OverDrive: An Overlay-based Geocast Service for Smart Traffic Applications

Bernhard Heep, Martin Florian, Johann Volz, Ingmar Baumgart
Institute of Telematics, Karlsruhe Institute of Technology (KIT), 76131 Karlsruhe, Germany
Email: {heep,florian,baumgart}@kit.edu, johann.volz@gmail.com

***Abstract*—For smart traffic scenarios, communication between traffic participants is of high importance. Classical approaches (e.g. for information about congestions) employ a server-based architecture, which raises scalability and privacy concerns. In this paper, we propose *OverDrive*, a decentralized overlay-based geocast service that is applicable in smart traffic scenarios and not prone to the shortcomings of centralized designs. Information requests for points in geographic space are routed directly via traffic participants until they reach a node in the proximity of that point. In contrast to other approaches, our overlay is specifically tailored towards supporting mobile nodes—vehicles connected via cellular networks—and leverages their speed and direction for optimizing peering decisions and minimizing maintenance overhead. Exhaustive simulations in complex smart traffic scenarios show that OverDrive achieves high delivery ratios even in high mobility environments. At the same time, communication overhead is kept low, making OverDrive suitable for the use with cellular networks.**

I. Introduction

The assistance of people by computer systems in areas of their everyday lives is a topic of increasing significance. One such area is *traffic*, where the concept of *smart traffic* is becoming more prevalent. In smart traffic, an integrated computer system assists the driver with various tasks. A single car's computer system by itself can make only choices based on locally limited knowledge. Thus, communication with other traffic participants, e.g. other cars or stationary infrastructure, is crucial. Examples of smart traffic applications are dynamic route planning, the localization and reservation of charging stations for electric cars and the management of car-sharing fleets. For each of these examples, communication that involves the participants of the network at specific *locations* is a necessity. For dynamic route planning, one might need to query the congestion status along suggested routes. Charging stations might advertise their status to only these cars that are close-by. Queries for available shared cars may be flooded into an acceptable region around the requester.

All of these use cases depend on a common *geocast* communication service: a system which allows each traffic participant in the system to send messages to other participants based on their location data. This includes both forwarding a message to a *single participant* and *flooding a message to all participants* in a given area, for requesting and distributing information.

In this paper we propose the distributed geocast service *OverDrive*, which does not depend on a server-based infrastructure. Instead, the participants communicate directly over cellular networks. By establishing *overlay connections* between the participants, a logical *peer-to-peer overlay* is created that is used for routing messages. For our scenario, such an architecture has the following advantages: First, the overlay network has no single point of failure and does not require permanent availability of a well-known trusted service. Second, since the capacities of a peer-to-peer system grow with the amount of users, the approach has an inherent scalability. Finally, it avoids the aggregation of location data in a centralized database. Any scalable peer-to-peer design requires that each participant stores information about only a bounded number of its peers, thus relaxing privacy concerns. Even though several overlay protocols for geocast have been proposed recently (see section II), neither of these approaches fulfills all requirements of the smart traffic scenario: Mobility is only rarely considered, and even then, no special care is taken to reduce the required maintenance traffic.

For the evaluation of *OverDrive* we augmented the overlay framework *OverSim* with a new underlay model that mimics the properties of cellular networks and considers mobility of nodes. In order to keep this mobility model close to reality, we use real-world road data and speed limit information from *OpenStreetMap*. A large number of simulation runs was carried out to identify suitable overlay parameters. Simulation results show, that OverDrive is very scalable while maintaining a low communication overhead suitable for cellular networks. Additionally, it provides efficient flooding to geographic regions and is very robust in regards to churn and node mobility.

The remainder of this paper is structured as follows: In section II we provide an overview on related work. This is followed by the description of our *OverDrive* protocol in section III. In sections IV and V we provide details on our simulation setup and our evaluation results followed by our conclusion in section VI.

II. Related Work

Overlay-based geographic information services can be realized using structured overlay approaches. Geographic positions can be hashed and used as keys for a DHT, yielding a *locality-extended DHT*. Another approach is to use geographic positions for the generation of node identifiers, yielding *location-based nodeIds* [1], [2]. Here, nodeIds are divided into a location-dependent prefix and a random suffix. When using a DHT with this approach, it enables the storage and retrieval of location-specific content.

Additionally, several proposals for unstructured location-aware overlays exist. *GeoPeer* [3] realizes geographic routing based on a Delaunay triangulation. A mapping of each point in space to exactly one node is derived. *Globase* [4] implements a hierarchical partitioning of the geographic space into layers of non-overlapping rectangular zones managed by dedicated *superpeers*. As in other superpeer-based systems, the relocation or failure of a superpeer triggers complex operations that lead

TABLE I: Properties of related locality-aware overlays

	Considers Positions	Considers Mobility	Considers Speed and Direction	Geographic Flooding	Information Locality	No Use of Superpeers
Locality-extended DHT	-	-	-	-	-	+
Location-Based NodeIds	+	-	-	-	+	+
GeoPeer	+	-	-	+	+	+
Globase	+	-	-	+	+	-
GeoKad	+	+	-	-	+	+
Geodemlia	+	-	-	+	+	+

to a high maintenance overhead. In *GeoKad* [5], each node partitions the geographic space in logical concentric rings around its own position. A constant number of nodes from each ring are stored as neighbors. Ring radii increase linearly with the circle index. GeoKad is explicitly designed towards supporting moving nodes: Each node sends location updates to its neighbors whenever its position becomes significantly different than the one advertised last. *Geodemlia* [6] also uses rings to divide up the geographic space. However, the radii of its rings increase exponentially. Furthermore, each ring is divided into subdivisions to create a more geographically balanced neighborhood structure. Both Geodemlia and GeoKad use an iterative routing approach inspired by the key-based routing overlay Kademlia [7] even though iterative routing is evidentially slower than recursive methods [8].

