https://doi.org/10.26552/com.C.2013.2.6-12

Jan Janech – Tomas Baca – Anton Lieskovsky – Emil Krsak – Karol Matiasko *

DISTRIBUTED DATABASE SYSTEMS AND DATA REPLICATION ALGORITHMS FOR INTELLIGENT TRANSPORT SYSTEMS

One can look at the vehicular ad hoc network (VANET) as at the source and storage of information useful for the safety, fluency and comfort of road traffic, which makes it a valuable tool for ITS. Our research is focused on data dissemination in VANET. We perceive the nodes of VANET as the nodes of a distributed database system (DDBS). As such they do not only store data but they also create new data. Normally, in the course of time the data lose their usefulness. The system we designed was tested by means of a computer simulation carried out by the simulation tool AdHocSim.FRI.

Keywords: *VANET, ITS, DDBS, simulation, data dissemination, AdHocSim.FRI, AD-DB.FRI.*

1. Introduction

The vehicular ad-hoc networks (VANETs) are the subset of mobile ad-hoc networks (MANET) with the differentiating quality based on vehicles being the network nodes. This means that how the node moves is affected by the traffic regulations. In a network like this each node plays the role of an end-system, as well as it behaves like a node that is capable of sending data into the environment, or communicating with the other nodes. The mobility of VANET networks and partial absence of a fixed infrastructure are what makes VANET attractive for time-critical applications. As the movement of on-board-unit nodes (*OBU*) is limited by traffic rules, *VANET* utilizes certain nodes called road-side-units (*RSU*) that are fixed to road-traffic infrastructure.

In our article we look at *VANET* as it is a distributed database system. In a distributed database system, data are often replicated. The aim of the replications is to improve the reliability and availability of the data throughout the whole network. In fixed networks, the replicas are often stored on the nodes that need them the most, in order to reduce the cost of the remote data access. However, in *VANET*, due to the excessive mobility of *OBU* nodes, this kind of solution is inadequate. The availability and reliability (consistency) of the data become an important problem. The fragmentation of the network and slow responses from some of the nodes in particular has an influence over the availability and reliability (consistency) of the data in VANET.

2. State of the Art

Besides our own *AD-DB .FRI* protocol described in Section 3, there is no other solution that would focus specifically on the *VANET* environment. On the other hand, there are some more general solutions for the *MANET*.

The *TriM* protocol [1][2] is the one of the first ones which attempted to solve the data distribution in *MANET* in a general way. It was developed as a part of a dissertation thesis at the University of Oklahoma [1]. It was designed mainly with the regard to energy consumption and the possibility of using three modes of communication [2]:

- Push mode – it represents sending the data via broadcast messages.
- Pull mode – it represents sending the data on the basis of a request.
- P2P communication – data request.

The protocol takes into account the distinction between two node types: *SMH* and *LMH*. A client or a small mobile host (*SMH*) is a node with limited computing resources, limited storage capacity, limited possibilities of communication and limited energy source. A server or a large mobile host (*LMH*), in contrast to *SMH*, has less limited resources. In the protocol, LMH is used as a database server.

The main disadvantage of the protocol is that it requires each server to have the same data [3].

The *HDD3M* protocol tries to fix the problems of the *TriM* protocol. Just like the *TriM*, it uses all of the three modes of communication. It tries to optimize their usage for the energy consumption. But, unlike the *TriM*, it provides the possibility of data fragmentation in the mobile *DDBS*, and it also deals with the data changes in the transactions [3].

* **Jan Janech[1], Tomas Baca[1], Anton Lieskovsky[2], Emil Krsak[1], Karol Matiasko[2]**
[1] Department of Software Technologies, Faculty of Management Science and Informatics, University of Zilina, Slovakia, E-mail: jan.janech@fri.uniza.sk
[2] Department of Informatics, Faculty of Management Science and Informatics, University of Zilina, Slovakia

The *HDD3M* distinguish three types of nodes: a request node which sends queries to *DDBS*, a database node containing its own mobile database, and a database directory which contains the information about the fragment location in the distributed database. The database directory is also responsible for receiving and processing the queries from the individual nodes.

