Dynamic Map Update Protocol for Highly Automated Driving Vehicles

Florian Jomrich[1,2], Aakash Sharma[2], Tobias Rückelt[1,2], Daniel Burgstahler[2] and Doreen Böhnstedt[2]
[1]*Adam Opel AG, Bahnhofsplatz, 65423, Rüsselsheim, Hessen, Germany*
[2]*KOM Multimedia Communications Lab, Technical University of Darmstadt, Rundeturmstr. 10, 64283, Darmstadt, Hessen, Germany*

Keywords: Highly Automated Driving, High Definition Street Map, Map Updates, Provisioning, Context Specific, Data Efficient.

Abstract: Highly automated driving vehicles are currently subject of strong research efforts to enable novel mobility experiences. To achieve this goal a high definition street map is required. It provides the vehicles with centimetre accurate references to all geographic objects in its surrounding. So this street map enables driving capabilities of the automated vehicle in terms of safety and comfort for the passengers that could not be obtained while only relying on the cars own inbuilt sensor equipment. This high definition street map has to ensure the accuracy and timeliness of its data, necessary for the task of highly automated driving, at any time. Therefore those maps have to be constantly provided with updates from a remote server. This paper describes a protocol based mainly on preselection of contextual relevant map data to provide a car in an efficient way with such a continuous stream of updates. The capabilities of the protocol have been evaluated on a map database of Berlin. The obtained results verify that it achieves a significant decrease in transmission data and processing time, compared to existing map update approaches.

1 INTRODUCTION

With the development of highly automated driving vehicles also a new kind of navigational map a so called high definition street map is introduced. This map has got much higher requirements in terms of detail and updates compared to the currently available navigation systems (see Section 2.1 and 2.2). The reasons therefore are explained in the following.

At the current technological development status (2016) the sensor system inside an automated driving car is not able to replace completely the capabilities of a human driver (Aeberhard et al., 2015),(Barker, 2015). Various weather conditions, but also bad markings on the road, occlusion of objects through other vehicles or just high speeds are sources of potential sensor inaccuracy and false detection events (Ziegler et al., 2014). For such scenarios, a high definition street map is necessary to compensate the sensor insufficiencies (Lawton, 2015). Companies like Google (Madrigal, 2014), HERE (Stevenson, 2016), TomTom (TomTom, 2016), Continental (Hammerschmidt, 2016) and car manufacturers like BMW (Bender et al., 2014), (Aeberhard et al., 2015) and Tesla (Perkins, 2015) rely on high definition street maps for their own highly automated vehicle programs. The map itself serves as an additional "virtual" sensor in the sensor system of the car. It significantly enhances the performance and accuracy of perception and localisation algorithms, which are necessary for the vehicle to drive on its own (Jo and Sunwoo, 2014). Only with the support of the high definition street map the algorithms are able to perform well, since they can rely on its weather independent and centimetre accurate (Schumann, 2014) information as reference.

To achieve this enhanced functionality and keep the map operational it has to be constantly provided with a continuous stream of updates (Hammerschmidt, 2016). Only in this way it is possible to ensure constant validity of the maps data for the vehicles manoeuvre planning. Any obtainable information relevant to the car should be provided to it in advance through the representation as a map object (Madrigal, 2014). This could be for example major events like accidents or traffic jams as well as highly detailed geographic references like the position and orientation of lane markings, curbs and traffic lights. In this way the high definition street map changes its purpose from sole road guidance to a severe feature (Boensch,

Jomrich, F., Sharma, A., Rückelt, T., Burgstahler, D. and Böhnstedt, D.
Dynamic Map Update Protocol for Highly Automated Driving Vehicles.
DOI: 10.5220/0006279800680078
In *Proceedings of the 3rd International Conference on Vehicle Technology and Intelligent Transport Systems (VEHITS 2017)*, pages 68-78
ISBN: 978-989-758-242-4

2016) to ensure safety and comfort for the passengers inside the car.

To conclude, the high definition street maps necessary for highly automated driving require an amount of data to describe their highly detailed information, which is increased by several factors compared to currently existing navigational maps. The timely constraints regarding the contained information further require a constant stream of map updates. This makes it necessary to think about new approaches to further decrease the amount of data, which has to be transmitted to the highly automated driving vehicles. Existing map update concepts as presented in Section 2 have been developed with a human driver and a normal navigational map in mind. As explained the car as an automated machine has got different and more challenging requirements. Therefore those approaches are not sufficient enough to fulfill those requirements while keeping the operational costs of the high definition street map in a near minimum range. Our approach, the Dynamic Map Update Protocol, presented in this work aims to solve this problem by investigating the contextual relevance of map updates for a vehicle.

The outline of the remaining paper is described in the following. In Section 2 current navigation systems and already existing map update concepts are referenced. Afterwards the general idea of our approach and the concept in its details is explained in Section 3. In Section 4 the evaluation scenario of the city of Berlin to test the capabilities of our protocol is described and the obtained results are stated. We conclude the paper with a short summary and an outlook regarding future work.

2 RELATED WORK

The development of navigation systems for cars started more than 25 years ago (Ishikawa et al., 1991) and nowadays they are a widely used equipment. Most of these systems can be grouped into two different categories in terms of how they obtain the map information for routing purposes.

There are the so called offline navigation systems that use map information from a data storage built into the device itself. Examples for such systems are fully integrated devices offered directly by the car-manufactures or specific mobile navigation devices provided by companies like Garmin [1] and TomTom [2].

[1] www.garmin.com

[2] www.tomtom.com

On the other hand, there are the online navigation systems mostly represented by smart phone applications like Google Maps [3]. Such systems are storing their data on a distinct internet server, which is containing the newest map material data. Each time a route is requested by the user, the required route calculation is performed on the server. The associated map material and the guidance instructions are subsequently downloaded on the device over a wireless data connection.