The properties of the presented approaches are summarized in Table I. None of the presented approaches fulfills the requirements of a smart traffic scenario completely. Most notably, and with the exception of GeoKad, all presented approaches lack consideration for nodes that change their geographic positions. Node movement was considered neither in the design, nor in the evaluation of either GeoPeer, Globase and Geodemlia. Applying these protocols to moving nodes would thus result in drastic increases in maintenance overhead and a heavy performance decline. This is also true for the presented structured overlay approaches, that are additionally unable to realize region-based flooding due to the impossibility of realizing efficient range queries in DHTs.

GeoKad is the only existing approach that considers node movement and is able to handle it. However, it does not leverage speed and direction information for optimizing neighbor selection and minimizing update traffic. Additionally, GeoKad is based on a suboptimal iterative routing scheme, does not support flooding operations, and relies on the existence of a central node that holds global information. To our knowledge, there are also no conclusive evaluations of the communication latencies and bandwidth requirements of the GeoKad overlay. For that manner, no complete evaluation of any overlay-based geocast system in a smart traffic scenario is known to us.

III. The *OverDrive* Approach

In the following we present the OverDrive approach in detail, i.e. its fully decentralized overlay structure, maintenance procedures, and message routing.

A. Neighborhood Table: Structure and Maintenance

This section first describes the structure *OverDrive*'s overlay topology followed by a description of maintenance operations.

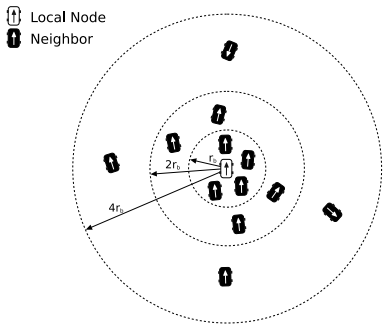


Fig. 1: A node's neighborhood table: Neighbors farther away may have higher deviations in speed and bearing.

1) Neighborhood Structure: The neighborhood structure is based on mapping neighbors to a set of k concentric rings around the position of the current node. Similar to [6], the radii of these rings start with r_b and grow exponentially with the distance from the common center. Based on the distance from the current node, neighbors fall into one of the different rings. This concept is illustrated in Fig. 1.

Each ring has an index, starting with 0, and the index i for a neighbor with distance d can be determined as follows:

$$i = \begin{cases} 0 & \text{if } d < r_b \\ j & \text{if } r_b \cdot 2^{j-1} < d \leq r_b \cdot 2^j \text{ for } 0 < j \leq k \\ k & \text{if } r_b \cdot 2^k < d \end{cases}$$

We chose exponentially growing radii in order to allow efficient long-distance routing while also maintaining extensive knowledge of nearby nodes. When routing a message, the initial hops can cover large distances via the sparsely distributed far-away neighbors. As more distance is covered, more neighbors are known in the proximity of the target, thus eventually enabling the exact geographic delivery. Extensive local knowledge is also necessary in order to ensure that flooding operations will reach a significant portion of the nodes within a desired area.

The amount of neighbors present in each ring is determined by the parameters n_{des} and n_{max}. The parameter n_{des} indicates the *desired* number of neighbors a node wishes to keep in each ring. As long as it has not reached this number, it will actively search for more nodes, a process described in III-A3. The parameter n_{max} indicates how many neighbors each ring should have *at most*. A ring that has a neighbor count between these two parameters will potentially accept neighbor connections, but will not actively search for new ones.

Nodes store node handles for each of their neighbors, together with a corresponding *geographic descriptor* including the coordinates (latitude and longitude) of the neighbor and its movement direction and speed.

2) Scoring of Neighbors: The decision whether to accept or search for new neighbors is based on a scoring function. A score is assigned to each potential neighbor based on the deviation of its *predicted future position* from that of an *ideal neighbor*. The *predicted future position* after a timespan t_{fut} is determined by the current position, the movement direction and the speed of the potential neighbor. For this prediction we

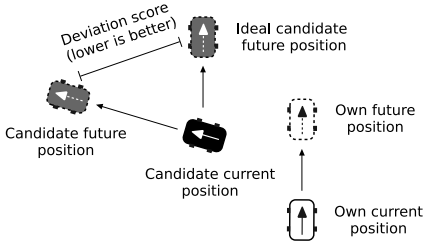


Fig. 2: The scoring function based on the deviation of the candidate and a hypothetical ideal neighbor.

assume that the potential neighbor does not change its speed or direction. The conceptual *ideal neighbor* is one that has exactly the same speed and bearing as the node itself. Ideal neighbors lead to a very stable neighborhood structure: the relative position of the neighbor to the node does not change, and therefore the neighbor always maps to the same ring. This concept is illustrated in Fig. 2.