However, the problem of these protocols is that they focus primarily on the minimum consumption. This is an important feature to have for the general *MANET*, but at the same time it constitutes an obstacle in the deployment in the *VANET* environment. There are many restrictions for the systems operating in *VANET* but the limitation of the energy source or computing power is not one of them.

3. The concept of the AD-DB .FRI database system

With the classical distributed databases, it is necessary for each node to know about all of the other nodes, as well as about the distribution of the data on them. This information makes it possible to execute distributed queries. The query node needs to know whom to send which parts of the query. The information about the data distribution in the system is stored in so called Global Directory/Dictionary (*GD/D*) [4].

If we try to transfer this principle into the *VANET* environment, we will find out that it is impossible [1][5][6]. The individual nodes of the network are not aware of each other, and it is even technically impossible to guarantee constant possibility of their mutual communication.

The only way how to use the distributed database systems in the *VANET* environment is to replace the *GD/D* by a different principle. In *VANET*, every node is familiar with only its immediate surroundings. Therefore, making any queries in such environment is remarkably limited. The only nodes we can require data from are those within the communication range of the query node. That is why the virtual node-clusters are naturally created in the system. They can communicate with each other and pass on the data between each other.

Therefore, our solution is to restrict the possibilities of making queries only to the nodes accessible within a cluster, and to introduce a new notion – Cluster Directory/Dictionary (*CD/D*). The *CD/D* contains information about the distribution of just the data accessible in the current cluster. But, to simplify the system, this catalogue never exists as one whole. Each of the nodes remembers only a part of it, concerning the fragment of the distributed database handled by the node in question. Also Ozsu and Valduriez [4] describe this method as one of the ways of storing *GD/D*.

Our *AD-DB .FRI* system develops this idea even further. The *CD/D* does not get folded even when a query is being carried out. Instead, the query is sent to all nodes neighboring to the query node. The data nodes which receive the query decide on their own, on the grounds of their part of the *CD/D*, whether to send any data to the query node. This way, most problems caused by the introduction of the distributed database system in the *VANET* environment are solved. The query execution process does not depend on whether the query node can communicate and who with. The only thing that changes is the set of data which the node receives as a reply. Therefore, the *AD-DB .FRI* system is possible to use only in case it does not matter that completed data are not the reply. Thanks to using the specialized *OSACP* protocol (see Section 4), the system even enables processing incomplete data in case the connection between the data node and query node is interrupted during sending the reply.

By means of the principle described above, the *AD-DB .FRI* enables to use two querying methods, according to the communication mode being used: the pull method and the push method.

3.1 Pull method

A query node sends a query packed as a broadcast message. Each data node which receives the query checks whether it has the requested data or their part. If it does, it sends the answer to the query node in the form of the unicast message. In case it could respond to the query only partly, it informs so in its reply, and says specifically which part of the query it was able to process. The query node waits for the response for a specified period of time, and then it starts processing the replies. If it receives partial responses, it tries to interconnect them by the operations which remained unexecuted in the query.

The principle is illustrated in Fig. 1.

3.2 Push method

A data node in regular intervals sends responses to a pre-programmed query, bundled with the query itself, in the form of a broadcast message. If any of the nodes receives this response, it checks if it needs the given data, and if it does, it processes them.

The principle is illustrated in Fig. 2.

4. Object Structure Aware Communication Protocol (OSACP)

The *OSACP* is an application level communication protocol. It is designed to provide the communication layer for the *AD-DB .FRI*. It is supposed to transfer large messages with a well-defined internal object structure that is known equally to both, the sender and the receiver of the message. The protocol follows the philosophy of *AD-DB .FRI* that the query node does not expect and does not require to receive complete responses from all data nodes in *VANET*. Instead, it tries to put together as much valuable information as possible from the data that it has managed to receive at a given moment.

