Int. J. Electron. Commun. (AEÜ) 81 (2017) 192–204

Contents lists available at ScienceDirect

International Journal of Electronics and Communications (AEÜ)

journal homepage: www.elsevier.com/locate/aeue

Regular paper

Fault tolerant time synchronization using offsets table robust broadcasting protocol for vehicular ad hoc networks

Khedidja Medani *, Makhlouf Aliouat, Zibouda Aliouat

LRSD Laboratory, Faculty of Sciences, Computer Science Department, UFAS1, Sétif, Algeria

ARTICLE INFO

Article history:
Received 4 February 2017
Accepted 21 July 2017

Keywords:
VANETs
Intelligent transportation system
TTD
Fault-tolerant time synchronization
Offset table robust broadcasting

ABSTRACT

The requirement of time synchronization emerged in distributed systems remains one of the most significant issues that should be addressed to the extent of that systems evolve. As clock synchronization is important for any type of network, Vehicular Ad hoc networks (VANETs) are being considered for their basic communication platforms, but also for providing the ability to detect movement, location, proximity, and other network capabilities. The intrinsic characteristics of VANETs like: the high speed of nodes and the lack of permanent network connectivity generated by an instable environment, which make communication difficult or temporarily impossible, have created new challenges. These challenges make solutions that have been already proposed for classical networks no longer appropriate. Therefore, to overcome this deficiency, new and adaptive clock synchronization mechanisms should be devised and implemented, dealing so with communication and scalability issues. In this paper, we propose "Offsets Table Robust Broadcasting" (OTRB) algorithm. In this algorithm, instead to each node communicates with its vicinity, a set of nodes is selected to spread the time information over the entire network. The proposed time synchronization protocol is well-adapted to random network topology changes, high nodal velocity while offering good precision and robustness against nodal failure and packet loss. The analytical study and protocol simulation for evaluating the system performance, carried out by a combination of VanetMobiSim and NS2 simulators, have yielded convincing results, outperforming those exhibited by the basic referred protocols.

1. Introduction

Recent advances in technologies enable a variety of objects, such as laptop computers, smartphones, wearable sensors, vehicles and other smart devices, to interact with each other easily and effectively via wireless ad hoc networks. Wireless ad hoc network cut across many applications that improve and facilitate human lifestyle. Among these applications are the healthcare, industrial, social, transportation and communication industries. The integration of mobile ad hoc networks (MANETs) in transportation means brings a new paradigm called Intelligent Transportation Systems (ITS). The ITS advocates safety, efficiency, conviviality and performance when driving.

Vehicular Ad hoc NETworks (VANETs), a sub-class of MANETs, consider vehicles as mobile nodes. Nodes in VANETs are characterized by high-velocity and predictable movement in open space areas (roads). VANET applications can be divided into two major categories: safety and user applications. The safety applications increase road safety using traffic information systems to prevent accidents and road collisions. The user applications provide additional, interesting and useful on-road services that aim to increase passenger comfort via Internet connectivity, mobile applications, multimedia and peer-to-peer applications.

Recently, VANET design and modeling have drawn significant attention in large scale networks such as future Internet of Vehicles (IoV). The ideal solutions applied in wireless ad hoc networks field cannot be directly integrated in vehicular communications context. This is mainly due to the high-mobility of vehicles, which frequently changes the network topology. Also, roads infrastructure (intersection, traffic jams, and the presence of buildings beside the roads) imposes new constraints, like radio obstacles, the effects of multipath and fading, disconnection in sparse areas, and the bandwidth issue.

VANETs constitute an emergent and attractive research field; however, some obstacles are slowing down their development leading to their lack of maturity. Among these obstacles, we can mention some of issues, which remain without satisfactory

* Corresponding author at: LRSD Laboratory, Faculty of Sciences, Computer Science Department, UFAS1, Sétif, Algeria.
E-mail addresses: khadidja-medani@hotmail.fr (K. Medani), aliouat_m@yahoo.fr (M. Aliouat), aliouat_zi@yahoo.fr (Z. Aliouat).

http://dx.doi.org/10.1016/j.aeue.2017.07.026
1434-8411/

solutions, like: security, connectivity, robustness, reliable communication, and time synchronization. Our study considers time synchronization and to point out the problem let us use an example:

When an urgent event is detected by a node N_i, it will be directly delivered to its neighbors N_j and N_k. Since the node clocks are non-synchronized, then node N_j will get an urgent event notification at 10:11 PM, while node N_k will receive it at 00:01 AM. The serious problem encountered by such a case is that the recipient nodes find difficulties in making consistent decisions about the arriving messages. They cannot distinguish whether the received real-time data is recent or out-of-date. This generates decision-making errors in the environment in case of deleting a recent real-time packet because it appeared to be outdated. For that reason, clock synchronization is one of the most significant issues that must be addressed, especially in distributed environments. To avoid such problems, node clocks must be accurately synchronized to correctly maintain the event causality property.

Time scheduling protocols are put in place to avoid communication collisions. If the distributed node clocks are non-synchronized, time scheduling protocols do not make sense since node clocks are not synchronized with one another.

Synchronization is a vital concern that must be taken into consideration when evaluating a performance system. Communication, coordination, security services all strongly depend on synchronized clocks of the different nodes in the network. A search of specialized literature revealed that the most recent research focused on addressing the following issues: how to rapidly disseminate emergency messages [1,2], how to optimize data transfer from a source to a specific destination [3–8], and how to control channel medium access [9–15]. Also, wireless mobile services and applications have paid more attention last few years [16–20]. Unfortunately, there is a remarkable lack solutions for the clock synchronization problem in VANETs. For such a reason, this paper brings new solution in satisfactory solving such problems in unstable environments like VANETs. A suitable clock synchronization protocol improves VANET requirements. Among these requirements are: low-cost low-message overhead, short time convergence (time convergence is the time taken to synchronize the network), scalability, precision, and long synchronization lifetime. Robustness or fault tolerance are also essential criterion that must be taken into consideration when designing the protocol. Fault tolerance defines the ability of the network to maintain its functionality in case of failure. The accuracy of the synchronization protocol is important, but accurate clock synchronization between neighboring nodes is also of great importance. Most of protocols in state-of-the-art systems are designed to optimize the global skew of nodes using hierarchical architectures (e.g., clustering, like in [21,22], or spanning tree, like in [23]), which ensures that some neighboring nodes are not well synchronized because the error propagates down at different rates on different paths. Indeed, in spanning tree structures, it is very difficult to maintain highly dynamic networks, like VANETs, as every hop increases the synchronization error. On the other hand, in other protocols like RBS [24] and GTSP [25], synchronizing direct neighbors using timing message exchange generates more overhead and leads to transmission channel interfering.

The proposed protocol, called Offset Table Robust Broadcasting (OTRB), provides an accurate and fault-tolerant time synchronization in distributed fashion, focusing on direct neighbor synchronization in which, every node runs with the value of its local clock, but maintains the time information needed to synchronize other nodes for different purposes (e.g., point to point communications, medium access control and security). The solution exploits the broadcasting channel to synchronize neighbor clocks without exchanging any time information between them, which minimizes the communication overhead and minimizes the number of delivered messages to each node.

The remainder of this paper is organized as follows: Section 2 provides an overview about the previous work related to clock synchronization in VANETs. Sections 3 and 4 present our contribution, OTRB. Section 5 discusses the validation and the simulation results. The last section concludes our work and illustrates the future perspectives.