The scoring function $score()$ for a node X can be formalized as follows (with pos_{pred} and pos_{ideal} denoting the predicted future position and the predicted ideal future position, respectively):

$$score(X) = d(pos_{pred}(X, t_{fut}), pos_{ideal}(X, t_{fut}))$$

Note that a *lower* score is a *better* one. When scoring candidates for themselves, nodes additionally consider the positions of their current neighbors. This is done to prefer the inclusion of nodes that cover sparse regions *within* a ring. Specifically, nodes calculate the average angular distance between the potential neighbor and all existing neighbors in the same ring. The average distance is normalized first to a $[0, 1]$ interval and then over the number of existing nodes in the ring. For an average angle distance of a (in radians) and $n > 1$ other neighbors in the ring, the total factor that would be applied to the score is:

$$f_{angle} = 1 + (1 - \frac{a}{\pi n})$$

Using this scoring function as it is, stationary nodes such as charging stations will likely receive low scores. We mitigate this by applying a diminishing factor to the score of stationary nodes. Since the overlay itself is agnostic of any special-purpose nodes, it asks the application whether a diminishing factor f_{bonus} should be applied.

3) Looking for New Neighbors: Each node periodically calculates its satisfaction with each of its rings. If there are less than n_{des} neighbors in a ring, or if the scores of the neighbors in the ring are below a given threshold, the respective nodes starts a search for new neighbors for this ring. There are different threshold scores depending on the ring index. This allows the toleration of higher deviations from the ideal neighbor positions in far away rings. Higher index rings are larger and it is thus less likely that a neighbor will quickly leave the area of the ring.

The search for new neighbors for a ring works by routing a *find neighbors request* to a random point within the geographic area covered by that ring. The request is answered by the node that is closest to that point. The response contains the descriptors of a subset of that node's neighbors. The subset is chosen by filling the response with nodes starting from the neighbors in its innermost ring and going outward from there. Nodes from the same ring are sorted by the score the requester would give them. The scores are calculated based on the requester's geographic descriptor included in the request.

Once the response has arrived at the originator of the request, the score for each neighbor candidate is calculated. If the current number of neighbors n for the given ring is lower than n_{des}, the node tries to add $n_{des} - n$ new neighbors to this ring. This is done by sending *neighbor connect requests* to the $n_{des} - n$ best scored neighbor candidates of the *find neighbors response*. In a second step the node additionally tries to replace existing neighbors by neighbor candidates with better scores.

4) Handling of Neighbor Connect Requests: Whenever a node X receives a *neighbor connect request* from another node Y, it needs to decide whether to accept it. If there are less than n_{max} neighbors in the appropriate ring, Y is always accepted. Otherwise, Y is only accepted if it has a better score than the currently worst scored neighbor in that ring.

5) Updating Location Data: Since our overlay is specifically designed for mobile nodes, keeping the information about neighbor locations up-to-date is a critical part of neighbor maintenance. Instead of sending *location updates* at a fixed time interval, we use a more efficient approach by taking into account the *geographic descriptors* that have been exchanged. Nodes store two pieces of information about the previous *location update*: *when* the last update message was sent, and *which* geographic descriptor was given in the message. With this information, a node can calculate what position its neighbor would predict. If the deviation between this prediction and the true position exceeds a certain threshold, a *location update* message with new geographic information is sent. Since the *location update* messages also serve as *ping* messages to verify that neighbors are still reachable, there is a maximum interval after which an update will be sent in any case.

Neighbor connections are always bidirectional—state needs to be maintained and maintenance traffic generated for each neighbor that needs a node's position data, even if that node does not require the data of the neighbor.

B. Message Routing

OverDrive uses a *recursive*, greedy routing scheme for delivering messages towards destinations in geographic space. This means that at every hop, the message is forwarded to the neighbor that is closest to the target point in geographic space. If none of the current hop's neighbors is closer to the target than itself, it assumes the responsibility for the message and processes it. To prevent routing loops between two moving nodes, which might occur if both nodes have outdated location information about the other, messages are also considered to be at their last hop if the closest node according to local neighbor location information is also the previous hop. This routing mechanism is used for both geocast services provided by OverDrive: *geographic unicast* and *geographic flooding*.

1) Unicast: Geographic unicast enables an application to send a *payload* to a node in a given *target area*. The overlay encapsulates the payload in a *geographic unicast message* and routes it to the center of the target region. On the destination node the payload is passed to the application, which may send a direct response back to the originator.

2) Flooding: Geographic flooding allows the application to broadcast a certain payload to all nodes within a certain area. To reach the flooding area a *geographic flooding message* is routed geographically to the center of the area. The first node within the destination area that receives the message is called the *initial flooder*. It will send a confirmation that the flooding has started back to the sender of the flooding request and start the actual flooding process. The initial flooder will forward the message to all of its neighbors that are within the target area. These will then forward the message to all of their neighbors, and so on. In order to avoid an infinite flooding loop, each node maintains a duplicate cache and forwards every messages once to each neighbor.

Still, naive forwarding to all neighbors at once would incur a large overhead: Since a node's neighbors, especially those in the inner rings, are very likely to be interconnected as well, a large number of duplicate messages is sent. To reduce the amount of duplicates the retransmission of messages is randomly delayed at each hop (by up to f_{wait}). If, during this delay, the message is received again from other nodes, it is not forwarded to these nodes after the delay.

C. Fault Tolerance

Especially in mobile networks, it cannot be assumed that every packet will be delivered in time, correctly, or even at all. To deal with these problems, *OverDrive* uses acknowledgements and retransmits *location updates* if no acknowledgement is received in time. At the moment, we additionally evaluate the usage of hop-by-hop acknowledgements for recursively forwarded *geographic unicast* ans *geographic flooding* messages to increase the delivery ratio further.

IV. Implementation and Simulation Model

We implemented an prototype of *OverDrive* for the overlay simulation framework *OverSim* [9]. OverDrive was realized as an *OverArch* [10] component, thus enabling the flexible combination with other modules. For example, it enables the use of *OverDrive*'s *geocast* service by other OverArch components and applications.