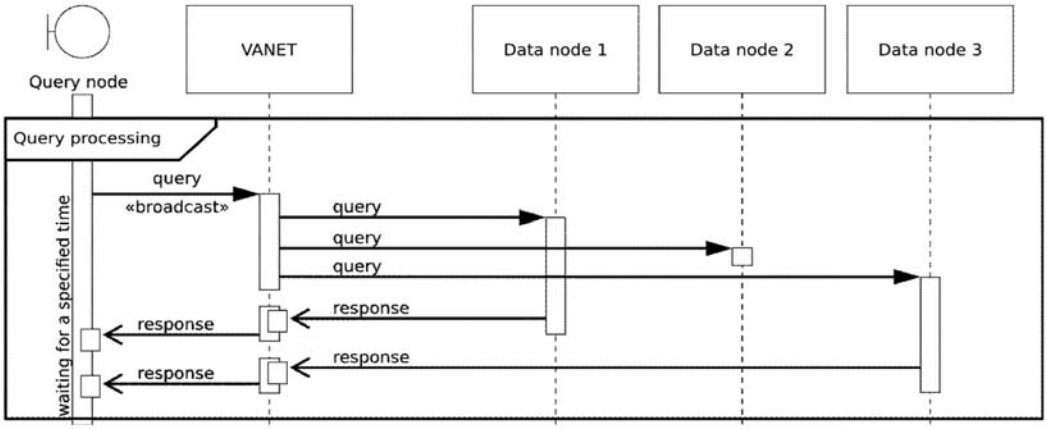

Fig. 1 PULL method

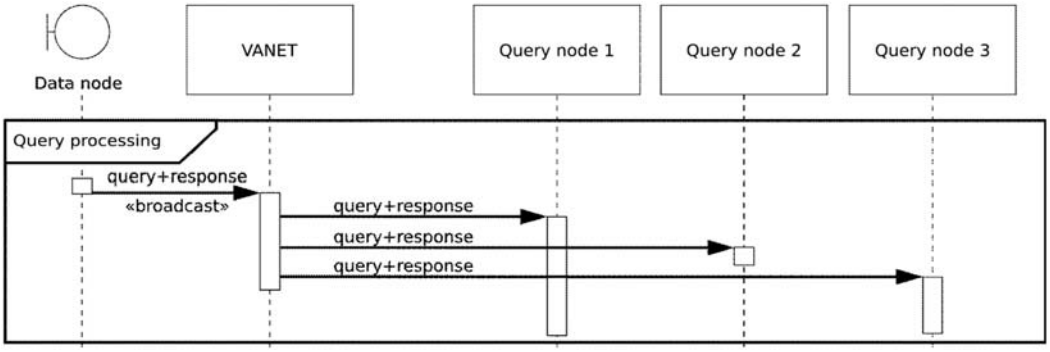

Fig. 2 PUSH method

The endpoint-to-endpoint communication service is provided by the User Datagram Protocol (*UDP*) protocol, which runs atop Internet Protocol version 6 (*IPv6*). We have experimentally compared it with the Transmission Control Protocol (*TCP*), the only other available alternative, in the simulation of *AD-DB .FRI* in *VANET* and we discovered that the UDP outperforms the *TCP* in efficiency and the total amount of useful data delivered [7][8][9]. Since neither the sender nor the receiver is informed by the *UDP* on the successfulness of a datagram delivery, it is up to the *OSACP* to deal with losses, duplications and changed order in the delivery of datagrams by itself. Moreover, the network communication between two arbitrary endpoints in *VANET* is frequently short-lived, and its interruptions are often irreparable. Therefore, the *OSACP* handles any interruption as a standard situation, and not an error.