Both approaches have got specific advantages and disadvantages when compared to each other as explained in the following.

2.1 Offline Navigation Systems

The biggest advantage of an offline navigation system is the fact that it can operate completely independent from any kind of data connection. The whole navigation map material is stored and processed completely within the device. To store as much information as possible and to perform route calculations in shorter time, proprietary binary storage formats are used for the representation. This however renders the system incapable of introducing updates to the initially stored dataset as stated by Min et al.(Min, 2011). When an updated map version is available, the complete map material has to be exchanged, even though there might only be some slight changes to the map itself. This often results in the copying of several gigabytes of data. The update procedure therefore has to be performed by the user himself and takes a long time to complete it.

However, if an offline navigation system is never updated its routing performance will degrade over time, since the overall structure of the street network, the base of the navigation system, is subject to change (Ling, 2013).

Asahara et al. (Asahara et al., 2008) describe in their work that the map material of Japan, which they used for research on updates for normal street maps, had to fill in 70 Megabyte of new roadside information over a period of 4 months. Mapping and navigation company HERE [4] as well states that their own worldwide map database receives 2.7 million updates every day, leading to huge efforts in keeping the map material up-to-date (Plack, 2013). Current offline navigation systems are incapable of responding to such quick changes, because the update cycle of the

[3] https://play.google.com/store/apps/details?id=com.google.android.apps.maps&hl=de

[4] https://here.com/en

mapping companies (e.g. TomTom [5]) is normally in the range of months.

2.2 Online Navigation Systems

Otherwise online navigation systems obtain up-to-date map material over a wireless connection (e.g. a smartphones cellular connection), each time the user requests information about a new route. The map server, which provides this information, gets itself constantly updated through information provided by local authorities and other related information sources. This advantage of always new data however is also a great disadvantage of online navigation systems. They often do not have their own storage space to save the obtained map data for future reuse. Each time a route is being calculated the system has to obtain the full map material for the specified area again. This might lead to unnecessary redundancy in the data transmission, which is directly connected to increased costs for the cellular connection. It also might happen that a data connection is not available because of the users current position. In such a case, online navigation systems cannot be used at all, since a route calculation is not possible.

2.3 Provisioning of Incremental Map Updates

To overcome the problems of current online and offline navigation system approaches, as mentioned in Section 2.1 and 2.2, versatile research has been conducted in the field of map representation and over the air map updates. Several different approaches providing so called incremental map updates have been published (Cooper and Peled, 2001), (Bastiaensen and others, 2003), (Min et al., 2008), (Asahara et al., 2008), (Liu et al., 2010), (Min, 2011), (Lee and Lee, 2013).

The common idea presented by all these publications is that a navigation system (equipped with its own storage space for map data) only receives incremental updates of the initial map via over the air updates. Only the changes between the old map version of the car and the newly available version have to be transmitted. This required update information is provided through a dedicated map server. The server itself has to keep track of the history of changes that have been applied to its own database. When informed about the current database version of the requesting car, the server then can generate individual updates for it. Obviously this approach reduces the necessary data volume and update time tremendously in comparison to the before mentioned common offline and online approaches.

Min et al. (Min, 2011) develop and implement such a database management system (DBMS) that sends incremental map updates through a wireless connection to a mobile device.

To be able to calculate the incremental map updates the map itself is divided into smaller parts through a grid. This common indexing approach divides the initial map into sub parts. Each sub part is than treated as a self maintained individual smaller map, a so called map tile. These tiles are treated as individual and independent sectors regarding updates.

2.4 Ensuring Consistency between Map Updates

To rely on this segmentation and to be able to build up a reliable version history of all the map tiles, different concepts exist and further requirements have to be satisfied. For example it might be the case that through an update of a single map tile the streets, which are going through this tile, are losing their previous connections to the streets of the surrounding map tiles. In this situation a proper path planning for a vehicle is not possible any more. Asahara et al. (Asahara et al., 2008) provide a solution to this problem. They propose an updating concept that keeps track of the connections of roadways. The proposed concept therefore updates the neighbouring map tiles as well, if direct connections are effected through the initial update. Through this approach it is ensured that the map material is always consistent and correctly routeable. Asahara et al. also stated that these intelligent updates should only be applied locally to keep the amount of update data which has to be provided via an over the air connection as low as possible. Hence they propose an area of 20 km to be updated in the surrounding of the car as a good compromise between update consistency and amount of data which has to be transferred.

Members of Hitachi Automotive Systems, Ltd. (Hitachi, 2016) claim that the general approach of updating individual map tiles as proposed by Asahara et al. leads to unwanted ”updating cascades”. This means if one map tile gets updated the neighbouring tiles also have to be updated to ensure the routing capability of the map. Then the cascades lead to unnecessary control data traffic over the network for the exchange of further map updates. To solve this problem they propose a different map update approach. Instead of updating specific map tiles, the updates are provided through individual map update ob-

[5]http://uk.support.tomtom.com/app/content/id/9/locale/en_-gb/page/4

jects. They consist of all the changes that have to be applied to a certain, connected area of the map. Thus no individual updates of all the affected map tiles are necessary. The amount of data needed to store the update information is also reduced.

2.5 Map Updates Along a Specific Route

Min et al., Asahara et al. and also the approach of Hitachi do not directly take the calculated route of the car into consideration for requesting the map updates. They just describe the necessity of updating the map in general.

An approach, which does in this way for the update process, is described by Bastiaensen et al. (Bastiaensen and others, 2003) as result of the discussions made in the ActMap-Project. The basic idea described in the paper, is to update only the map tiles, which are crossed by the calculated route of the car. Bastiaensen therefore describes the example of a car driving through the city area of Bruessel. Consequently, the car is only interested in updates regarding the city of Bruessel, but not the whole country of Belgium.