Since OverSim is primarily designed for non-mobile nodes using Ethernet or DSL connections, we extended OverSim's underlay abstraction by several new models: (1) a *network model*, that reflects the characteristics of data transmissions over cellular networks, (2) a *mobility model* providing geographic information (such as position, speed, and direction) and modelling node movement and (3) a *churn model*, to handle the arrival and departure of nodes at the beginning and the end of car trips.

A. Network Model

Since our scenario involves mostly moving nodes, the purpose of our network model is to reflect the characteristics of common cellular networks: (1) network latencies are dominated by the time to reach the internal IP network of the provider, (2) latencies in *LTE* networks are vastly improved over *HSPA* [11], and (3) many radio cells usually share one IP access router [12].

Our model is as follows: we divide the simulated area into hexagons, which represent areas that are accessed via the same access router. If two nodes are mapped to the same hexagon, they only receive their base latency, but no distance-related component. If they map to different hexagons, the distance between the access routers is calculated as well and contributes linearly to the bandwidth. Therefore, the total delay from the sending of data on terminal A until it reaches terminal B is the sum of three components: The radio access delay for terminal A, a distance-based backbone latency between the access routers (if applicable), and the radio access delay for terminal B. When two terminals are served by a different provider, we also add a constant latency on top of the result to simulate having to route into another backbone network. The radio access delays for the individual components are treated separately e.g. allowing only a part of the node population to work with LTE. Our network model is illustrated in Fig. 3.

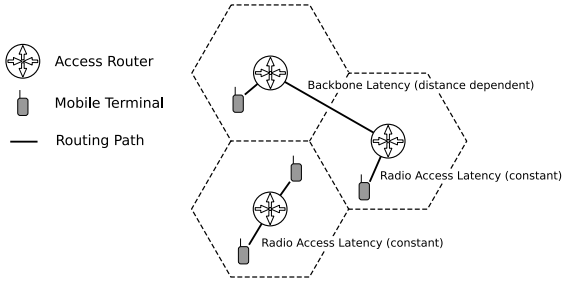


Fig. 3: Network Model: Mapping of terminals to hexagons and resulting latency calculation.

B. Mobility Model

For achieving a high degree of realism, we define following desirable properties for a mobility model for smart traffic scenarios:

1) Car movement is constrained to roads.
2) Different types of roads exist (e.g. highways vs. residential roads).
3) Cars on the same road will drive at roughly the same speed.
4) Cars have a general direction away from their origin.
5) Faster roads are generally preferred to slower ones.

Since the well-known *random waypoint model* fulfills none of our desired characteristics, we consider mobility models used for simulations in the context of vehicular ad hoc networking [13], [14]. Both works suggest using either an approach based on making a random turn at each junction or following a fastest path to a randomly selected destination. Our underlay model supports both approaches. Both require obtaining a road graph structure. For this, we extracted data from *OpenStreetMap* [15].

1) Random Turn Model: In this model, each node is placed randomly into the road network graph and is initially driving towards one of its neighbors. Whenever the node reaches a junction, a decision on where to head next is made. The probabilities for turning into a different street are assigned proportional to the speed limit of the road.

2) Pathfinding Model: Using this model, each node randomly chooses a source and destination location. Then, the *fastest* path is calculated between these two nodes, taking into account the different driving speeds on the individual roads.

The car then moves along this path until it has reached its destination. Contrary to the *random turn model*, this model exhibits all the characteristics we deemed necessary for our mobility model but comes with higher computational costs. To ensure high performance during simulations, nodes select their paths randomly from a pool of pregenerated paths.

C. Churn Model

The node churn produced by our churn model depends on the number of nodes as well as the average path length. When the pathfinding mobility model is used, the end of a path also signals that the node should be deleted. If a node has reached a one-way dead-end when using the random turn model, the trip is over. In both cases this means that the node will leave the overlay immediately and a new one will join from another position.

V. Evaluation

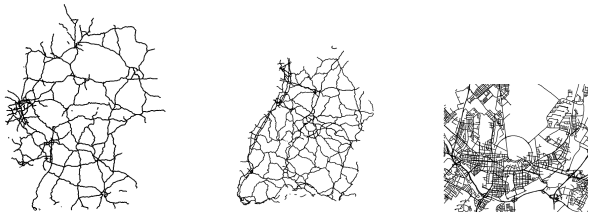
In this chapter, we present our evaluation methodology and results. Our evaluation serves two purposes: we identify optimal overlay parameters and prove that they fulfill our requirements in realistic scenarios.

A. General Experiment Setup

In this Section, parameter values are stated that are valid for all runs, except where explicitly noted otherwise. The parameters for the neighborhood structure—the base ring radius r_b, the number of desired neighbors n_{des} and the maximum number of neighbors n_{max}—are not set here, as finding optimal values for them is part of the evaluation itself.

Performing runs with nodes that run only the OverDrive overlay component allows only observations about neighborhood structure and maintenance traffic. In order to evaluate the quality of the geocast service provided by OverDrive, we thus developed an additional test application. Our test application sends *geographic unicast messages* (GUMs) to random points within the bounds of the simulated road network. Messages are sent at random intervals, which are normally distributed (with truncation) around $t_{app} = 60\,s$. There is a certain chance $p_{flood} = 1\,\%$ that the flooding of area is requested instead of sending a GUM. The radii for flooding requests and GUMs are determined by the parameters $d_{flood} = 5\,km$ and $app_{gumRad} = 2\,km$, respectively. The test applications responds to received GUMs, communicating that the routing of the message has been successful. Whenever a GUM fails, the test application verifies (using global knowledge) whether there is any node inside the target region at all.