Any message transmitted by the *OSACP* is expected to have an object structure with one major object that directly or indirectly refers to all of the other objects of the structure. If any object in the original structure is referenced to more than once, then, before the message is sent, all but one reference are replaced by the virtual

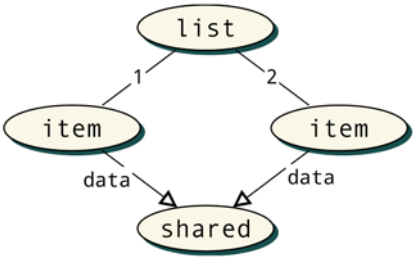

Fig. 3 Object structure with references forming a cycle

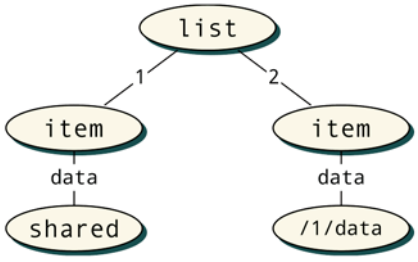

Fig. 4 Object structure with replaced reference by new Address object

Address objects. Afterwards the message takes up a tree structure, in which the main object is in the root of the tree. Then we assume that every node in the tree carries enough data to make its contents at least partly meaningful even without the data stored in its child nodes. On the other hand, its data are not expected to make any sense without the context of the data stored in its parent nodes. Practically, the elementary values (numbers, texts, etc.) are stored in the leaf nodes. The non-leaf nodes represent complex objects composed of the references to other objects, and thus they create a multi-level tree structure.

```
ROOT ::= LIST[65535] of Person;

Person ::= SEQUENCE {
   id       INTEGER {1 .. 1000000},
   name     STRING[20],
   surname  STRING[20],\
   shared   POINTER to STRING[65535]
}
```

Fig. 5 Sample declaration of an object structure

Both, the sender and the receiver of the message are expected to share the same declaration of the transported object structure. The declaration specifies the class of the main object, the list of its attributes, their classes and so on. The declarations are either hard-coded or derived from their previous communication, e.g. it is possible to derive the declaration of database response from a database query. Using the declaration enables the communication protocol to recognize the object structure.

Messages are expected to be relatively large, hence, inevitably fragmented, so that every fragment fits into one datagram. Depending on the momentary network state, any fragment may get lost during the transmission, therefore every fragment has to be processable independently on the other fragments. That is why the *OSACP* inserts an identifier of the message as a whole into every fragment. The identifier, among other things, matches the transmitted object structure within the message to its declaration. Every fragment also contains its identifier within the message, composed of the relative address of first leaf node encoded in the given fragment. Furthermore, every fragment contains the data from the tree-root node, as well as the data from the parent nodes for every node which is transferred in the given fragment. Although it may seem that this would cause too much redundancy, we believe that most of the non-leaf node data are possible to derive from the declaration, and that is why it is not necessary to transfer them within the fragment.

Our data-encoding was inspired by the *ASN.1 DER*, which uses the *type-length-value* triplet for every object of the object structure. The *OSACP* uses the declaration of the object structure which already contains the information about the type and length of most of the objects. The length is not clear only in cases of collections and texts, and the type is ambiguous only rarely.

Thanks to the identifier of the message and the identifier of the fragment, the declaration of the object structure can be assigned to the received message. It is possible to create the skeleton of the message from the declaration. Consequently, the skeleton of the message gets filled in by the receiver with the data received in the individual fragments. Should any datagrams not be received properly, then the skeleton of the message is not filled in completely, yet, because its missing parts are explicitly denoted, the message-receiving process is able to recognize them.

5. Replication algorithms

The application usage of the technologies mentioned so far (Pull, Push, *AD-DB .FRI, OSACP*) is applied namely with distributing the data in the Ad-Hoc network. Since our work deals with the data distribution in the Ad-Hoc network of *VANET* type, we can also speak of the data replication in *VANET*. Therefore, replication algorithms for *VANET* which would use the mentioned technologies are required.

Due to the specific attributes of the *VANET* (frequent changes of the network topology, high node mobility etc.), we designed a series of our own algorithms, to add up to the existing ones.