2.6 Foundation for the Dynamic Map Update Protocol

Besides the ActMap Project (Bastiaensen and others, 2003) none of the mentioned papers in Section 2 addresses the requirements of the scenario of highly automated driving cars. All presented solutions have been developed to update normal navigation maps, which are currently in use. The general concepts thereby can also be applied on high definition street maps. However, the requirements and specifications of high definition street maps as explained in Section 1 make it necessary to develop more sophisticated solutions to provide map updates.

One of the first approaches, which focuses on advanced driver assistance systems and has been standardized by the European Telecommunications Standards Institute (ETSI), is the so called Local Dynamic Map (ETSI, 2011). The ETSI however states that the design specification does not include a high definition street map for navigation purposes. The representation and the provisioning of such a map is seen as area of further research.

Our approach as presented in Section 3 was built on the existing concepts and provides a solution specifically designed for these new requirements.

3 CONCEPT OF A DYNAMIC MAP UPDATE PROTOCOL

The fundamental idea for our Dynamic Map Update Protocol, which is presented below, is based on the fact that the autonomous car has to be well aware of its current driving course through a navigation system. The protocol uses this fact to further decrease the amount of data, which has to be provided as map updates. Therefore it focuses on the relevance of the data, which has to be transmitted to the vehicle. All the existing incremental map update approaches of Section 2 do not take the contextual relevance of the update information into account. They provide map tile updates to the vehicle whether they are relevant for its current driving task or not. The Dynamic Map Update Protocol now closes this gap by providing the functionality to take information regarding the relevance into consideration for processing map updates. Therefore, it enables significant savings in terms of transmission of data for a map update as shown in Section 4.2. The specification of relevance or irrelevance of information however is subject to individual selection criteria. The Dynamic Map Update Protocol configured for the presented evaluation results considers a map update as relevant, if it is directly effecting the current route on which the car travels. Other updates in the surrounding are only considered as optional updates with not such a high relevance for the car. This is further illustrated by the example shown in Figure 1. The Dynamic Map Update Protocol however can be configured with individual selection criteria regarding the relevance. The stated definition of relevance is just one possibility. The protocol therefore could be configured to provide relevant information for special interest groups like bus, truck (NDTV, 2017) or taxi drivers, police officers and fire fighters. Different routing functions can be defined for these interest groups, satisfying their requirements. The requesting vehicles can then tell the map server via the Dynamic Map Update Protocol which routing function is required for them.

3.1 Basic Example

The example in Figure 1 illustrates the general working principle of the Dynamic Map Update Protocol. The map database of a highly automated driving vehicle is represented by six different map tiles. The road network on which the car can base its navigation is drawn in grey colour. The car wants to reach the destination from its current start point. Therefore it exchanges information regarding its own database as explained in Section 3.3 with the map server through a

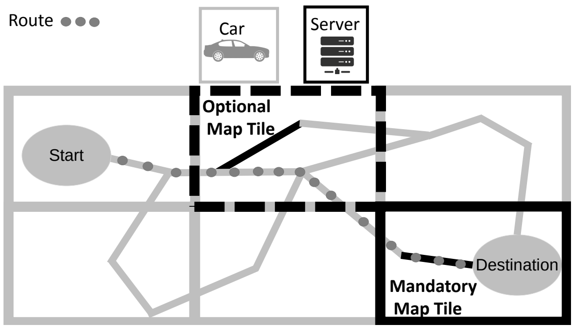

Figure 1: General working principle of the Dynamic Map Update Protocol.

wireless connection. The server itself has got a newer version of the map database with two available updates indicated by black colour. The approaches presented in the Related Work (Section 2) would provide both map tiles, which include the updates for the car. The Dynamic Map Update Protocol instead takes the relevance of the updates into account before the provision. Therefore it will only directly provide the right of the two map tiles indicating available updates as a mandatory update. This is due to the fact, that the update directly influences the optimal path on which the car travels to reach its destination. The update in the left map tile is not mandatory for the car to reach its current destination. The map tile therefore is flagged as an optional update. The car can decide itself whether it wants to receive this update or not. The update for example might be of certain interest to the car if it drives often in this area.

3.2 Protocol Fundamentals

Modern routing algorithms divide the amount of streets into different subcategories accordingly to their type. This hierarchical layering enhances the overall performance of the route calculation process as described by Min et al. (Min, 2011). For example for longer travel distances only highway routes are taken into consideration. Streets on lower layers like in the city or urban area are used when arriving at the destination or for short distance calculations. Min et al. discuss this concept only to be used for the routing algorithm, not for the map update process. However we ourselves see it as a great opportunity to provide map updates with the consideration of their actual street layer (highway, urban road, city street, ...) to further shrink down the amount of data, which has to be provided to the vehicles in the context of high definition street maps. As one of the contributions of the Dynamic Map Update Protocol we adapted and modified this general concept for the use case of map updates. The division of different

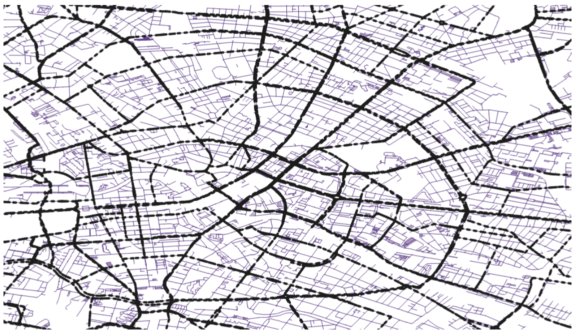

Figure 2: Division of streets into highway and city layer indicated by thick and thin lines for streets. ©OpenStreetMap contributors