In each simulation, we use node populations consisting of 99 % mobile and 1 % stationary nodes. Stationary nodes model stationary smart traffic entities like charging stations and do not exhibit churn behavior. For mobile nodes, we use the pathfinding mobility model with 100,000 precomputed paths. For building up the overlay, we initially insert nodes at a low rate until 200 nodes have been added. After that, nodes are inserted more quickly. We did this to be able to fill our network without requiring an excessively long warm-up time, while avoiding an overload of the few bootstrap nodes available in the early phases. The lifetime distributions of nodes are equivalent to the path driving time distribution. As we want to focus on scenarios that have a higher percentage of longer drives, we filtered out half the paths that had a shorter driving time than the total simulation time. This results in about 30 % of paths taking a shorter time to drive than all of the 4200 s of simulation time. As a default road network, we use Baden-Württemberg's major roads (Fig. 4b). Unless stated otherwise, node populations of $N = 10,000$ were used.



(a) Germany (b) Baden-Württemberg (c) Karlsruhe

Fig. 4: Different road networks used for the evaluation.

B. Evaluation Metrics

We define a set of metrics for evaluating the performance and costs of our geocast overlay. The *GUM success rate (SR_{GUM})* represents the percentage of GUMs which were successful out of those that *could* have been successful. If no node was present in the target area at the time the GUM was sent, the GUM is not counted towards the SR_{GUM}. Let m_{succ} be the number of successful responses, m_{sent} be the number of requests sent, and $m_{unavoid}$ the number of unavoidable errors. The GUM success rate is then formally defined as:

$$SR_{GUM} = \frac{m_{succ}}{m_{req} - m_{unavoid}}$$

There are two possibilities for a GUM to be unsuccessful if a node is presented in the target area: it may get dropped (in case of packet loss) or the greedy routing may not reach the target area. To avoid the first, we use end-to-end timeouts and retries. The second case is not prevented in our current design. One possible approach for future work on this aspect is to send a message across multiple paths, or the usage of hop-by-hop acknowledgement.

The key cost metric we use for the evaluation of OverDrive is bandwidth consumption, as communication in our scenario is via mobile cellular technologies like UMTS. We use two different statistical values for measuring bandwidth consumption. The first value is the *average bandwidth per node*. To prevent that certain nodes effectively become "super nodes" which carry a much larger bandwidth burden than normal nodes, we additionally look at the 99th percentile of the required bandwidth, that is, the bandwidth that is higher or equal to that of 99 % of all nodes.

For choosing suitable parametrizations for our overlay based on these metrics, we use the *performance vs. cost framework* proposed in [16]. The set of candidates found using this approach are called the *hull* of the result set. The hull includes exactly these results for which no other result offers the same or better performance at a lower cost. More formally, if a candidate c is in the hull H of a result set R with respect to the cost metric C and the performance metric P, it holds that:

$$c \in H \Leftrightarrow \forall x \in R : C(x) > C(c) \text{ or } P(x) \leq P(c)$$

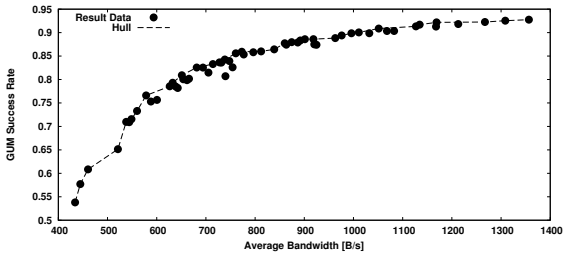


Fig. 5: The average bandwidth vs. the GUM success rate.

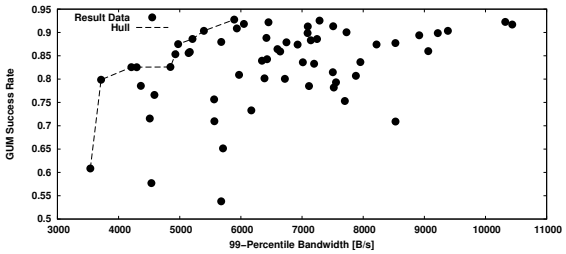


Fig. 6: The peak bandwidth vs. the GUM success rate.

C. Ring Parametrization

The most important parameters to set when configuring the overlay are the ring parametrizations—how large the rings are (r_b), how many neighbors we actively try to find for each ring (n_{des}), how many we store at most (n_{max}), as well as the number of rings (k). Through simulation, we have found suitable values for these parameters which then were used in subsequent studies.

We set the number of rings to a fixed value of $k = 8$ since even at the smallest evaluated r_b value of 1 km, each node can reach at least half of our main evaluation area (Baden-Württemberg) with its outermost ring. For all other evaluated r_b values, the entire area is virtually completely covered by the rings regardless of node position.

For all other ring parameters, we tested every combination within certain ranges, averaging over several simulation runs each. We varied the desired number of neighbors $n_{des} \in \{3, 4, 5, 6, 7\}$, the factor c used for calculating the maximum number of neighbors ($n_{max} = c \cdot n_{des}$, $c \in \{2, 3, 4, 5\}$), and the base radius $r_b \in \{2\,km, 3\,km, 4\,km\}$.

