery if message forwarding is in phase 2 (line 5).

```
   procedure forward(m)
2    if phase(m) = 2 or s(r) ≤ t(m) then
       phase(m) ← 2
4      forward_phase2(m)
       forward_to_gms(m)
6    else
       forward_phase1(m)
8    fi
```

Algorithm 1. Routing algorithm executed by GR r

A. Phase 1: Forwarding to Target Area

The basic idea for phase 1 is as follows. Each GR r searches for another GR r' responsible for a location on a path leading from its service area $\mathrm{s}(r)$ to the target area $\mathrm{t}(m)$ in the hierarchy, where $\mathrm{s}(r')$ is closer to $\mathrm{t}(m)$ than $\mathrm{s}(r)$ with respect to the number of locations between the service areas and $\mathrm{t}(m)$. Consequently, m eventually reaches $r_{\mathrm{t}(m)}$ with $\mathrm{s}(r_{\mathrm{t}(m)}) = \mathrm{t}(m)$. This router in $R_{\mathrm{t}(m)}$ starts phase 2.

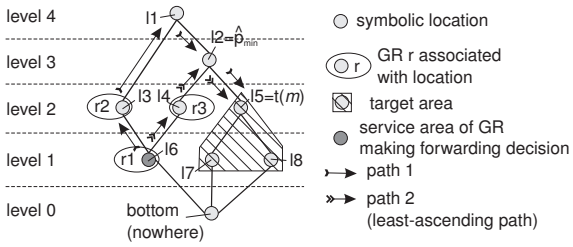

Fig. 4. Routing along least-ascending path

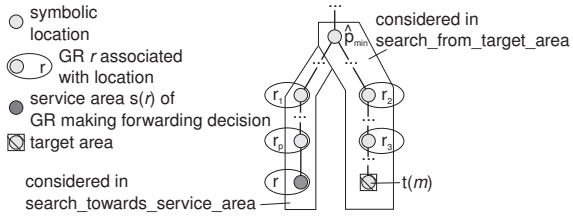

Fig. 5. Forwarding to target area

Since our location model is a lattice, possibly multiple paths from $\mathrm{s}(r)$ to $\mathrm{t}(m)$ exist. $P^{\mathrm{t}(m)}_{\mathrm{s}(r)}$ denotes the set of lattice paths from $\mathrm{s}(r)$ to $\mathrm{t}(m)$. Each path $p \in P^{\mathrm{t}(m)}_{\mathrm{s}(r)}$ has a *culmination point* $\hat{p}$, which is the location on path p on the maximum level. In the search for the next GR, r only considers GRs responsible for locations on *least-ascending paths* $\lfloor P^{\mathrm{t}(m)}_{\mathrm{s}(r)} \rfloor \subseteq P^{\mathrm{t}(m)}_{\mathrm{s}(r)}$ with

$$\forall p \in \lfloor P^{\mathrm{t}(m)}_{\mathrm{s}(r)} \rfloor, p' \in P^{\mathrm{t}(m)}_{\mathrm{s}(r)} : \mathrm{level}(\hat{p}) \leq \mathrm{level}(\hat{p}') \qquad (1)$$

That means, path $p \in P^{\mathrm{t}(m)}_{\mathrm{s}(r)}$ is least-ascending if the level of its culmination point is minimal compared to the levels of the culmination points of other paths in $P^{\mathrm{t}(m)}_{\mathrm{s}(r)}$. It follows that all least-ascending paths $\lfloor P^{\mathrm{t}(m)}_{\mathrm{s}(r)} \rfloor$ have the same *minimal culmination point* $\hat{p}_{\min}$. Since the supremum is by definition the least upper bound of two locations, it holds $\hat{p}_{\min} = \mathrm{s}(r) \sqcup \mathrm{t}(m)$. In Fig. 4, $\lfloor P^{\mathrm{t}(m)}_{\mathrm{s}(r)} \rfloor$ consists of one least-ascending path, which is path 2 leading through $l_2 = \hat{p}_{\min}$. Choosing a least-ascending path form the number of possible paths increases the scalability of the system. In the GR hierarchy, the number of GRs that exist per level decreases with increasing level. By selecting a least-ascending path, the level of the GRs involved in message forwarding is kept as low as possible, which reduces the load put on higher-level GRs.