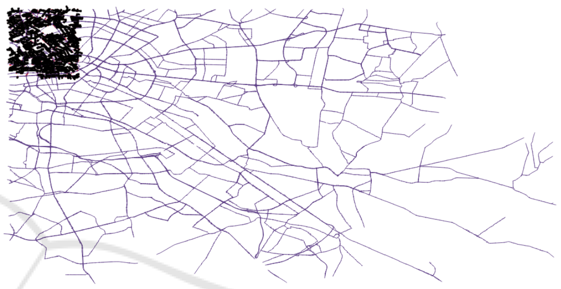

Figure 3: Size of a city street layer map tile (bold) in comparison to a highway layer map tile. ©OpenStreetMap contributors

road types is performed as a preprocessing step on the map database. All street types are clustered into two different map layers. The first layer contains all highways, the second the remaining other streets (see Figure 2). Each of the two layers is than divided into map tiles of different size (see Figure 3). This is done due to the different available speed limits. The highway layer map tiles are larger than the city street map tiles, as there the vehicles can drive a long distance in less time. Highways are therefore expected to be more important than city streets in terms of updates, since they are more frequently used by the routing algorithms. This design decision helps to reduce the amount of control traffic, which has to be conducted in the process of the protocol when requesting further map tiles. When calculating its new route the highly automated car receives only the map tile updates required for the currently used layer. While driving on a highway for example it is not interested in the city streets of a town nearby, because normally it never goes through the town on its trip. It will therefore only get updates regarding the highway, but not the city streets layer in this area.

For the identification of a specific map tile we use the indexing structure provided by the concept of Geohashes (Suwardi et al., 2015). A Geohash is a

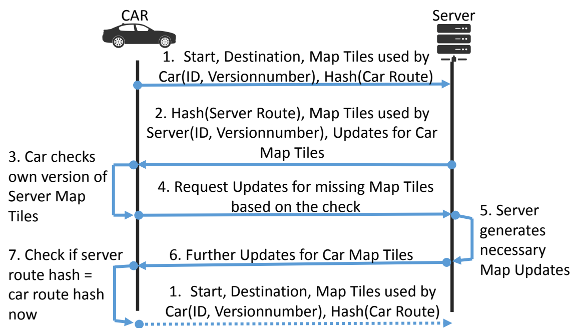
CAR
Server
1. Start, Destination, Map Tiles used by Car(ID, Versionnumber), Hash(Car Route)
2. Hash(Server Route), Map Tiles used by Server(ID, Versionnumber), Updates for Car Map Tiles
3. Car checks own version of Server Map Tiles
4. Request Updates for missing Map Tiles based on the check
5. Server generates necessary Map Updates
6. Further Updates for Car Map Tiles
7. Check if server route hash = car route hash now
1. Start, Destination, Map Tiles used by Car(ID, Versionnumber), Hash(Car Route)

Figure 4: The map exchange protocol sequence.

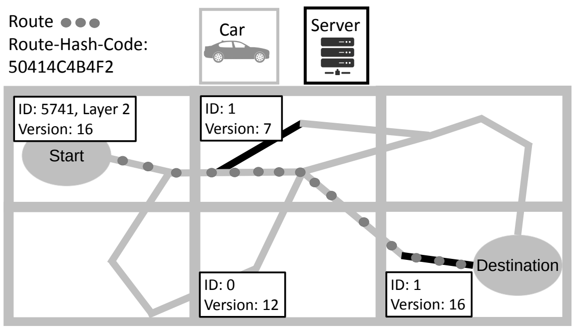

Figure 5: The sequence of used map tiles.

unique string that identifies a certain geographic area on the globus. Each of them identifies its personal rectangular bounding box. Neighbouring geographic areas just differ in the last letter of the Geohash. By adding up further letters to the end of the it the area of the bounding box is shrinked accordingly. Geohashes therefore are a well suited indexing structure for the Dynamic Map Update Protocol. They satisfy the needs of easy transitions between the different layers of the map, because upper layers cover completely the area of a discrete amount of lower layer map tiles. For the evaluation of the scenario of the city of Berlin in Section 4 we chose the Geohash string length of four letters for our highway street layer and five for the city street layer. These string lengths correlate on bounding box sizes of 39.1km x 19.5km for the highway layer and 4.89km x 4.89km for the city street layer. Thus one highway layer map tile covers the same area as 24 city street layer map tiles. The different sizes shall reduce the control flow overhead for their provisioning by taking the achievable travel distances due to speed limits into account. The general concept of different map layers can be enhanced further by adding up more fine granular layers if beneficial for future use cases of the Dynamic Map Update Protocol. Also a different more specific indexing scheme might be worth of consideration for future improvements.

3.3 Protocol in Detail

The Dynamic Map Update Protocol has to ensure that after a performed map update, the requesting vehicle will calculate the same route as the map server with its updated personal database. However it should not provide unnecessary map updates to keep the footprint of transmission costs as low as possible. The necessary steps to solve this challenging task are explained in the following. Additionally they are illustrated by the sequence diagram in Figure 4.

1.) The car requests an update for its calculated route by providing the server the following parameters. First of all the car sends the start- and destination-point based on which it calculates the route by itself. It further provides the server with the information about which tiles on the available map layers it has used for the calculation of its route. Figure 5 illustrates this in further detail. Providing this list of map tiles is done in a data efficient way. Only the map tile that contains the start point will be specified through a complete identification ID (eg. ID 5741 in the example) and layer level number (e.g. 1 for highway and 2 for city streets). The following map tiles used by the car will be described in relation to the previous map tile. Therefore only one number (ID 0-3) is required, which indicates the possible ways (up, down, left, right from the last tile) in which the vehicle can move. If the current map layer is changed it is indicated by the provisioning of an additional new layer number. Cause of drawing reasons the example assumes that the car is travelling only on one layer (2 for city streets). For each traversed map tile the car will also provide its current personal version number (e.g. version 16 of the map tile containing the start point). The exact route, which the car will use, is identified through a unique hash-code. The hash-code of the exact route is mandatory to identify the relevance of the related map tiles, which contain the route. Therefore each individual street segment, that the car has to take on its course, is identified by an own personal identifier and adds up as an additional input value to a hash function. After the completion of this process the generated hashcode (e.g. 50414C4B4F2) of the current specific route is sent to the server.