First, we examined how different parameters affect the SR_{GUM} when other parameters are fixed[1]: (1) An increasing number of desired neighbors n_{des} positively affects the SR_{GUM}, but with diminishing returns. (2) It can be very beneficial to allow for far more neighbors (n_{max}) than we actively look for. This indicates that despite the symmetrical scoring for neighbors, the situation still arises that some nodes are in need of more neighbors while their potential neighbors are already satisfied with their own number. (3) A higher r_b consistently leads to a worse SR_{GUM}, but the results vary significantly only when fixing the neighbor count parameters at very low values. This correlation is expected, as a higher base radius leads to a lower neighbor density when other parameters are fixed.

Now that we know the influence of the different parameters on the SR_{GUM}, we need to determine which combination of the tested parameters to choose. As a cost metric, we first use the average upstream bandwidth. The data points over all different parameter values, along with the hull, are depicted in Fig. 5: Most points are on or very near the hull. This indicates that none of the combinations tested are outstandingly negative, but rather that each GUM success rate value inherently comes with a given bandwidth cost that is more or less the same regardless of the parametrization through which the GUM success rate was achieved. GUM success rates initially increase rapidly when more bandwidth is expended, but there are strongly diminishing returns after roughly the 85 % mark.

[1]These results are not shown here due to a lack of space.

TABLE II: The intersection of both hulls.

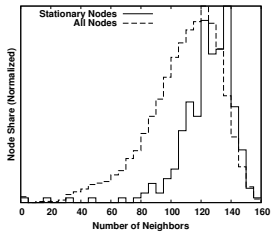
n_{des}	n_{max}	r_b	SR_{GUM}	Average Bandwidth	Peak Bandwidth
3	6	2km	60.8 %	460 B/s	3573 B/s
4	12	2km	82.6 %	681 B/s	4209 B/s
4	20	2km	88.6 %	919 B/s	5208 B/s
7	35	2km	92.8 %	1357 B/s	5890 B/s

In Fig. 6, we use the 99 % percentile of bandwidth usage to represent the peak bandwidth. Here, the situation is very different from that of the average bandwidth. Only very few points present a reasonable trade-off between bandwidth and GUM success rate, and the highest GUM success rate is reached before half the maximum peak bandwidth of the results needs to be expended.

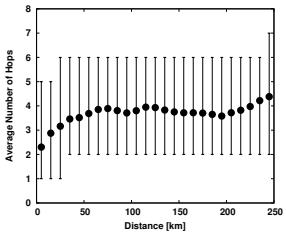
Since we want to determine a set of parameters that provides a good trade-off with respect to both cost metrics, we consider the intersection of both hulls. Formally, points in this intersection have the following property: any other point with a higher SR_{GUM} must also have *both* a higher average bandwidth cost *and* a higher peak bandwidth cost. This leaves us with only four points, which are listed in Table II. We chose the third parameter set as we prefer its higher success rate over the second option's slightly lower bandwidth requirements. Therefore, our final parametrization values are: $n_{des} = 4, n_{max} = 20, r_b = 2\,km$.

D. Traffic Categorization

In this section, we analyze how different factors contribute to the total send datarate (919 B/s) of a node using our final parametrization values. The overlay maintenance traffic clearly dominates the total traffic (896.7 B/s). This means that applications could be created that send out significantly more requests than our test application (17.5 B/s) without proportionally increasing the traffic requirements. A second observation is that the traffic required for handling other nodes' maintenance operations, i.e. for forwarding their find neighbor requests (782.2 B/s), is much higher than the traffic required for our own maintenance (114.5 B/s). Similar to the maintenance traffic, the handling of other nodes' requests requires more upstream traffic (4.7 B/s) than our own requests (1.3 B/s). The reason for both is the geographic routing: a request that needs to be sent out once gets forwarded multiple times by different hops. We also see that flooding produces a substantial amount of traffic (11.5 B/s) despite the flood chance being only 1 %.



(a) The distribution of neighbor counts.



(b) Hop count in relation to the distance to the target.

Fig. 7: Neighbor and hop counts

E. Analysis of Neighborhood Structure

In Fig. 7a we illustrate the distribution of neighbor counts. It appears that stationary nodes are roughly as well-connected as any general node. This shows that giving a bonus score to stationary nodes in the network works in practice. The overall results are also satisfactory: each node seeks out to have 80 neighbors and allows up to 160, and the vast majority of nodes fall within that range.

F. Forwarding Behavior

The purpose of *OverDrive*'s structure is to enable efficient geographic routing over both short and long distances. To analyze how it can provide this, we examine the necessary number of hops required in relation to the distance between the sender and the target point. We divided the occurring distances from 0km to 250km into intervals and calculated the average of all values in each interval. The results that are illustrated in Fig. 7b show the average hop count in each interval, as well as the 97.5 % and 2.5 % percentiles, i.e. the bar indicates the range in which 95 % of all hop counts fall. The observations are consistent with our expectations. The hop count increases sublinearly with the distance to the target point. This proves that maintaining connections to far-away nodes achieves the intended result of enabling long-distance routing with a low hop count. At the same time, target points very close-by are easily reached in fewer hops.