The forwarding algorithm for phase 1 is shown in Alg. 2. The goal of this algorithm is to find a GR responsible for a location on a least-ascending path that is as close as possible to the target area location with respect to the number of intermediate locations in the lattice. The message is sent directly to this GR using the underlying network. In other words, instead of traversing all GRs responsible for locations on the least-ascending path, a shortcut is chosen if possible.

First, GR r determines the culmination point $\hat{p}_{\min}$ of the least-ascending paths $\lfloor P^{\mathrm{t}(m)}_{\mathrm{s}(r)} \rfloor$ by calculating the supremum of its service area location and the target area location (line 2). Then, r searches for a location l between $\hat{p}_{\min}$ and $\mathrm{t}(m)$ for which a GR is known. The GR closest to $\mathrm{t}(m)$ is the wanted next destination r_{next}. In Fig. 5 this is GR r_3, which is for instance closer to $\mathrm{t}(m)$ than r_2. The function `search_from_target_area` (line 3) implements this search. It starts at the target area location and searches in the direction of $\hat{p}_{\min}$ using a breadth-first strategy. This search only considers locations on paths in the hierarchy leading from $\mathrm{t}(m)$ to $\hat{p}_{\min}$.

If r cannot find a GR on a path from $\mathrm{t}(m)$ to $\hat{p}_{\min}$, then r continues searching on the least ascending paths from $\hat{p}_{\min}$ towards the service area of r by using the function `search_towards_sa` (line 5). This search also uses a breadth-first strategy to find a GR closest to $\hat{p}_{\min}$. This GR is also closest to $\mathrm{t}(m)$. In Fig. 5, this is GR r_1, which is closer to $\hat{p}_{\min}$ than r_p. Clearly, this function will always find a GR because at least one parent GR lies on a path leading from $\hat{p}_{\min}$ to $\mathrm{s}(r)$ (r_p in Fig. 5).

```
   procedure forward_phase1(m)
2    p̂min ← s(r) ⊔ t(m)
     r_next ←search_from_target_area(p̂min, t(m))
4    if r_next = ∅ then
       r_next ←search_towards_sa(p̂min, s(r))
6    fi
     sendmsgto(m, r_next)
```

Algorithm 2. Routing algorithm for phase 1 executed by GR r

To illustrate this algorithm, consider the example in Fig. 3 where a message is addressed to location l_9. Assume that GR r_{10} has the extended architectural knowledge depicted in Fig. 3c. $\hat{p}_{\min}$ is l_3. While searching from the target area l_9 towards $\hat{p}_{\min}$, r_{10} finds GR r_6, which is the GR closest to the target area on a least-ascending path from $\mathrm{s}(r_{10})$ to $\mathrm{t}(m)$. Therefore, r_{10} forwards the message directly to r_6, skipping for instance r_7. Figure 3b shows the same situation, but now r_{10} only has the minimal architectural knowledge. Since r_{10} cannot find a GR on a path from $\hat{p}_{\min} = l_3$ to $\mathrm{t}(m)$, it continues the search on all paths from $\hat{p}_{\min}$ to $\mathrm{s}(r_{10})$ and finds r_7 to which it forwards the message.

B. Phase 2: Forwarding in Target Area

In phase 2, the message is distributed within the target area by GRs whose service areas are completely within the target area of the message. Formally, message m has to be forwarded to each GR r' with $\mathrm{s}(r') \leq \mathrm{t}(m)$. We achieve this using the following algorithm: Each GR r in the target area forwards the message to all GRs in $\mathrm{parentrouters}(r)$ and GRs in $\mathrm{childrouters}(r)$, except for the one from which r received the message and those with service areas that are not completely within the target area.