2. Related work

Based on the way that nodes exchange time information, clock synchronization protocols can be classified into the following three approaches: burst position measurement, continuous correlation of timing signals, and clock-sampling methods [26]. Burst position measurement-based protocols rely on the periodic transmission of bursts or pulses. Each node measures the power associated with these pulses, and the delay of the detected pulses with respect of its local burst. The difference is used to correct its local clock. The papers [27–30] are examples of protocols relying on the burst position method, where the purpose is to allow all nodes transmit their periodic pulses simultaneously, to mark the start of the packet data slots. The transmission of pulses has the disadvantage of requiring a large bandwidth, and possibly a dedicated channel [26]. In continuous correlation of timing signal-based protocols, each node continuously transmits a sequence of signals and computes the phase offset according to the receiving sequences. For instance, a clock can drive a Pseudo-Noise sequence generator, and this sequence can be transmitted to other nodes. Many-to-one mapping is needed to extract the correction term used to adjust the local clock. In clock sampling-based protocols, the information exchanged is the clock value. Each node reads the time of its local clock and transmits it to neighboring nodes. When receiving this information, each node performs certain operations, depending on the clock model, to synchronize its local clock. Because they are greater than the burst-position measurement and continuous correlation methods, in term of simplicity, clock-sampling synchronization protocols are widely used in distributed systems [10,21–25,31–36].

Table 1 hereafter compares the related work regarding a valuable criterion on vehicular environment including the scalability of the solution, communication overhead, fault-tolerance, neighbors synchronization, the average error, GPS requirement and the resynchronization period.

- *Reference Broadcast Synchronization (RBS).* The well-known reference broadcast synchronization (RBS) algorithm, introduced by Elson et al., initiates the receiver-receiver message exchange handshake to synchronize neighboring clocks in WSN networks [24]. The main purpose is to benefit from the broadcasting channel to reduce the synchronization error caused by the non-determinism in transmission delays. A sender, called the reference node, broadcasts a beacon to initiate the synchronization process. Each receiver notes the synchronization packet receiving instance, and waits for a random period to exchange the relevant timestamps with its neighbors. Using these timestamps, each node estimates the relative phase offset. To improve the algorithm's accuracy, the reference node broadcasts multiple synchronization packets, and the average of the offset is calculated. However, since each node must exchange its timing information, RBS protocols requires a large amount of communication overhead to synchronize networks with high node density. A large communication overhead influences the protocol's latency. These properties make RBS non-scalable over large scale networks, such as VANET networks.

Table 1
Table of comparison.

Protocol	RBS	CTS	DTT	HCS	Diffusive clock synchronization in highly dynamic networks	JSL
Scalability	No	Yes	Yes	Yes	No	No
Mobility handle	No	Yes	Yes	Yes	Yes	No
Average overhead	$O(n^2)$	$O(n)$	$O(n)$	$O(n)$	$O(n^2)$ burst	$O(n)$
Average error	29.1 μs	–	16.9 μs	–	–	16.9 μs
Fault tolerant	No	No	No	No	No	No
Neighbors synchronization	Yes	Cluster based	Yes	Cluster based	Yes	Tree based
Resynchronization period	When new node arrives	Every 3 s	–	Every 3 s		–
GPS requirement	No	Yes	No	Yes	No	No

- *Converging time synchronization (CTS).* In [21], Saurabh et al. proposed a new scheme that aims to synchronize node clocks in VANETs. The so-called Converging Time Synchronization protocol (CTS) performs time synchronization in master/slave mode to synchronize the largest group of nodes. The algorithm continuously executes the following steps every three seconds. Initially, any node can randomly initiate the synchronization process by broadcasting a synchronization packet to its neighbors. A neighbor's reply will be used by the initiator node to select the sponsor. The sponsor node corresponds to the node with the largest synchronized group. This latter broadcasts an adjustment message to all its neighbors. The adjustment message contains the synchronized group, and the sponsor's time difference with respect of a standard like GSM or UTC time. Each node, when it receives the adjustment message, updates its synchronized group and adjusts the time deviation in accordance to the sponsor node.
  Dahlia et al., in [22], adopt the idea of the CTS protocol to synchronize nodes' clocks in a hybrid VANETs architecture by using the Hybrid Clock Synchronization (HCS) protocol. In the proposed architecture, nodes can be ordinary vehicles, sensors, or road-side units (RSUs).
  In contrast to the RBS protocol, CTS and HCS protocols show better performances over the density and the velocity of vehicles. Also, new arrival nodes could simply get synchronized to the existing group, while RBS algorithm must restart the synchronization process whenever new vehicles arrive. Nevertheless, simulation results show that the algorithm latency converges with the nodes' velocity. As mentioned earlier, both CTS and HCS protocols aim to synchronize the largest group of nodes with one another. But, neighboring nodes, which are not in the same group, are not well synchronized, which increase the average synchronization error.
- *Diffusive clock synchronization in highly dynamic networks.* In [35], the authors focused their work on presenting a diffusive clock synchronization algorithm that communicates via pulses only. Computation evolves in rounds. At each round k, a node i broadcasts its pulse and waits for its local clock to increase by a constant R before adjusting to a correct offset. This action assumes that a node i always receives its own pulses. The generation of the round $k+1$ pulses depends only on the received k pulses. To ensure that all nodes receive all round k pulses of their incoming neighbors before they broadcast their round $k+1$ pulses, the term R must be large enough. The correcting term is a weighted average of the time differences of received round k pulses:

$$t_i(k+1) = t_i(k) + T_i(k+1) + corr_i(k+1)$$

  where $t_i(k)$ is the real-time node i broadcasts its own pulse,

$$(1-\varrho)R \leqslant T_i(k+1) \leqslant 1+\varrho)R$$

  And, $\varrho \in [0,1]$ is the drift of the clock.
  Therefore, each node transmits its pulse to other nodes. In addition to the large communication overhead, the transmission of pulses requires large bandwidth.
- *Joint Synchronization Localization (JSL).* A new scheme that jointly issues the time synchronization and the localization problems in under-water sensor networks (UWSNs) was proposed by the authors of [23]. The proposed protocol, called Joint Synchronization and Localization (JSL) performs the synchronization and the localization in rounds. At each round, the output of the synchronization is sent back as the input/outputs of the localization is fed back as the input of synchronization exits for the next round. The network architecture consists of surface buoys equipped with a GPS, anchor nodes, powerful sensor nodes, which can communicate with the surface buoys, and ordinary nodes, which are those sensor nodes used for autonomous underwater vehicles (AUVs). Both surface buoys and anchor nodes act as reference nodes. Considering the critical challenges brought about by the nature of the environment, such as the stratification effects of the water medium on the propagation delay and nodal mobility, the algorithm performs the synchronization in three phases: propagation delay estimation, linear regression and propagation delay update. In linear regression phase, sender/receiver messages are exchanged between the reference and the ordinary nodes. The timestamps collected are used to update the estimated clock skew and the offset. The sender/receiver mechanism, used to exchange the timing information between the reference and any other node, generates more communication overhead. The number of messages will increase to maintain the frequent change on the network topology, mainly due to the high mobility of nodes in such environments, like VANETs.
- *Time Table Diffusion protocol.* The Time Table Diffusion (TTD) protocol proposed by Medani et al., provides time synchronization among the nodes in vehicular ad hoc network in distributed fashion [34]. The fact that each node exchanges its time information, like in RBS and JSL protocols, generates more communication overhead. The solution improves the ability of nodes to synchronize without any message exchanged between them. The main idea is to set a group of nodes to spread the time information over the networks. In general, TTD algorithm works in four principle phases. A set of nodes, called transporter nodes, broadcast a synchronization packet to initiate the synchronization process. The sender's receiver mechanism is used by the transporter node to estimate the clock's offset relative to its neighbors. Then, the transporter node broadcasts these offsets to its neighbors, allowing them to synchronize with each other without message exchange. Simulation results show that the proposed protocol has better performance in term of time convergence and message overhead with respect to the main constraining properties of VANETs, which are the high density

and velocity of nodes. However, the protocol lack of the robustness over nodes failure.