2.) Based on the start and destination point, which the server received from the car, it will calculate the shortest available route between the two points relying upon its own up to date map material. Afterwards it compares its own generated hashcode and the provided one of the car. If both do not

match, this implies that the car has got an outdated status of the map and needs to receive mandatory updates from the server as explained in the following. If both hashcodes match, it means that at least the current road to drive on is up to date in terms of the server's database.
It still might be the case that optional updates for the car are available. These are updates for the specific map tiles the car traverses, which do not belong to the actual road it drives on (see Section 3). In future situations the car could drive in these areas as well. So an update might still be beneficial for the car. Therefore the server checks the different versions of the traversed map tiles. If they also match everything is up to date. The server then just replies with a short response message, which indicates that the map material of the car does not require updates for the route.
If optional map updates are available, the server will provide the car the IDs of the map tiles, which could receive an update. Based on its driving criteria and other parameters like remaining cellular data volume, the car can decide if and when it wants to receive those updates later on.
If mandatory updates have to be provided, the server directly generates those updates and sends them back to the car.
If the server's route uses other map tiles than the car, the server provides its own hashcode and the map tile IDs and version numbers used for calculation to it . This step is necessary to ensure that the car does not find another as well outdated alternative route in its own database.

3.) till 6.) Then the car has to complete the update of its own map database for the newly provided route. Therefore it checks the map tiles indicated by the server in its own database for mandatory map updates. It requests them accordingly and gets them provided by the server.

7.) In rare occasions the provided map updates might not be enough to ensure an up to date routing capability of the car for its current route. It still might be the case that the car finds a faster, but outdated alternative route using other map tiles. The protocol has to ensure that the car uses the currently fastest route now and in future calculations. Therefore the car has to check whether its personal hash code matches with the server's code or not. If it does not, the whole procedure has to be repeated with the new conditions. This however should not happen very often because of the actual size of the map tiles. It makes shorter alternative routes most unlikely.

4 EVALUATION

To verify the capabilities of the Dynamic Map Update Protocol, we conducted our evaluation on the scenario of the German city of Berlin as explained in the following Section 4.1. The obtained results are presented than afterwards in Section 4.2.

4.1 Berlin OpenStreetMap Scenario

The currently most common map formats for high definition street maps are OpenDrive(VIRES, 2011) and the Navigation Data Standard with its Open Lane Model (NDS Navigation Data Standard e.V, 2016). To the best of our knowledge there exist no public databases for testing and evaluation of those map formats. Only small sample maps are available. They are only usable to show the capabilities of the specific map format. A map database with a long version history and a certain magnitude of applied changes however is necessary to properly test the capabilities of the Dynamic Map Update Protocol. To satisfy these requirements we decided to use the available map material of the OpenStreetMap project (Haklay and Weber, 2008). OpenStreetMap itself offers database dumps of its map material in intervals of up to one minute, one hour and one day between the newest database dump and its predecessor. The open source mapping project is community-driven and has got a strong user base of volunteers. However to take the update frequency of a high definition street map into account we selected the daily database dumps as the dataset to be tested with the Dynamic Map Update Protocol. By this decision it is ensured that the map material of OpenStreetMap has a comparable amount of changes that can be expected for a high definition street map in far less time (well below one day). As the maintenance of map material is community-driven, we selected the city of Berlin, Germany as the map material to be considered for the evaluation process. The map material of Berlin is highly detailed compared to other areas of the world, which are often mapped only very sparsely. This is due to the fact that the community of OpenStreetMap in Berlin is considerably active. This fact also ensures that the protocol is tested under realistic conditions for the scenario of highly automated driving. However it can be stated that the Dynamic Map Update Protocol can be applied to any form of navigational map, that includes a feature to weight the relevance of its content. The actual amount of data, which can be saved in the process of map updates, will scale accordingly to its update frequency. As stated in Section 3.2 the map is divided up into different layers regarding the type of streets. All the streets considered as highways by OpenStreetMap definition are grouped into one layer. All the other streets form their own layer.

To ensure the significance of our test results we conducted the evaluation as explained in the following with a test data set of 30 consecutive days of OpenStreetMap map dumps (from 1st August till 31st August 2016).

With this data set we tested the protocol on differences of 1 day and of 15 days between two map databases to show the effects of map updates. The setup with 1 day differences resembles the daily driving use case of a vehicle, the 15 day differences resembles a situation, where a highly automated vehicle is not used for a longer period of time. This could be for example the case, when his owner has been on vacation before.