C. Forwarding Messages to Geocast Message Servers

In phase 2, message m with target area $\mathrm{t}(m)$ reaches each GR r with $\mathrm{s}(r) \leq \mathrm{t}(m)$. These GRs forward m to each GMS n whose service area $\mathrm{s}(n)$ overlaps the target area since there

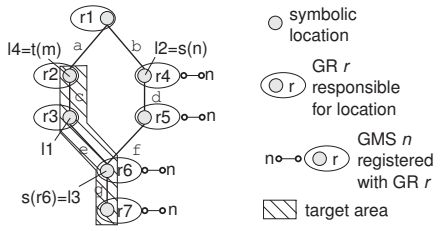

Fig. 6. Forwarding to Geocast Message Servers

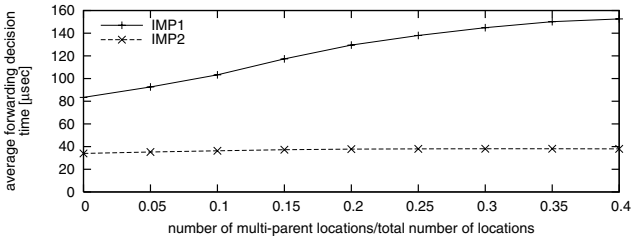

Fig. 7. Average forwarding decision time

are possibly GHs in the intersection of $\mathrm{s}(n)$ and $\mathrm{t}(m)$, and n must forward m to these GHs. In order to save bandwidth, a message should not be delivered to a GMS more than once by different GRs. The following algorithm assures this.

GMS n registers with each GR r whose service area $\mathrm{s}(r)$ is covered by n's service area $\mathrm{s}(n)$, i.e., with each GR r with $\mathrm{s}(r) \leq \mathrm{s}(n)$. Note that the total number of GMSs registered with one GR is small if we assume that the number of GMSs covering a certain location is small. Therefore, the number of GMSs a GR has to consider when a message is forwarded and the number of message copies forwarded to GMSs by a GR are also small. The number of GRs at which a GMSs has to register may be large depending on the size of $\mathrm{s}(n)$. Therefore, GMSs use the algorithm described above to send registration messages by geocast to the target area $\mathrm{t}(m) = \mathrm{s}(n)$ rather than sending it by unicast.

To make sure that message m with target area $\mathrm{t}(m)$ is forwarded to GMS n not more than once, we determine the so-called *designated router* $r_{\mathrm{t}(m),\mathrm{s}(n)}$ for each pair $(\mathrm{t}(m), \mathrm{s}(n))$. Only $r_{\mathrm{t}(m),\mathrm{s}(n)}$ forwards m to n. The algorithm is shown in Alg. 3. N denotes the set of GMSs that have registered at GR r. First, the intersection of the target area and the GMS's service area is calculated for each registered GMS in line 3. r is the designated router $r_{\mathrm{t}(m),\mathrm{s}(n)}$ if this intersection is equal to the service area of r. In this case, r forwards m to n (line 4).

```
   procedure forward_to_gms(m)
2    foreach n ∈ N do
       if t(m) ⊓ s(n) = s(r) then
4        sendmsgto_gms(m, n)
       fi
6    od
```

Algorithm 3. Forwarding messages to Geocast Message Servers

Figure 6 shows an example for the registration of GMSs and message forwarding to GMSs at designated GRs. GMS n with service area $\mathrm{s}(n) = l_2$ registers with each GR r with $\mathrm{s}(r) \leq \mathrm{s}(n)$, i.e., r_4, r_5, r_6, and r_7, by sending a registration message via geocast to address /b. Now consider a message to /a, i.e., $\mathrm{t}(m) = l_4$. The intersection of $\mathrm{s}(n)$ and $\mathrm{t}(m)$ is $\mathrm{s}(n) \sqcap \mathrm{t}(m) = l_3$. Since $\mathrm{s}(r_6)$ is l_3, r_6 is the designated GR $r_{\mathrm{t}(m),\mathrm{s}(n)}$ that delivers m to n.

VII. Performance Evaluation

In this section, we evaluate the performance of our routing algorithm by measuring the local forwarding decision time.