The work presented in this paper attempts to eliminate the drawbacks of the related work. Among these drawbacks are: large communication overhead and time convergence, synchronization of neighbor nodes, and the lack in fault-tolerance. The solution is an extension of our previous work TTD [34]. In the following sections, we aim to analyze TTD solution, which our goal is to enhance TTD performance in case of nodes failures. The enhanced OTRB protocol considers failures in both communication and clock models.

3. Time synchronization using time table diffusion protocol

Time table diffusion protocol (TTD) provides time synchronization in vehicular ad hoc networks independently of the network topology [34]. The main idea is to set a group of nodes to spread the timing information. Based only on vehicle to vehicle communications (V2VC), TTD algorithm performs time synchronization in the following steps. Initially, any node can randomly initiate the synchronization process and broadcasts a synchronization packet to its two-hops neighbors. Other nodes, mark the packet receiving instance using their local clocks and send a reply message. The initiator, say transporter node, relies on the offset delay estimation method to calculate the offset relative to node *i* (see Fig. 1), as well as, the offsets relatives to all its neighbors.

The initiator node broadcasts the time information, in table form, to its neighbors, allowing them to synchronize each other without any message exchange. Each node runs repeatedly while the algorithm runs using a fixed amount of time to deal with the mobility of nodes and to prevent clock drift. Fig. 2 hereafter encapsulates the synchronization process using the TTD protocol in a single round.

As in the example shown by Fig. 3, nodes 0 and 5 are two transporter nodes. They broadcast a synchronization request to its neighbors 1, 2, 3, 4 and 1, 3, 4, 6 respectively. After they receive all the responses from its neighbors, the transporter nodes estimate and broadcast the offsets relative to its neighbors. Fig. 3 shows the synchronized group of each node. Nodes 1, 3 and 4 synchronize with the two groups because they are in the scope of the two transporter nodes.

The most important requirement in VANET protocol design the convergence time and the average overhead with respect to the parameter of vehicles density. The number of messages generated by the synchronization process is important because it may influence the network behavior positively (i.e., messages flow smoothly and easily over the network) or negatively (i.e., messages contribute to network congestion). The convergence time is the time it takes to synchronize the network. For analyzing the

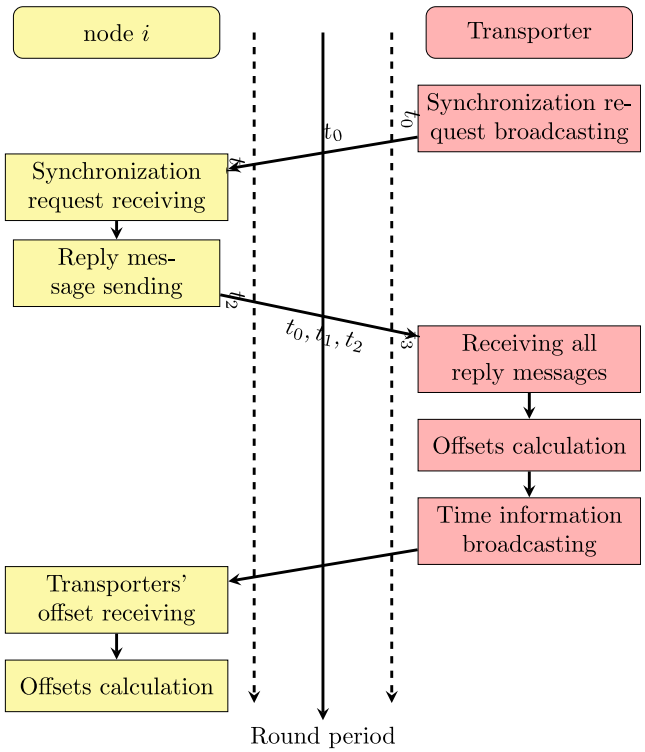


Fig. 2. TTD protocol flowchart.

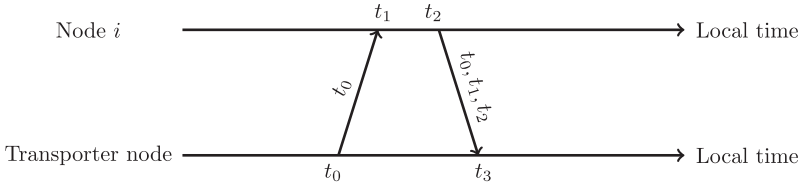
performances of the TTD proposal, we calculate the number of messages and the convergence time as below:

- The number of messages (*nbMsg*) generated to execute the TTD algorithm is estimated as follows:

$$nbMsg = \sum_{i=0}^{n_t-1}(N_i + 2) \quad (1)$$

where n_t is the number of transporter nodes in the current round, and N_i is the number of neighbors of the transporter node.

Suppose there are n_t number of transporter nodes in the current round. The number of messages generated to initiate the synchronization process is equal to n_t, i.e., the number of transporter nodes. Neighbors responses phase generates several messages equal to the number of neighbors of all the transporter nodes, $\sum_{i=0}^{n_t-1} N_i$. In addition, each transporter node broadcasts its time table, which generates n_t time table messages. The total number of messages generated to accomplish the synchronization process is given by:



$$Offset = ((t_3 - t_2) - (t_1 - t_0))/2$$

The transporter node estimates the clock value of node i following the offset delay estimation method as follows:

$$C(\mathrm{i}) = C(Transporter) + Offset$$

Fig. 1. Offset delay estimation mechanism.

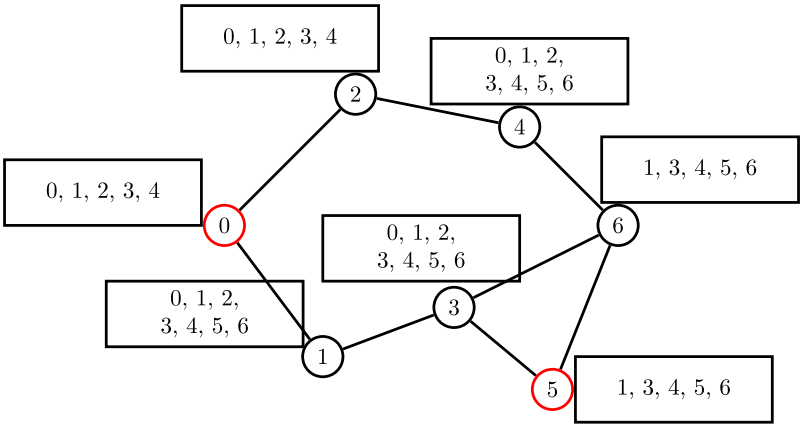


Fig. 3. Illustrative example explaining TTD protocol.

$$nbMsg = n_t + \sum_{i=0}^{n_t-1} N_i + n_t$$

After simplification, we obtain:

$$nbMsg = \sum_{i=0}^{n_t-1} (N_i + 2)$$

- To synchronize a network having N number of nodes, TTD algorithm must take place within the total time:

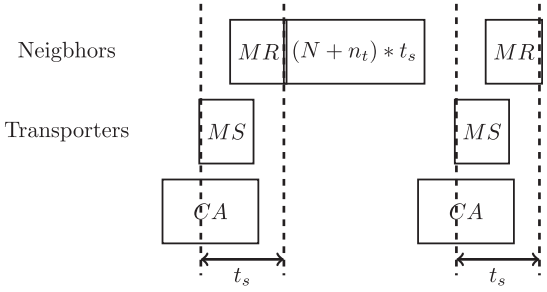
$$t_{total} = (N + n_t + 4) * t_s \quad (2)$$

where t_s is the maximum time required to deliver one message. The timing sequence relative to each operation of the TTD protocol is given in Fig. 4.