4.2 Results

To verify the hypothesis that the Dynamic Map Update Protocol has a significant impact on the reduction of transmitted map updates, the following two tests described in Section 4.3 and 4.4 have been executed. For each test we compared the performance between an existing simple incremental map update approach (see Section 2) and the Dynamic Map Update Protocol. The amount of processed map tiles (processing load for the map server) and of changed map objects (the actual data transmitted to the vehicle) have been selected as quality metrics (see Table 1). The simple approach provides the vehicles with all available map tile updates for its currently calculated route as proposed by Bastiaensen et al. (Bastiaensen and others, 2003). The Dynamic Map Update Protocol is configured to provide only updates for map tiles along the route, which have been identified as relevant, as defined in Section 3. To make sure that the expected benefits are only gained through the Dynamic Map Update Protocol, both approaches have been applied on the same preprocessed map database (see Section 3.2). For each test a specific set of random trips through the city of Berlin has been generated. The average trip driving distance of a European is between 10 to 30 km, as stated by (Pasaoglu et al., 2012). We included this fact in the evaluation process and set up 60% of all the trips to be in this range. Half of the remaining trips were configured to be either above or below this value range. For the process of routing we used the Dijkstra algorithm included in the Each route consists out of a unique sequence of nodes and connecting way IDs as specified by the OpenStreetMap database, which we imported into pgRouting. An exemplary excerpt of the achieved test results

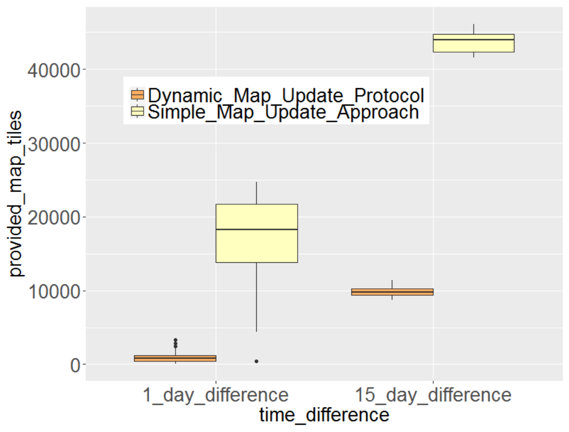

Figure 6: Number of Map Tiles which have to be processed for transmission for 10,000 independent trip requests.

is presented in Table 1.

4.3 Savings on the Server Side

The first test was conducted to gain an insight into the amount of data that could be saved from being processed and provided by the server side (e.g. tile updates in Table 1). Therefore 10,000 individual trips were processed by the two algorithms (see Figure 6). For the whole set of trips the simple map update approach had to process an average of 17,272 map tiles for a one day difference between the databases. For the same set the Dynamic Map Update Protocol only had to provide an average of 928 map tiles. This means that the server had to process only 5% of the overall update data when using the Dynamic Map Update Protocol. As we expected, for the databases with 15 days of difference this ratio degraded to 22%. This is due to the fact that the overall amount of updates to be provided for the older map database increased. The Dynamic Map Update Protocol had now an average of 9,945 map tiles to handle for transmission compared to 43,732 for the simple map update approach. The amount of savings is still huge.

4.4 Savings for a Highly Automated Driving Car

In contrast to the map server the actual amount of map data, which has to be transmitted to a requesting vehicle, is its most important quality criteria. The requesting vehicle has to pay the cellular network provider directly. The on board processing costs of the data are negligible in comparison. To consider this situation we executed a second test. It was conducted to evaluate the amount of actual map objects, which had to be transmitted to the vehicles (e.g. objects with

Table 1: Exemplary excerpt of the achieved test results.

Execution Run	Old Map Date	New Map Date	Dynamic Map Update Protocol Tile Updates	Simple Map Update Approach Tile Updates	Tile Savings in %	Dynamic Map Update Protocol Objects with Changes	Simple Map Update Approach Objects with Changes	Object Savings in %
1	1-8-2016	2-8-16	1,128	7,413	84.78%	30,427	85,734	64.51%
2	2-8-2016	3-8-16	169	3,252	94.80%	597	9,567	93.76%
...	...	...	...	...	...	...	...	...

changes in Table 1). The amount of map tiles, used as evaluation metric in the first test, is not a direct indicator therefore. It is expected that different trips will often use the same connecting streets (e.g. main roads or highways), which results in a decreasing amount of map objects to be delivered to a car over time. To resemble this behaviour we created a scenario of a car requesting 100 consecutive trips. In contrast to the server scenario the provided map updates were stored as already available in the databases of the vehicle for the remaining amount of requests. After each of 10 requests we took a snapshot of the additionally provided map objects. The obtained results are presented by Figure 7 for one day and by Figure 8 for 15 days of difference in time. Both figures show that the Dynamic Map Update Protocol in average always stays below the amount of data, which is provided by the simple map update approach. The Dynamic Map Update Protocol only had to provide an average of 43.1 map objects for the full sequence of 100 trips compared to an average of 70.6 for the simple map update approach for one day time difference. For the time difference of 15 days we achieved similar results (an average amount of 775.4 map objects for the Dynamic Map Update Protocol and 962.5 for the simple map update approach). To ensure the statistical significance of the achieved means we further conducted a paired t-test on the test results. The obtained p-values for the different amount of requests are presented in Table 2 and indicate clearly that we can reject the null hypotheses H_0: "The two approaches provide the same overhead in terms of update data."

$$H_0 : \mu_{dynamicMap} = \mu_{simple} \tag{1}$$

Thus we accept the alternative hypothesis H_1: "The Dynamic Map Update Protocol transmits less data than the simple map update approach."

$$H_1 : \mu_{dynamicMap} < \mu_{simple} \tag{2}$$

Where μ is the mean of the number of updated map objects.
To conduct 100 trips in a temporal sequence would only leave around 5 minutes for each of the trips to be finished in average over an 8 hours working day. It is highly expected with reference to (Pasaoglu et al., 2012) that nearly all drivers won't come close to this huge amount of requests. Therefore the behaviour of the Dynamic Map Protocol is beneficial too. It can be seen in both figures that the simple map update approach provides most of the overall map updates very early (e.g. after 20 requests for 1 day of difference). The Dynamic Map Update Protocol however shows a more steady increase in providing the necessary map updates. Therefore it saves even more data when not requesting a huge amount of daily navigation requests.
To ensure that the achieved results were not influenced by the chosen set of 100 trips we conducted another test in which 1,000 independent cars performed 10 consecutive trips. These results are presented in Figure 9 and show that the Dynamic Map Update Protocol outperforms the simple map update approach also under those test conditions.