G. Flooding Behavior

Our primary performance metric for flooding operations is the *flood coverage* F_{cov}: When the application sends out a flooding request, it first checks via global knowledge which nodes are currently inside the area that is to be flooded. Here, n_{area} is the number of nodes in that area. When a flood is received by a node, it checks whether it was determined to be in the area at the time of the sending. If so, $n_{arearec}$ is increased. Thus, the *flood coverage* F_{cov} is defined as:

$$F_{cov} = \frac{n_{arearec}}{n_{area}}$$

Additionally, we are interested in the costs of the flooding operation. This is mainly described by the traffic required per successful receiving of the message (F_{traf}) and the number of duplicates per message receipt F_{dup}:

$$F_{traf} = \frac{b_{traf}}{m_{total}}, \quad F_{dup} = \frac{m_{dups}}{m_{total}}$$

TABLE III: Flooding evaluation with redundancy reduction.

t_{fwait}	F_{cov}	F_{traf}	F_{dup}
0	94.5 %	8.7 KB	22.8
1	95.5 %	5.6 KB	14.4
10	95.2 %	4.9 KB	12.4

TABLE IV: The results for the Karlsruhe scenario.

Mobility Model	Node Count	r_b	d_{gum}	SR_{GUM}
Pathfinding	10000	2km	2km	24.1 %
Random Turn	10000	2km	2km	97.3 %
Random Turn	1000	2km	500m	82 %
Random Turn	1000	1km	500m	94.3 %

The cost metric F_{traf} is calculated from the total number of floods received in the network m_{total} and the total flooding traffic caused by the network b_{traf}. F_{dup} is defined analogously. Note that m_{total} only counts each receipt once per message and node—duplicates do not increase this value. Instead, they are counted in m_{dups}.

To determine the effects of *delayed forwarding* on flooding redundancy we tested varying values for the t_{fwait} parameter, that indicates the time range for the forwarding delays. We tested t_{fwait} values of 0, 1 and 10 seconds. The results are listed in Table III. The flood coverage metric barely varies between the different parametrizations. This is to be expected, as the technique do not exclude nodes from getting a forwarded message. The value for the coverage—always around 95 %—shows that flooding operations are very reliable in practice. *Delayed forwarding* leads to a significant decrease in flooding duplicates and in the amount of generated traffic.

Overall, the flooding operation is feasible to use in practice due to the high flood coverage. Due to the large bandwidth cost, regulations for the use of the flooding mechanism might be required. Possible approaches here are to restrict the number of flooding operations per neighbor or to limit the allowed flooding radius.

H. Alternative Road Networks

Next, we evaluated *OverDrive* on Germany's *Autobahn* network (s. Fig. 4a). Here, the area is much larger, allowing for longer paths, and nodes move faster due to the higher speeds possible on highways. To properly compensate for the larger area, we set $N = 20,000$ and adjusted the GUM radius to 10 km for these runs. The measured SR_{GUM} of 92 % proves that the overlay works correctly in this scenario as well. However, the average traffic is much higher (1884 B/s), though the peak traffic is lower (3040 B/s) due to higher neighbor counts. The lower peak bandwidth is likely to be caused by the larger network being able to spread the maintenance load more evenly.

To experiment how *OverDrive* works in a much smaller region, we finally utilized the road network of Karlsruhe (s. Fig. 4c). As a starting point, we used the same parameters as our initial runs: $N = 10000$ with mobile nodes (99 % of the node population) following precomputed paths, sending out GUMs with a radius of 2km. For the ring parametrization, we also used the configuration we arrived at as our optimum: an r_b of 4km, n_{des} of 4, n_{max} of 20. Due to the paths generated in Karlsruhe being very short this leads to extreme churn conditions where the GUM rate is 24.1 %. In order to see

the operation of the overlay in Karlsruhe without the effects of churn, we additionally performed runs using the random turning mobility model. In order to prevent all cars from reaching a dead end, we excluded dead ends from the network. With this churn-free network, we can reach an SR_{GUM} of 97.3 % even without choosing suitable parameters for this scenario.

In addition to these runs where we mostly transferred the existing parametrizations into another road network, we used some parametrizations specific to the small network. We reduced the node count to 1000, and the GUM radius to 500m. With this, we tested *OverDrive* with base radii of $r_b \in \{1\,km, 2\,km\}$. With $r_b = 2\,km$, the SR_{GUM} is about 82 %. However, it can benefit from a smaller $r_b = 1\,km$ that was adapted to this scenario, allowing it to find more close-by neighbors. This leads to a SR_{GUM} of 94.3 %. We conclude from these results that extreme churn causes low performance of *OverDrive* in scenarios like the Karlsruhe network. However, slightly adapted ring parameters to the new network—which is smaller and more densely populated with nodes—yielded a significant increase in success. The results for all parametrizations are summarized in Table IV.

VI. Conclusion and Future Work

Efficient communication is a central element to realizing smart traffic applications on a large scale. In this paper, we thus propose *OverDrive*, an overlay-based geocast service for smart traffic applications. Our approach enables the routing of messages (e.g. traffic information requests) towards points in geographic space and the flooding of messages within geographic areas. We use an unstructured overlay approach, with routing tables based on the partitioning of geographic space into concentric rings. In contrast to related systems, OverDrive leverages the speed and movement direction of nodes, which allows for overlay operation with low maintenance traffic. Additionally, OverDrive realizes all requirements to a geocast service for smart traffic scenarios while each existing proposal fails at one or more aspects.

We evaluated OverDrive using a realistic underlay abstraction providing a mobility model for smart traffic scenarios based on actual road data from the *OpenStreetMap* project, which leads to realistic movement and churn patterns. Based on extensive simulations, we found that *geographic unicast messages* in OverDrive succeed in over 90 % of the cases and flooding operations achieve an average node coverage of 95 % in the selected area.

In future works, we plan to asses and improve the privacy-aspects of our design, minimizing the amount of private data being shared by individual nodes [17]. Additionally, we will explore the possibilities for leveraging spontaneous ad-hoc communication links for lowering the consumption of mobile data traffic.