Figs. 5 and 6 illustrate the change in the number of messages and the convergence time with respect to node number.

Note that the number of generated messages is closely related to the convergence time. When the former increases, it contributes to the network's congestion. A congested network with collision produces message loss, which in turn slows down the convergence time.

Another evaluation metric that should be considered is the synchronization rate. The synchronization rate refers to the rate of nodes that can become well synchronized. The frequency of departure of the vehicles lead to an increase in dropped communication links during the synchronization task. On the other hand, the new



Where:

- CA refers to thhe collision avoidance period. At max $CA = t_s$;
- MS = message sending;
- MR = message receiving;
- $MS/MR = t_s$.

Fig. 4. The timing sequence of TTD protocol.

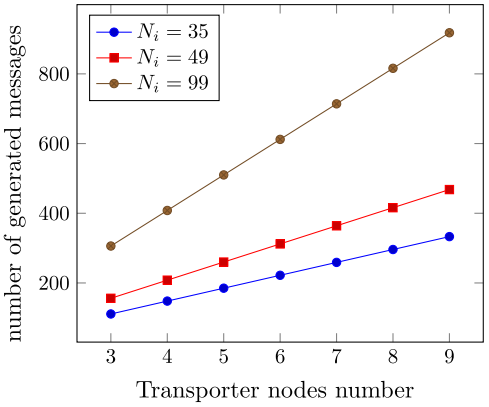


Fig. 5. Messages complexity TTD solution.

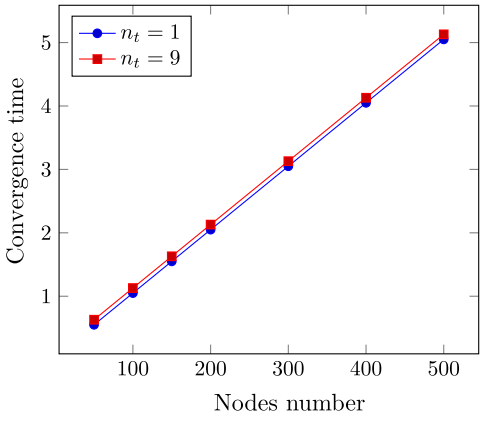


Fig. 6. Convergence time in TTD solution.

arrival of vehicles creates new communication links between the nodes. A reliable communication is quietly needed to synchronize moving vehicles, which in turn increases the synchronization rate. The proposed protocol would be able to face those challenges (new arrival node, link life time, etc.). The TTD protocol, as described in [34], does not consider the reliability of message exchange, nor does it consider the synchronization of new arrival nodes, which affects its reported synchronization rate metric.

In the following sections, we describe our contribution of fault-tolerant TTD, called Offset Table Robust Broadcasting (OTRB). The OTRB solution considers the synchronization of new arrival nodes

and studies the effect of dropped links. Additionally, the aim of the fault-tolerance in our setting is to prevent the fault-time synchronization information in the network, and to ensure the appropriate behavior of the solution in the presence of faulty nodes and communication channels.

4. Contribution

The enhanced OTRB protocol provides efficient time synchronization in VANET, thus optimizing message complexity. The system consists of vehicles communicating with each other over multiple wireless hops by using an embedded WiFi card. Inter-vehicle communications may or may not include a Road Side Unit (RSU) access point. Vehicles communicate with each other through Vehicle to Vehicle communication (V2VC), as well as they communicate with the available RSU through Vehicle to Infrastructure Communication (V2IC). The network relies on the ability of transmission time synchronization messaging to perform the time synchronization task. The messages are sent via a wireless channel based on the *IEEE* 802.11 standard. We assume that the wireless channel is symmetric, e.g., if node A hears node B, then node B can also hear node A. For time synchronization, we adopt the following model:

- Initially, each node has a unique identity in the network, and maintains a list of neighbors, that are nodes within its transmission range.
- Every node has a notion of time that is based on the oscillation of crystal quartz. The clock time, $C(t)$, is the time reported by the clock at the real time t. For an ideal clock, the clock time $C(t)$ is equal to the real time t. The clock offset is defined as the difference between the clock time and the real time $(C(t) - t)$.
- The offset of the clock C_j relative to the clock C_i in real-time t is given by:

$$\Delta_{ij} = C_i(t) - C_j(t) \tag{3}$$

Or:

$$\Delta_{ij} = \Delta_{ik} + \Delta_{kj} \tag{4}$$

Table 2
Clocks' notations table.

Notation	Value
$C_i(t)$	The time value shown by the local clock of node i at the real instance t
Δ_{ij}	The clock offset of node j relative to node i
ρ	The maximum clocks' drift defined by the manufacturer
ϵ	The average error tolerated by the application
Δt	The round period, that defines the validity period of the calculated clocks' offsets

- A node is considered well-behaved if the time shown by its local clock is bounded by the maximum drift (ρ), which is defined by the constructor.
- Each node moves freely per its local clock but maintains a time table containing the offsets related to neighboring nodes.
- A node is considered well-synchronized if the time difference between this node and each one of its neighbors does not exceed an acceptable average ϵ, which depends on the application.
- For mobility management reasons, and since the clock drifts naturally, achieving a consistent synchronization is important. Each synchronization period consists of Δt time units. The re-synchronization Δt value must be large enough to deal with all nodal mobility and clock drift issues (see Table 2).

The following sections describe the algorithm in detail.

4.1. OTRB algorithm

The enhanced OTRB algorithm performs time synchronization message-passing by using four principal phases. The initialization phase consists of the *transporter node selection*. The latter broadcasts a synchronization packet to request its two-hop neighbors to participate in the synchronization process and to collect the timing information. In the second phase, the *information retrieval phase*, the neighbors synchronize their local clocks and then reply to the transporter nodes within their scope. The transporter node uses the timing information received to *calculate an offset table and update the rounded period value* (Δt). Finally, the broadcasting of the calculated offsets table by the transporter allows other nodes to calculate their own offsets and synchronize with its neighbors, which is the *synchronization phase*.

Each normal node runs continuously via these four steps every Δt time unit (see Fig. 7).

4.1.1. Transporter node selection phase

The transporter node selection phase strongly affects the algorithm's performance. Hence, the better the choice of transporter node, the better algorithm performs. Each transporter node and in-range neighboring nodes form a small network unit, or cluster. Selecting the wrong transporter node may cause an instability of the cluster. In the cluster, the transporter node is the cluster head (CH) and its neighbors are the cluster members (CMs). The vehicles that participate in the synchronization process under multiple transporters are called gateway members (GMs) (see Fig. 8).