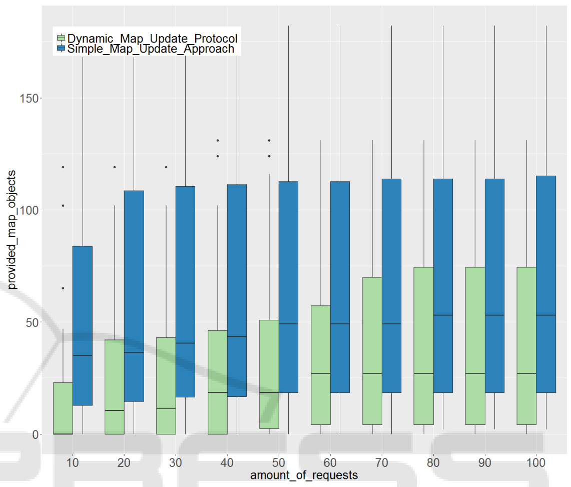

Figure 7: Map objects provided after 10 to 100 consecutive requests. Databases with 1 day difference.

Table 2: Obtained p-values of the paired t-test for the results of the second test.

number of requests	10	20	30	40	50	60	70	80	90	100
1 day time difference	1.90e-06	1.46e-06	1.06e-06	2.69e-06	9.75e-07	5.40e-07	1.40e-06	1.42e-06	1.36e-06	1.40e-06
15 days time difference	8.60e-09	3.96e-09	1.36e-09	7.14e-11	1.39e-12	1.69e-13	3.43e-11	2.12e-10	4.14e-11	8.03e-11

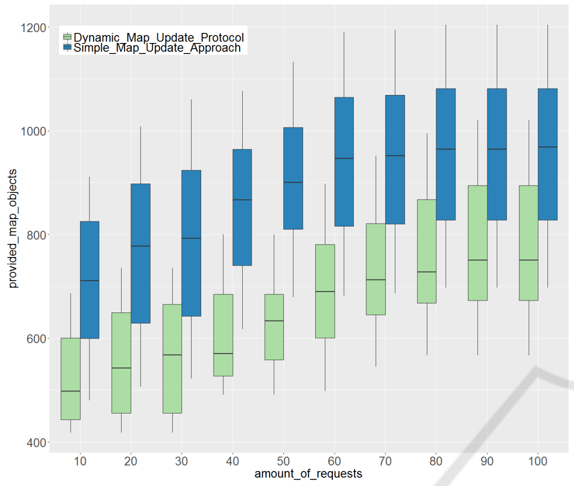

Figure 8: Map objects provided after 10 to 100 consecutive requests. Databases with 15 day difference.

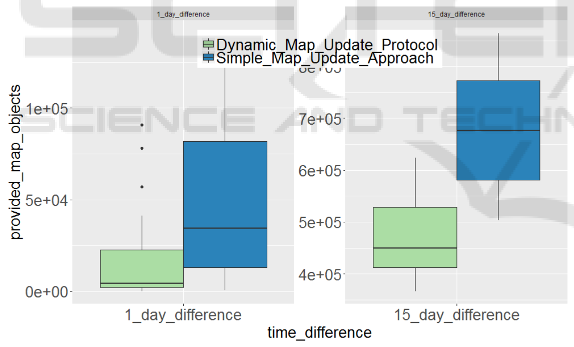

Figure 9: Map objects provided to 1,000 independent vehicles performing 10 consecutive trips.

5 CONCLUSIONS AND FUTURE WORK

This paper presents the Dynamic Map Update Protocol. The protocol has been developed to provide a new data efficient strategy to update high definition street maps, which are required for future highly automated vehicles to be able to drive. In contrast to existing map updating concepts our protocol takes the relevance of the map data into account for the process of update generation. To verify the capabilities of the newly developed protocol extensive evaluation tests with OpenStreetMap databases of the German city of Berlin have been conducted. The gained results show that the Dynamic Map Update Protocol achieves a significant decrease in transmission data and processing time to enable the same driving tasks compared to existing solutions. The current configuration of the protocol still leaves further optimisation potential for future work. Dynamic indexing structures that adapt themselves to the requirements of the highly automated driving vehicle and even more efficient data transmission concepts (resolving the general concept of map tiles) are two research directions, which we will investigate further.

REFERENCES

Aeberhard, M., Rauch, S., Bahram, M., Tanzmeister, G., Thomas, J., Pilat, Y., Homm, F., Huber, W., and Kaempchen, N. (2015). Experience, results and lessons learned from automated driving on germany's highways. In *IEEE Intelligent Transportation Systems Magazine*, volume 7, pages 42–57.

Asahara, A., Tanizaki, M., Morioka, M., and Shimada, S. (2008). Locally differential map update method with maintained road connections for telematics services. In *Mobile Data Management Workshops, 2008. MDMW 2008. Ninth International Conference on*, pages 11–18. IEEE.

Barker, P. (2015). *Industry expert explains why autonomous cars need a map - HERE 360.* http://360.here.com/2015/05/07/brad-templeton-autonomous-cars-maps/, [online accessed 2016-04-08].

Bastiaensen, E. and others (2003). ActMAP: real-time map updates for advanced in-vehicle applications. In *Proceedings, 10th World Congress on ITS, Madrid.*

Bender, P., Ziegler, J., and Stiller, C. (2014). Lanelets: Efficient map representation for autonomous driving. In *Intelligent Vehicles Symposium Proceedings, 2014 IEEE*, pages 420–425. IEEE.