5.1 Skip Copy algorithm

It is a replication algorithm designed for the *MANET* networks but, of all generally known algorithms, it meets the *VANET* network needs the best. Skip Copy algorithm [10] functions as follows: it provides sending data only up to a certain distance from the source node (Position parameter and Hop parameter). The data replicas are not created on all of the neighboring nodes but only on each nth node determined by Skip parameter.

The main advantage of this algorithm is that it is not necessary to keep the chart of the information on the neighboring nodes.

5.2 AORPlD algorithm

The Active *OBU* Replica Pull Dissemination algorithm (*AORPlD*) is based on Pull method mentioned in Section 3.1. It functions as follows: after a query is sent out, the node waits for an established period of time (timeout) for the reply. When the reply comes, the query node matches it to the sent query. After the given time limit expires, the node ceases to expect any replies, and considers the query processed.

5.3 AORPsD algorithm

The Active *OBU* Replica Push Dissemination algorithm (*AORPsD*) as the method of the data replication in the *VANET* network, using *AD-DB .FRI*, also enables mobile nodes to function as replication nodes. *AORPsD* runs in two threads. The first one

runs in *RSU* and *OBU*, and it is in charge of distributing data into the network using Push method, according to Section 3.2. The second one runs only in the *OBU* nodes, and takes care of processing the replicas [11][12].

5.4 SPA algorithm

The Simple Pulling Algorithm (*SPA*) principle of the replication algorithm is based on the fact that the *OBU* nodes in *VANET* do not try to behave like replication nodes. Their aim is to satisfy only their own queries. All they try to do is to update the replicas they are interested in in the given time. Doing so, they do not try to extend their local DB by the data (replicas) which could have some meaning to them in the future. Each node in the network owns its own local DB over which it implements local queries.

5.5 IRA algorithm

The Independent Replication Algorithm (*IRA*), unlike *SPA*, also thinks of creating the replicas of the data *OBU* currently does not need. In established intervals, each *OBU* in *VANET* sends the information about the content of its local *DB*. On the basis of the information the other *OBU*s in the network do the update of their databases. The remarkable difference between the *IRA* and *SPA* algorithms is that the node, while using *IRA*, always does a query on its local *DB*, and gets the result immediately.

5.6 DRA algorithm

This algorithm combines the good properties of both SPA and the IRA algorithms. The first part of this algorithm is based on SPA algorithm. Second part is realized by IRA algoritm.

It means, that decision to realised IRA algorithm is conditioned by request creation on OBU side.

5.7 PDDA algorithm

The Push Different Data Algorithm (*PDDA*) distributes data by means of Push method only. At first, the *RSU* nodes distribute data over the environment. Consequently, the *OBU* nodes distribute the data that they own but they did not acquire them from the environment. Therefore, the distributed data are different from the ones the current *RSU* distributes. The aim of this algorithm is to achieve a balanced information status in the *VANET*.

6. Simulation experiment

The goal of the simulation experiment was to implement a data replication in *VANET* using the described technologies and designed algorithms. The simulation experiments were implemented by means of the *AdHocSim.FRI* tool.

When assessing the simulation experiments, we followed two values:

- Satisfaction with the data a node needed in a given moment. We modeled the satisfaction with the data by means of so called L-function [13]. (later data meant higher satisfaction)
- The volume of the transmitted data, the load on the transmission channel of the *VANET* respectively.

The simulation scenario for all algorithms was identical. The *OBU* units that receive information from other *OBU* and *RSU* units, come into the road network. The information is received on a request from a particular *OBU*. The behavior of the *OBU* and *RSU* units is specifically determined by the replication algorithm.

6.1 The simulation tool AdHocSim.FRI

AdHocSim.FRI [7][14][15] is a discrete event simulation tool specialized on the simulation of deployed applications within the *VANET* environment. It contains the model of road traffic, models of *WAVE DSRC, IPv6, TCP, UDP* protocols and the model of an on-board computer with an operating system embedded in cars. It is implemented in Python, and the simulations are defined in Python as well [7][14][15].