To improve the algorithm performance, the cluster formation should be as stable as possible; this means that the transporter node should meet some basic requirements, including some that set limits to the position and the velocity of the individual nodes. For this reason, the proposed method adopts a novel clustering algorithm [37–42].

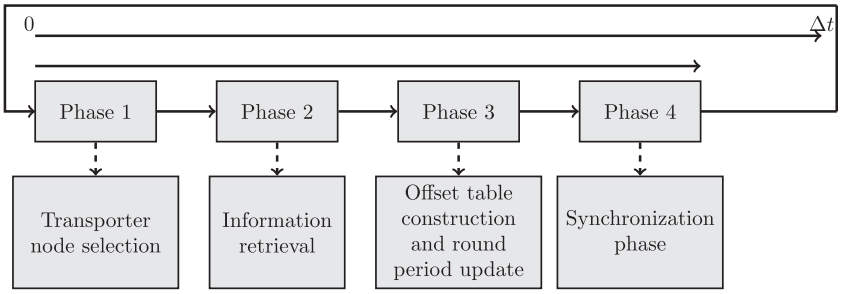


Fig. 7. OTRB protocol phases.

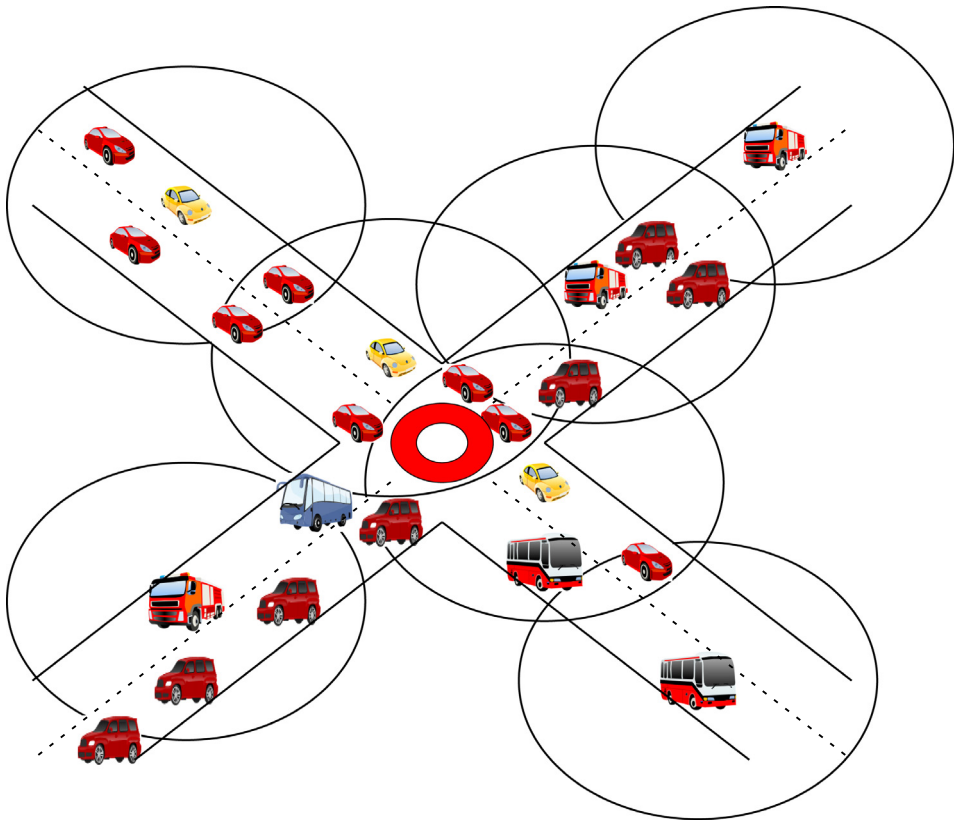


Fig. 8. Network partitioning.

The OTRB protocol can also adopt the transporter node selection with and without RSUs. If an RSU is available, the latter acts as a coordinator to set transporter nodes with respect to a set of conditions.

4.1.2. Information retrieval phase

In this phase, messages between the transporter node and its neighbor nodes are exchanged. The time they take to send and receive a reply message is used to estimates the offsets and update the rounded period value Δt.

Algorithm 1. Information retrieval algorithm

```
1  if isTransporter(i) then
2      TID ← i; t0 ← now_time; NT_ID ← nearest(i);
3      broadcast ADV − T(TID, t0, NT_ID)
4      for each JOIN − RESPONSE received from node j do
5          t3j ← now_time
6          t1j ← t1
7          t2j ← t2
8      end
9  else
10     wait(ADV − T)
11     for each ADV − T received do
12         t1 ← now_time
13         insert(MyTransporterList, TID)
14         wait(random_period)
15         t2 ← now_time
16         send JOIN − RESPONSE(TID, i, t0, t1, t2)
17     end
18 end
```

Each transporter node broadcasts a synchronization packet, called an advertisement message (ADV-T), which is timestamped by its own local clock. The synchronization packet contains the transporter node identity (*TID*) and the timestamps t_0. The transporter node also sends the identity of its nearest node, NTID. The nearest node will be used to improve the reliability of the message exchange, it can also calculate and broadcast the offsets table in case of transporter node failure.

For any other node, say for example node *i*, when node *i* receives the ADV-T message from one transporter node, it marks its local clock at t_1 and then sends a reply message, called a JOIN-RESPONSE message. The reply message contains the identity of the node i, the identity of the transporter node *TID* and the timestamps t_0, t_1 and t_2, where t_2 indicates the JOIN-RESPONSE message sending instance. Each node must 'reply to all' so that the ADV-T messages is received by the entire cluster.

4.1.3. Offsets table construction and round period update

After receiving the JOIN-RESPONSE message from each one of its neighbor nodes, the transporter records the time of its local clock at the instance t_{3i}. By using the timestamps founded in the reply messages, the transporter node relies on the offset delay estimation method (see Fig. 1) to calculate the clock offsets relative to its neighbors.

$$\Delta_{(i\_TID)} = \frac{(t_1 - t_0) - (t_{3i} - t_2)}{2} \tag{5}$$

The obtained result is stored in a time table as:

$$time\_table_{TID}(i) = \Delta_{(i\_TID)} \tag{6}$$

Also, considering only non-bad clocks and the average error ϵ, the transporter node updates the round period Δt value as follows:

$$\Delta t = \frac{(n-r)\epsilon - 3m\epsilon}{2(n-r)\rho} \tag{7}$$

For any Δt, two well-behaved clocks may drift away at most by $\delta = 2\rho$, where ρ here is the maximum drift of all normal nodes defined by the manufacturer. In distributed systems, it is possible to have clocks synchronized with a maximum average ϵ, if no more than one-third of the clocks are bad or byzantine. Bad clocks exhibit large drift, they can be easily identified and their time values will be removed. Byzantine clocks are inconsistent nodes that transmit different timestamps to different nodes at the same time. The presence of one byzantine clock in a distributed system with n nodes can make any two arbitrary nodes to differ by $\gamma = \frac{3\epsilon}{n}$ sec. If the maximum drift between any two clocks is restricted to ϵ, the transporter node can calculate the rounded period Δt as follows:

Suppose there are n nodes within the transmission range of the transporter node in which m byzantine clocks, and r defective clocks, where $m + r \leqslant \frac{n}{3}$. So, the time difference increases from $\frac{3m\epsilon}{n-r}$ to ϵ is:

$$2\rho\Delta t = \epsilon - \frac{3m\epsilon}{n-r}$$

By simplification, the transporter node calculates the rounded period value as:

$$\Delta t = \frac{(n-r)\epsilon - 3m\epsilon}{2(n-r)\rho}$$

Algorithm 2. Offsets table construction and round period update algorithm

```
1  Initialization : n ← 0; r ← 0; ρ ← 10^−4;
2  for each join_msg received do
3  |  Δ_TID,j ← ((t_3i − t_2i) − (t_1i − t_0i))/2
4  |  if |Δ_TID,i| ≥ ε then
5  |  |  r ← r + 1
6  |  |  insert(BadClockList, i)
7  |  else
8  |  |  n ← n + 1; time_table_TID(i) ← Δ_TID,i
9  |  end
10 end
11 Δt ← (n∗ε−3∗r∗ε) / (2∗n∗ρ)
12 Broadcast TIME − TABLE(TID, time_table_TID, BadClockList, NT_ID, Δt)
```

Algorithm 3. Synchronization phase algorithm

```
1  for each TIME − TABLE received do
2  |  if TID ∈ MyTransporterList & i ∉ BadClockList then
3  |  |  time_table_i(TID) ← −time_table_TID(i)
4  |  |  for eachnodej ∈ time_table do
5  |  |  |  time_table_i(j) ← time_table_TID(j) + time_table_i(TID)
6  |  |  end
7  |  |  if i = NT_ID then
8  |  |  |  t_4 ← now_time
9  |  |  |  broadcastACK(TID, NT_ID, t_4)
10 |  |  end
11 |  end
12 |  if TID ∉ MyTransporterList then
13 |  |  send request_sync(i, gateway)
14 |  end
15 end
16 if no TIME − TABLE received then
17 |  wait(ACK_period)
18 |  if ACK received then
19 |  |  sendrequest_sync
20 |  else
21 |  |  insert(nonTrustList, TID)
22 |  end
23 end
```

To get synchronized, when the node i receives the time table, it will proceed as follows:

- Initially, to synchronize with the transporter, node i estimates the offset relative to this latter as:

$$time\_table_i(TID) = -time\_table_{TID}(i) \tag{8}$$

We know that the offset of node i relative to the transporter node TID, Δ_{TID_i}, is given by:

$$\Delta_{TID\_i} = C_i(t) - C_{TID}(t)$$

Multiplying both sides of this equation by (-1), yields:

$$-\Delta_{TID\_i} = -C_i(t) + C_{TID}(t)$$

$$\Rightarrow -\Delta_{TID\_i} = C_{TID}(t) - C_i(t) = \Delta_{i\_TID}$$

$$\Rightarrow \Delta_{TID\_i} = -\Delta_{i\_TID}$$

Table 3 hereafter lists the type, number and content of all the message exchanged during the synchronization process.

4.1.4. Synchronization phase

At this phase, the transporter node broadcasts the updated offsets table to all its neighbor nodes allowing them to calculate their own one.

Table 3
Exchanged messages notation table.

Message notation	Designation	Content	Number
ADV-T	The advertisement message sent by the transporter node to initiate the synchronization process	• *TID*, the identity of the transporter node • *NT_ID*, the identity of the nearest node of the transporter • t_0, local time in the transporter node.	n_t, the number of transporter nodes in the current period
JOIN-RESPONSE	The reply message send by neighbors of the transporter node	Timestamps: $t_0,\ t_1,\ t_2$	$\sum_{i=0}^{n_t} N_i$
TIME-Table	The offsets table sent out by the transporter node	The offsets table and the round period Δt	n_t
ACK	The acknowledgment message sent out by *NT_ID* node to confirm the time table deliverance	*TID*, the identity of the transporter node, and the timestamps t_4	n_t

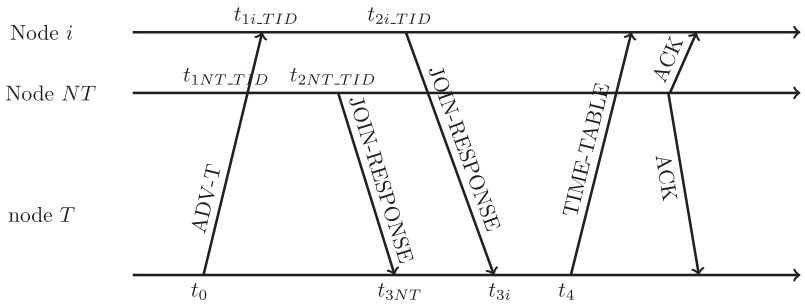


Fig. 9. Message exchange during one round in the synchronization process.

$$\Rightarrow time\_table_i(TID) = -time\_table_{TID}(i) \quad \square$$

- After estimating the clock offset relative to the transporter node, node i relies on Eq. (4) to calculate the clock offsets relative to each node j in the received table as follows:

$$time\_table_i(j) = time\_table_{TID}(j) + time\_table_i(TID) \tag{9}$$

Also, the nearest node of the transporter broadcasts an acknowledgment message (ACK) to confirm the time table delivery. The absence of messages or responses from the nearest node (*NT*) during a threshold time is used as an indicator of failed offsets. Fig. 9 illustrates the message exchange during the synchronization process in one round.

There are two cases that should be considered:

1. If node i doesn't receive any time table from the transporter node, it waits for a threshold time to receive the ACK message.
   - If no ACK message delivered, because of a faulty transporter node, node i updates its list of non-trusted transporters. The non-trust list contains the identities of all transporter nodes joined but node i does not receive any time table sent by them. If the occurrence rate of the transporter node exceeds a certain threshold value, node i affirms that this transporter node is no more trust again (Fig. 10 illustrates the comportment of node i when receiving the offsets table).
   - If node i receives the acknowledgment message, which means node i has been moved out of transmission range of the transporter node. In that case, node i sends a request message to the nearest gateway member.
2. Whenever new vehicle enters the group, it will be synchronized to the existing group by getting the offsets table from the gateway member or from its nearest neighbor. Contrary to the RBS protocol, it is not possible to synchronize new arrivals joining the group; therefore, all vehicles should restart the synchronization process, which may incur some drawbacks.

5. Performances analysis

To evaluate protocol performance, the proposed protocol OTRB is implemented using the NS2 simulator. The channel's physical characteristics are according to the specification of 802.11b, with channel bit rate of 11 Mbps. Many scenarios were used in which important parameters have been modified, such as the number of nodes and their speeds. Thus, at first, nodes were assigned to random locations and they moved according the mobility model, named the Intelligent Driver Model including Lane Change (IDM-LC) [43], made available by VANET MOBIlity SIMulator (VanetMobiSim). IDM-LC implements road intersection supervising strategy: making vehicle nodes slow down, stop or move in accordance with traffic lights. They are also capable of overtaking to change lane in multi-lane roads.

Regarding failure, we consider in this step, those occurring from the model used for communications. Thus, ten percent of the nodes are randomly considered as failing. In the simulation phase, clocks were assigned random values and generated in accordance to Gauss law [44] with $average = 0$ and $\delta = 10$ ppm. We fixed the maximum drift ρ at the value 10^{-4}.