Boensch, R. (2016). *With high resolution into the autonomous world.* VDI Nachrichten Ausgabe 03, http://www.vdi-nachrichten.com/Technik-Wirtschaft/Hochaufloesend-in-autonome-Welt, [online accessed 2016-12-05].

Cooper, A. and Peled, A. (2001). Incremental updating and versioning. In *Proceedings of 20 th International Cartographic Conference*, pages 2804–2809.

ETSI (2011). *ETSI TR 102 863, Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Local Dynamic Map (LDM); Rationale for and guidance on standardization*.

Haklay, M. and Weber, P. (2008). Openstreetmap: User-generated street maps. In *IEEE Pervasive Computing*, volume 7, pages 12–18.

Hammerschmidt, C. (2016). *With data from the cloud to the living map*. VDI Nachrichten Ausgabe 03, http://www.vdi-nachrichten.com/Technik-Wirtschaft/Mit-Daten-Cloud-lebenden-Karte, [online accessed 2016-05-12].

Hitachi, A. L. (2016). *Map update service/solution*. http://www.hitachi-automotive.co.jp/en/products/cis/02.html, [online accessed 2016-04-26].

Ishikawa, K., Ogawa, M., Azuma, S., and Ito, T. (1991). Map navigation software of the electro-multivision of the'91 toyoto soarer. In *Vehicle Navigation and Information Systems Conference, 1991*, volume 2, pages 463–473. IEEE.

Jo, K. and Sunwoo, M. (2014). Generation of a precise roadway map for autonomous cars. In *IEEE Transactions on Intelligent Transportation Systems*, volume 15, pages 925–937.

Lawton, C. (2015). *Why an HD map is an essential ingredient for self-driving cars - HERE 360*. http://360.here.com/2015/05/15/hd-map-will-essential-ingredient-self-driving-cars/, [online accessed 2016-04-08].

Lee, S. and Lee, S. (2013). Map generation and updating technologies based on network and cloud computing: A survey. In *International Journal of Multimedia and Ubiquitous Engineering*, volume 8, pages 107–114.

Ling, H. (2013). *NavInfo: Incremental Update is the Revolution for Map Production System*. http://www.navinfo.com/en/news/detail.aspx?id=944 &sort=3, [online accessed 2016-04-05].

Liu, Y., Yang, Z., and Han, X. (2010). Research for incremental update data model applied in navigation electronic map. In *Advanced Computer Control (ICACC), 2010 2nd International Conference on*, volume 1, pages 261–265. IEEE.

Madrigal, A. C. (2014). *The Trick That Makes Google's Self-Driving Cars Work - The Atlantic - GOOGLE*. http://www.theatlantic.com/technology/archive/2014/05/all-the-world-a-track-the-trick-that-makes-googles-self-driving-cars-work/370871/, [online accessed 2016-05-01].

Min, K. (2011). A system framework for map air update navigation service. In *ETRI Journal*, volume 33, pages 476–486.

Min, K.-W., An, K.-H., Kim, J.-W., and Jin, S.-I. (2008). The mobile spatial DBMS for the partial map air update in the navigation. In *Intelligent Transportation Systems, 2008. ITSC 2008. 11th International IEEE Conference on*, pages 476–481. IEEE.

NDS Navigation Data Standard e.V (2016). *NDS Open Lane Model Press Release*. http://www.nds-association.org/wp-content/uploads/20160914-PR-E.pdf, [online accessed 2016-11-14].

NDTV (2017). *How UPS Trucks Saved Million of Dollars by Eliminating Left Turns*. http://gadgets.ndtv.com/transportation/features/how-ups-trucks-saved-million-of-dollars-by-eliminating-left-turns-1657808.

Pasaoglu, G., Fiorello, D., Martino, A., Scarcella, G., Alemanno, A., Zubaryeva, C., Thiel, C., European Commission, Joint Research Centre, and Institute for Energy and Transport (2012). *Driving and parking patterns of European car drivers: a mobility survey.* Publications Office. OCLC: 847460656.

Perkins, C. (2015). *Tesla is mapping out every lane on Earth to guide self-driving cars*. http://mashable.com/2015/10/14/tesla-high-precision-digital-maps/#v3xHW6i4QSqU, [online accessed 2016-04-06].

Plack, J. (2013). *The unmatched quality of HERE Maps content - HERE 360*. http://360.here.com/2013/03/27/the-unmatched-quality-of-here-maps-content/, [online accessed 2016-05-01].

Schumann, S. (2014). *Why were mapping down to 20cm accuracy on roads - HERE 360*. http://360.here.com/2014/02/12/why-were-mapping-down-to-20cm-accuracy-on-roads/, [online accessed 2016-05-01].

Stevenson, J. (2016). *Making the invisible visible with the HD Live Map*. http://360.here.com/2016/09/23/making-the-invisible-visible-with-the-hd-live-map-video-demo/, [online accessed 2016-11-14].

Suwardi, I. S., Dharma, D., Satya, D. P., and Lestari, D. P. (2015). Geohash index based spatial data model for corporate. In *Electrical Engineering and Informatics (ICEEI), 2015 International Conference on*, pages 478–483. IEEE.

TomTom (2016). *TomTom Enables Autonomous Driving*. http://automotive.tomtom.com/uploads/assets/993/1475151547-autonomous-driving-product-info-sheet.pdf, [online accessesd 2016-05-15].

VIRES (2011). *OpenDRIVE - managing the road ahead*. http://www.opendrive.org/docs/VIRES_ODR_-OCRG.pdf, [online accessed 2016-11-14].

Ziegler, J., Bender, P., Schreiber, M., Lategahn, H., Strauss, T., and et. al. (2014). Making bertha drive - an autonomous journey on a historic route. In *IEEE Intelligent Transportation Systems Magazine*, volume 6, pages 8–20.