In order to make fast *forwarding decisions*, the lattice operations $<$ and $\sqcup$ have to be implemented efficiently. Especially phase 1 is critical since in this phase GRs have to calculate one supremum to determine $\hat{p}_{\min}$ and compare several locations using $<$. Phase 2 is comparably simple. A GR just forwards the message to all child and parent GRs in the target area. Additionally, GR r must decide in phase 2 whether to forward a message to a registered GMS n by testing for $\mathrm{s}(n) \sqcap \mathrm{t}(m) = \mathrm{s}(r)$. This infimum can be found "in practical constant time" [9] by encoding the lattice or by marking each location l with $\mathrm{s}(n) \sqcap l = \mathrm{s}(r)$ in the local lattice for each registered GMS n in a preprocessing step. Therefore, phase 2 is not critical and we focus our evaluation on phase 1.

In our performance evaluation, the location model and GR hierarchy consists of 7 levels, including the top and bottom level. Every location on levels greater one has 16 child locations. Forwarding decision times are measured at GRs on level 1 since their service areas have the most ancestors, and therefore we expect them to have the longest forwarding decision times. For the same reasons, also the target area is selected from locations on level 1.

We use two prototypical implementations for lattice operations. Implementation 1 (IMP1) navigates directly on the lattice to calculate $\sqcup$ and $<$. Implementation 2 (IMP2) encodes the locations of the lattice to bit strings with the algorithm proposed in [9] in a preprocessing step. Lattice operations are then implemented by code comparisons. We randomly select 100,000 pairs of GRs and target areas, measure the forwarding decision time at these GRs and calculate the average value. All tests are executed on a workstation with one 1.2 GHz Sun UltraSPARC III processor. For space restrictions we only present the results for GRs with minimal architectural knowledge.

Figure 7 shows the forwarding decision times for IMP1 and IMP2. Since we designed our algorithm to work on a flexible lattice-based location model that can also model overlapping locations rather than on a simple tree-based model, we vary the ratio of multi-parent locations (locations denoting intersections) to single-parent locations to see the influence on the forwarding decision time. That means, we start with a tree-shaped lattice (0% multi-parent locations) and add additional links between locations on different levels randomly in a way that the lattice properties are preserved.

We see that for IMP1 the forwarding decision time grows

with the number of multi-parent locations whereas IMP2 performs almost equally well on different lattices. IMP1 calculates $\mathrm{s}(r) \sqcup \mathrm{t}(m)$ by enumerating the set $\mathrm{ancestors}(\mathrm{s}(r)) \cap \mathrm{ancestors}(\mathrm{t}(m))$ and then selecting the location on the minimum level from this set. For the comparison $l_1 < l_2$, IMP1 tests for $l_1 \in \mathrm{ancestors}(l_2)$ and $l_2 \in \mathrm{ancestors}(l_1)$. Clearly, the number of ancestor locations influences the performance of IMP1. Since the number of ancestors grows with the number of multi-parent locations, IMP1 is slower for greater ratios of multi-parent locations to single-parent locations. For IMP2 the number of multi-parent locations influences the length of location codes. However, this only slightly influences the efficiency of code comparisons needed to calculate $\sqcup$ and $<$.

For a tree-shaped model (0% multi-parent locations) we measure forwarding decision times of 83 μs and 34 μs for IMP1 and IMP2, respectively. That means, a GR can make approximately 12,000 and 29,400 forwarding decisions per second on average using IMP1 and IMP2. In contrast to our symbolic forwarding approach geometric forwarding has to compare geometric target and service areas. On the same machine it takes 44 μs and 227 μs to compare two polygons with 4 and 20 vertices, respectively. In our scenario, a geometric GR r has to do 1 comparison in the best case (r's service area) and 17 comparisons in the worst case (16 child areas + r's service area) to make a forwarding decision. That means, for polygons with 4 vertices our algorithm is approximately 0.5 to 9 times faster for IMP1 and 1.3 to 22 times faster for IMP2; for polygons with 20 vertices our algorithm is 2.7 to 46.4 and 6.7 to 113.5 times faster. Note that a polygon with 4 vertices may only approximate a symbolic area poorly leading to misrouted messages if geometric forwarding is used. At the expense of longer forwarding decision times, a service or target area can be approximated more exactly by polygons with more vertices. With symbolic forwarding we can use the exact symbolic target area to address messages without efficiency penalty.