In our setting, the delegation of the transporter node has been made in a distributed manner by considering the main parameters, which make the cluster as stable as possible. These parameters include the velocity of the transporter node, its position, and the number of its neighbors. The velocity of the node should be averaged according to the environment and the speed the other vehicles are moving. Also, we should ensure that the transporter node will not leave its neighboring during the synchronization process. Therefore, the elected transporter must be located at the center of its neighboring nodes. The number of neighbors the transporter node has defines the cluster density and so affects the communication overhead. The number of neighbors of the transporter node depends on the transmission range. It is more suitable that this parameter should be average to minimize the communication overhead

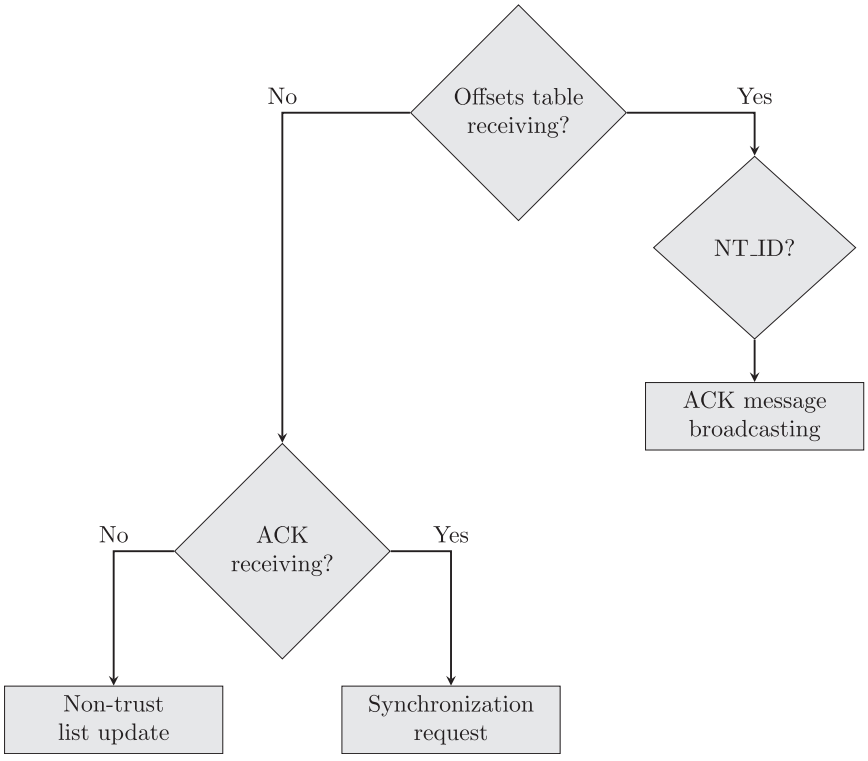


Fig. 10. Receiving offsets table by node *i*.

Table 4
Simulation parameters.

Topology (m^2)	$10{,}000 * 10{,}000$
Nodes number	100
Average speed (m/s)	15, 20, 25
Traffic light	6
Mobility model	Randomly according to IDM_LC with 2 obstacles every 100 m^2
Range data transmission (m)	250
Nodes' failure	10% of nodes
ρ parameter	10^{-4}
ϵ parameter	$10 * 10^{-3}$, $5 * 10^{-3}$, $2 * 10^{-3}$
Simulation time (s)	3600

and to facilitate a reliable and fast broadcasting of the offsets table (see Table 4).

5.1. Message complexity

The results from the simulation showed that the number of messages needed to achieve the time synchronization relies on two main factors: number of moving vehicle nodes (including the transporter), which is also related to the communication range. Indeed, the number of required messages increases with the number of active nodes (see Fig. 11). This results from message replies sent out by neighboring nodes, which constitute the most messages produced by a phase in the time synchronization algorithm.

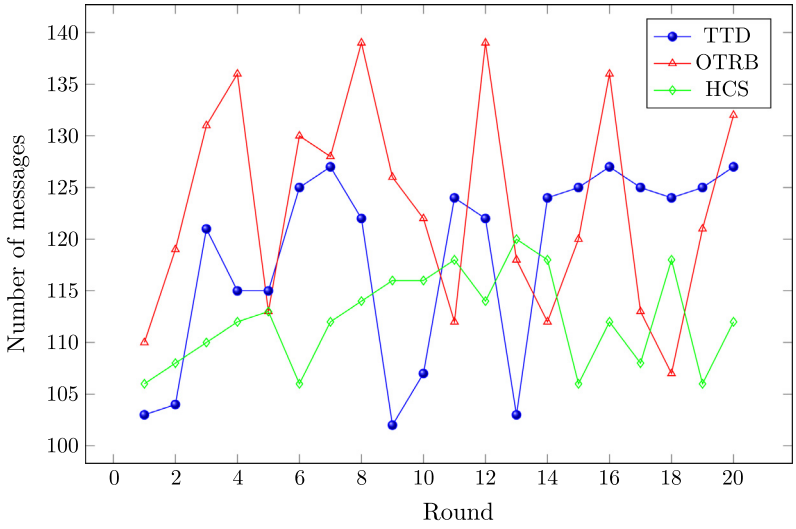


Fig. 11. Message's number in different rounds.

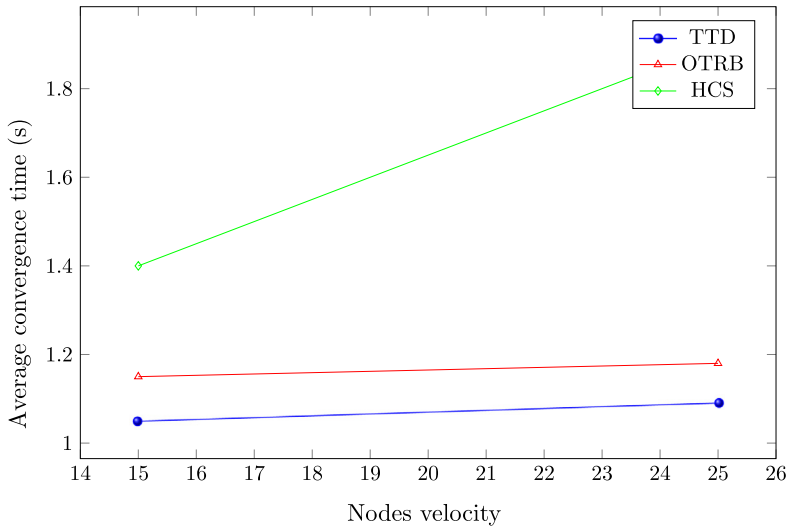


Fig. 12. Average convergence time.

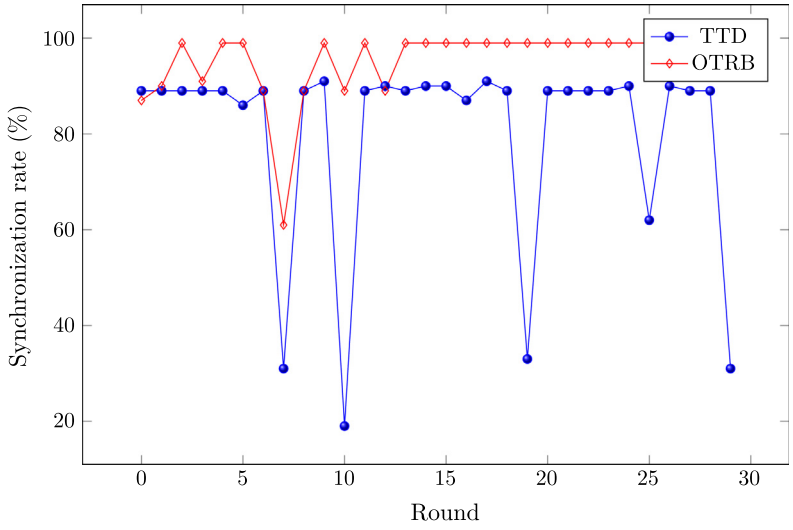


Fig. 13. Synchronization rate.