Acknowledgment

This research was supported by the German Federal Ministry of Economics and Technology as part of the *iZEUS* project 01ME12013. The authors are responsible for the content.

References

[1] F. Hartmann and B. Heep, "Coordinate-based Routing: Refining NodeIds in Structured Peer-to-Peer Systems," in *Proceedings of the International Conference on Ultra Modern Telecommunications & Workshops (ICUMT'09)*, St. Petersburg, Russia, Oct. 2009.

[2] S. Zhou, G. R. Ganger, and P. Steenkiste, "Location-based Node IDs: Enabling Explicit Locality in DHTs," School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA, Tech. Rep. CMU-CS-03-171, Sep. 2003.

[3] F. Araújo and L. Rodrigues, "GeoPeer: A Location-aware Peer-to-Peer System," in *Proceedings of the Third IEEE International Symposium on Network Computing and Applications (IEEE NCA'04)*, Cambridge, MA, USA, Aug. 2004, pp. 39–46.

[4] A. Kovačević, N. Liebau, and R. Steinmetz, "Globase.KOM - A P2P Overlay for Fully Retrievable Location-based Search," in *Proceedings of* 7^{th} *IEEE International Conference on Peer-to-Peer Computing (IEEE P2P'07)*, Galway, Ireland, Sep. 2007, pp. 87–96.

[5] M. Picone, M. Amoretti, and F. Zanichelli, "GeoKad: A P2P Distributed Localization Protocol," in *Proceedings of the* 8^{th} *IEEE International Pervasive Computing and Communications Conference (PERCOM 2010) Workshops*, Mannheim, Germany, Mar. 2010, pp. 800–803.

[6] C. Gross, D. Stingl, B. Richerzhagen, A. Hemel, R. Steinmetz, and D. Hausheer, "Geodemlia: A Robust Peer-to-Peer Overlay Supporting Location-Based Search," in *Proceedings of the* 12^{th} *IEEE International Conference on Peer-to-Peer Computing (IEEE P2P'12)*, Tarragona, Spain, Sep. 2012, pp. 25–36.

[7] P. Maymounkov and D. Mazières, "Kademlia: A Peer-to-Peer Information System Based on the XOR Metric," in 1^{st} *International Workshop on Peer-to-Peer Systems (IPTPS 2002). Revised Papers*, vol. 2429/2002, Cambridge, MA, USA, Mar. 2002, pp. 53–65.

[8] B. Heep, "R/Kademlia: Recursive and Topology-aware Overlay Routing," in *Proceedings of 2010 Australasian Telecommunication Networks and Applications Conference (ATNAC 2010)*, Auckland, New Zealand, Nov. 2010, pp. 102–107.

[9] I. Baumgart, B. Heep, and S. Krause, "OverSim: A Flexible Overlay Network Simulation Framework," in *Proceedings of the* 10^{th} *IEEE Global Internet Symposium (GI '07) in conjunction with IEEE INFOCOM 2007*, Anchorage, AK, USA, May 2007, pp. 79–84.

[10] I. Baumgart, B. Heep, C. Hübsch, and A. Brocco, "OverArch: A common architecture for structured and unstructured overlay networks," in *Proceedings of the* 15^{th} *IEEE Global Internet Symposium in conjunction with IEEE INFOCOM 2012*, Orlando, FL, USA, Mar. 2012, pp. 19–24.

[11] C. Serrano, B. Garriga, J. Velasco, J. Urbano, S. Tenorio, and M. Sierra, "Latency in Broad-band Mobile Networks," in *Proceedings of the* 69^{th} *IEEE Vehicular Technology Conference (VTC-Spring 2009)*, Barcelona, Spain, Apr. 2009, pp. 1–7.

[12] D. Kumar, A. Miyabayashi, and O. Kari, "Availability Modelling of the 3GPP R99 Telecommunication Networks," in *Proceedings of European Safety and Reliability Conference*, Masstricht, The Netherlands, Jun. 2003, pp. 977–984.

[13] P. Mogre, M. Hollick, N. d'Heureuse, H. Heckel, T. Krop, and R. Steinmetz, "A Graph-based Simple Mobility Model," in *Proceedings of the ITG-GI Conference on Communication in Distributed Systems (KiVS 2007)*, Bern, Switzerland, Feb. 2007.

[14] D. R. Choffnes and F. E. Bustamante, "An Integrated Mobility and Traffic Model for Vehicular Wireless Networks," in *Proceedings of the* 2^{nd} *ACM International Workshop on Vehicular Ad Hoc Networks (VANET 2005)*, Cologne, Germany, Sep. 2005, pp. 69–78.

[15] M. Haklay and P. Weber, "OpenStreetMap: User-Generated Street Maps," *Pervasive Computing*, vol. 7, no. 4, pp. 12–18, Oct. 2008.

[16] J. Li, J. Stribling, R. Morris, M. F. Kaashoek, and T. M. Gil, "A performance vs. cost framework for evaluating DHT design tradeoffs under churn," in *Proceedings of the* 24^{th} *Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2005)*, Miami, FL, USA, Mar. 2005, pp. 225–236.

[17] B. Heep and I. Baumgart, "Maintenance and Privacy in Unstructured GeoCast Overlays for Smart Traffic Applications," in *Proceedings of the* 4^{th} *International Conference on Ubiquitous and Future Networks (ICUFN 2012)*, Phuket, Thailand, Jul. 2012, pp. 286–287.