6.2 Results

Implementing the simulation experiments brought the following results: from the viewpoint of the satisfaction with the data, the algorithms *DRA* and *PDDA* proved themselves to be the most successful (see Fig. 6–7).

It is important to note that the related algorithm (IRA and DRA) have quite different results. Satisfaction of DRA algorithm is higher than satisfaction of algorithm IRA. It means, that higher satisfaction is not achieved by more frequent iterations of IRA algorithm. Higher satisfaction is achieved by using the algorithm in "the right time", that is, when the request on the OBU is created.

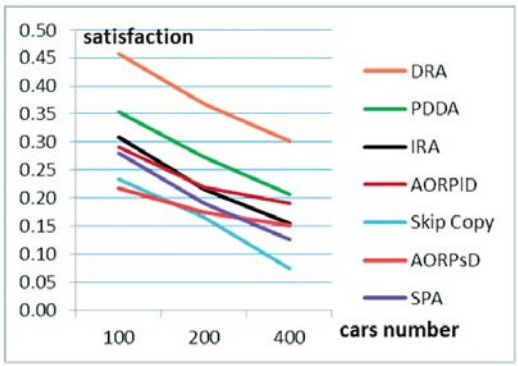

Fig. 6 The relation between the satisfaction and the number of cars in the simulation for the individual replication algorithms

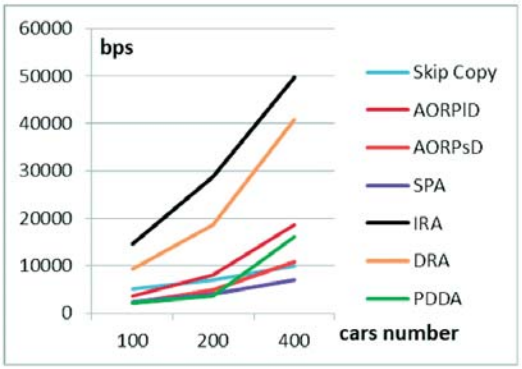

Fig. 7 The relation between the volume of the transmitted data and the number of cars in the simulation for the individual replication algorithms

As Fig. 7 shows, from the viewpoint of the network load, the algorithms *SPA* and Skip Copy achieved the best results.

From the viewpoint of the network is also important, that related algorithms (IRA and DRA) have also different results. We can see, that not only satisfaction but also number of transmitted data is better during using DRA algorithm than using IRA algorithm. These results means, that DRA algorithm is much better for data replication than IRA algoritmh.

But, it is clear from the results that it is impossible to say unequivocally which of the algorithms is better. Therefore, to get an over-all evaluation, it is necessary to define the way of using the data and the requirements on the replication algorithm. As far as volume-demanding replications are concerned, and the critical point is minimizing the transmitted data, it is possible to use the *SPA* algorithm or the Skip Copy algorithm. But if the goal is to maximize the satisfaction, the *DRA* and *PDDA* algorithms achieve better results.

7. Conclusion

Our article presents the on-going research focusing on improving the quality of *ITS* by means of the communication of close participants of road traffic through *DDBS*. The system we suggest to use is based on the *AD-DB .FRI* database system which communicates with the *OSACP* protocol. Over these, we designed several algorithms ensuring the data replication between the individual nodes. To compare those, as well as for the entire research, we used the computer simulation carried out by the *AdHocSim.FRI* tool.

Within the on-going research, we would like to concentrate on several areas. At the network level, it is the experimental verification, if the *OSACP* protocol will achieve higher efficiency, if it uses the Stream Control Transmission Protocol (*SCTP*) instead of the *UDP* for the communication between the end-nodes. The *AD-DB.FRI* system offers some space in the area of the optimization of the database query processing, and also in securing the communication channel. We will also continue to develop new algorithms for data distribution. One of our visions, conditioned, however, by the availability of hardware, is also implementing the described experiments in the real *VANET* environment, and not just by means of the computer simulation.