Regarding the number of possible forwarding decisions and the fact that a reasonable GR hierarchy has many GRs at lower levels (e.g., many street and building GRs), the load of low-level GRs is no problem. The load of high-level GRs like state or country GRs is more critical. This load will stay low as long as most senders are located geographically close to the addressed area since then most messages can be handled by GRs on lower levels and need not traverse for example a state or country GR. Like [10] we assume that the relevance of location-specific information decreases as distance increases. Therefore, we can assume that most senders address nearby areas. This makes bottlenecks at high-level GRs unlikely.

Another reasonable assumption is the existence of hotspot locations that are addressed frequently. Such hotspots may occur for instance in big cities like New York. Hotspots can be handled efficiently using the described possibility to define shortcuts in the hierarchy. Since shortcuts by-pass high-level GRs (cf. Fig. 3c), the load of these GRs is reduced further. In future work we are going to investigate strategies for the automatic installation of such shortcuts.

VIII. Summary and Future Work

We presented a routing algorithm for symbolically addressed geocast messages that is based on a symbolic location model and addressing scheme. This algorithm does not rely on any geometric information for forwarding. Consequently, a simple symbolic location model suffices, which can be set up easily compared to a complex geometric model. Since the algorithm operates on routers in an overlay network, it can be implemented without changing the existing IP router infrastructure. Scalability is assured by using a hierarchy of routers resembling the location hierarchy. In our evaluation we showed that our algorithm can be implemented efficiently, so a router can make fast forwarding decisions.

The presented symbolic routing algorithm is the first step towards a fine-grained geocast routing approach. Beside geographic attributes, we are going to integrate further object attributes like the object type into our addressing scheme in order to address groups of objects within a geographic area, e.g., all taxis in the vicinity of the central station. In order to forward such messages, geographic multicast protocols are required, which are subject of our future work.

Acknowledgments

Frank Dürr gratefully acknowledges the financial support by the Deutsche Forschungsgemeinschaft within the Center of Excellence 627 "Spatial World Models for Mobile Context-Aware Systems".

References

[1] J. C. Navas and T. Imielinski, "On reducing the computational cost of geographic routing," Rutgers University, Department of Computer Science, Tech. Rep. DCS-TR-408, Jan. 2000.

[2] B. Brumitt and S. Shafer, "Topological world modeling using semantic spaces," in *Proceedings of the Workshop on Location Modeling for Ubiquitous Computing, UbiComp 2001*, Sept. 2001, pp. 55–62.

[3] T. Imielinski and J. C. Navas, "GPS-based geographic addressing, routing, and resource discovery," *Communications of the ACM*, vol. 42, no. 4, pp. 86–92, 1999.

[4] Julio C. Navas and T. Imielinski, "Geocast – geographic addressing and routing," in *Proceedings of the Third Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom '97)*, Budapest, Hungary, Sept. 1997, pp. 66–76.

[5] J. Roth, "Semantic geocast using a self-organizing infrastructure," in *Innovative Internet Community Systems (I2CS)*, Leipzig, Germany, June 2003, pp. 216–228.

[6] C. Maihöfer, "A survey on geocast routing protocols," *IEEE Communications Surveys and Tutorials*, vol. 6, no. 2, 2004.

[7] F. Dürr and K. Rothermel, "On a location model for fine-grained geocast," in *Proceedings of the Fifth International Conference on Ubiquitous Computing (UbiComp 2003)*, Seattle, WA, Oct. 2003, pp. 18–35.

[8] D. Nicklas, M. Großmann, T. Schwarz, and S. Volz, "A model-based, open architecture for mobile, spatially aware applications," in *Proceedings of the 7th International Symposium on Spatial and Temporal Databases (SSTD 2001)*, Redondo Beach, CA, July 2001.

[9] D. D. Ganguly, C. K. Mohan, and S. Ranka, "A space and time efficient coding algorithm for lattice computations," *IEEE Transactions on Knowledge and Data Engineering*, vol. 6, no. 5, pp. 819–829, Oct. 1994.

[10] B. Xu, A. Ouksel, and O. Wolfson, "Opportunistic resource exchange in inter-vehicle ad-hoc networks," in *Proceedings of the IEEE International Conference on Mobile Data Management (MDM 2004)*, Berkeley, CA, Jan. 2004, pp. 4–12.