Acknowledgment

This contribution/publication is the result of the project implementation: Centre of excellence for systems and services of intelligent transport II., ITMS 26220120050 supported by the Research & Development Operational Programme funded by the ERDF.

Európska únia
Európsky fond regionálneho rozvoja

Operačný program VÝSKUM a VÝVOJ

Agentúra Ministerstva školstva, vedy, výskumu a športu SR pre štrukturálne fondy EÚ

"Podporujeme vyskumne aktivity na Slovensku/Projekt je spolufinancovany zo zdrojov EU."

References

[1] FIFE, L. D: TriM: *Tri-Modal Data Communication in Mobile ad-hoc network Database Systems.* Dissertation thesis. University of Oklahoma, 2005.

[2] FIFE, L. D., GRUENWALD, L.: *Mobile ad-hoc Network Data Communication in Large Geographic Areas. Wireless Communications and Networking Conference,* 2006.

[3] RAHBAR, A., MOHSENZADEH, M., RAHMANI, A. M.: HDD3M: *A New Data Communication Protocol for Heterogeneous Distributed Mobile Databases in Mobile Ad Hoc Networks.* ICETC '09: Proc. of the 2009 Intern. Conference on Education Technology and Computer, Washington DC, 2009.

[4] OZSU, M. T., VALDURIEZ, P.: *Principles of Distributed Database Systems,* 3rd Edition, Springer, 2011, ISBN 978-1-4419-8833-1.

[5] JANECH, J.: Distributed Database Systems in the Dynamic Networks Environment. *Intern. J. on Information Technologies and Security,* No. 1. Sofia, 2010, ISSN 1313-8251.
[6] JANECH, J., BACA, T.: Distributed database systems in vehicular ad-hoc network. *Communications – Scientific Letters of University of Zilina*, vol. 12, No. 3A, 2010, p. 50–54, ISSN 1335-4205.
[7] BACA, T.: Optimisation of Message Distribution in Ad-hoc Networks. *Information Science and Technologies Bulletin of the ACM Slovakia,* vol. 4, No. 4, 2012 (in print), ISSN 1338-6654.
[8] IEEE Vehicular Technology Society: IEEE Standard for Wireless Access in Vehicular Environments (WAVE) – Networking Services. IEEE Std 1609.3-2010. 2010.
[9] JANECH, J., BACA, T.: Distributed Database Systems in the Dynamic Networks Environment. *Communications - Scientific Letters of University of Zilina,* vol. 13, No. 4, 2011, pp. 72–77, ISSN 1335-4205.
[10] TAMORI, M., ISHIHARA, S., WATANABE, T., MIZUNO, T.: *A Replica Distribution Method with Consideration of the Positions of Mobile Hosts on Wireless Ad-Hoc Networks.* Proc. of the 22nd Intern. Conference on Distributed Computing Systems, IEEE Computer Society, 2002.
[11] LIESKOVSKY, A., JANECH, J., BACA, T.: *Data Replication in Distributed Database Systems in VANET Environment.* IEEE 2nd Intern. Conference on Software Engineering and Service Science (ICSESS), 2011, ISBN 978-1-4244-9696-9.
[12] LIESKOVSKY, A., JANECH, J., BACA, T.: *Data replication in Distributed Database Systems in VANET environment.* ITS Telecommunications (ITST), 2011 11th IEEE Intern. Conference on, 2011, ISBN 978-1-61284-670-5.
[13] CIRSTEA, M. N., DINU, A., KHOR, J. G., MCCORMICK, M.: *Neural and Fuzzy Logic Control of Drives and Power Systems.* San Diego, CA: Elsevier, 2002.
[14] BACA, T.: *Optimalizacia prenosu sprav v ad hoc sietach [Optimisation of Message Distribution in Ad-hoc Networks].* Dissertation thesis, Zilinska univerzita, 2012.
[15] *AdHocSim.FRI - Simulator of VANET Environment,* (online http://adhocsim. dotfri.info/).