5.2. Convergence time

The results provided by the simulation point out that when the number of nodes increases, it leads to the increase of the time convergence for OTRB. This situation is due to the number of involved nodes, which consequently generate a large number of neighbor replies (Fig. 12). The average convergence time is practically the same as in the TTD protocol (see [34]).

5.3. Synchronization rate

The synchronization rate is an important metric to evaluate the robustness of the proposed solution, which defines the ability of the protocol to perform time synchronization for the set of nodes in the network and to ensure reliable communication. As the protocol achieves the time synchronization per round, and the latter is calculated with respect to the clock drift, the synchronization rate is not incremental as our proposition hypothesized, but is calculated in each round separately. The synchronization rate in the TTD protocol can decrease in case of high node mobility and the instability of the cluster or the synchronized group (random transporter node). The simulation results show that the synchronization rate rises 100% in normal cases, but decreases to 90% with the injection of fault nodes, and to $(90 - x)\%$ if one of them is a transporter node with x neighbors. By contrast, in the OTRB protocol, the synchronization rate is improved by (i) the stabilization of the cluster construction; and (ii) the use of gateway members to synchronize the new arrival nodes (see Fig. 13).

5.4. The average synchronization error

The average synchronization error is given by averaging the time-difference between every pair of nodes. Fig. 14 shows that the average synchronization error of the proposed protocol is bounded by the average error defined by the application (see Fig. 14).

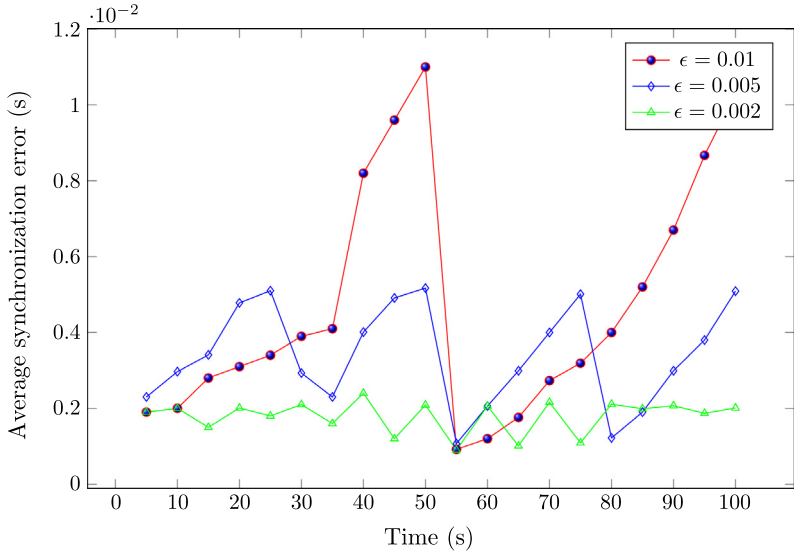


Fig. 14. Average synchronization error.

6. Conclusion

Clock synchronization is a critical issue in VANETs for diverse crucial purposes including communication, data fusion, coordination, real-time safety applications, localization, and for many other applications. In order to face the technical challenges due to the constraints of instable VANET environments, we have proposed in this paper, a new time synchronization protocol referred to as Offsets Table Robust Broadcasting (OTRB). The proposed mechanism provides robust time synchronization with high accuracy. The robustness is owed, in one side, to the use of the broadcasting channel and, in the other side, to the acknowledgment message that ensures the offset table delivery by all nodes. The clock offsets are calculated using round-trip time mechanism, which offers a high accuracy with an average error of 16.9 µs. The performance evaluation of the proposal was carried out using an analytical model and different pertinent simulation scenarios. The results obtained by both approaches concur and demonstrate that OTRB outperforms similar protocols in terms of synchronization rate, convergence time and messages complexity, preventing so network congestion and communication capabilities degradation. Even, while the proposed protocol operates by adopting the existing clustering algorithms, our future work includes the implementation and the comparison with some protocols in order to evaluate the algorithm performance.

References

[1] Chitra M, Siva Sathya S. Efficient broadcasting mechanisms for data dissemination in vehicular ad hoc networks. Int J Mob Network Commun Telem 2013;3(3):47–63.
[2] Fasolo Elena, Zanella Andrea, Zorzi Michele. An effective broadcast scheme for alert message propagation in vehicular ad hoc networks. IEEE international conference on communications, 2006, ICC'06, vol. 9. IEEE; 2006. p. 3960–5.
[3] Kim Jung-Hun, Lee SuKyoung. Reliable routing protocol for vehicular ad hoc networks. AEU-Int J Electron Commun 2011;65(3):268–71.
[4] Zhang Degan, Li Guang, Zheng Ke, Ming Xuechao, Pan Zhao-Hua. An energy-balanced routing method based on forward-aware factor for wireless sensor networks. IEEE Trans Ind Inform 2014;10(1):766–73.
[5] Zhang De-gan, Song Xiao-dong, Wang Xiang, Ma Yuan-ye. Extended aodv routing method based on distributed minimum transmission (dmt) for wsn. AEU-Int J Electron Commun 2015;69(1):371–81.
[6] Zhang De-gan, Zheng Ke, Zhang Ting, Wang Xiang. A novel multicast routing method with minimum transmission for wsn of cloud computing service. Soft Comput 2015;19(7):1817–27.
[7] Zhang De-gan, Wang Xiang, Song Xiao-dong, Zhang Ting, Zhu Ya-nan. A new clustering routing method based on PECE for WSN. EURASIP J Wirel Commun Network 2015;2015(1):1–13.
[8] Zhang De-gan, Liu Si, Zhang Ting, Liang Zhao. Novel unequal clustering routing protocol considering energy balancing based on network partition & distance for mobile education. J Network Comput Appl 2017;88:1–9.
[9] Park Chan-Ki, Ryu Min-Woo, Cho Kuk-Hyun. Survey of mac protocols for vehicular ad hoc networks. SmartCR 2012;2(4):286–95.
[10] Khurana Manju, Thalore Ranjana, Raina Vikas, Jha Manish Kumar. Improved time synchronization in ml-mac for WSN using relay nodes. AEU-Int J Electron Commun 2015;69(11):1622–6.
[11] Nasrallah Yamen Y, Al-Anbagi Irfan, Mouftah Hussein T. Distributed time synchronization mechanism for large-scale vehicular networks. In: 2016 International conference on selected topics in mobile & wireless networking (MoWNeT). IEEE; 2016. p. 1–6.
[12] Nasrallah Yamen Y, Al-Anbagi Irfan, Mouftah Hussein T. Qos-based distributed time synchronization mechanism for high intensity vehicular networks. In: 2016 IEEE symposium on computers and communication (ISCC). IEEE; 2016. p. 958–63.
[13] Zhang Degan, Wang Xiang, Song Xiaodong, Zhao Dexin. A novel approach to mapped correlation of id for rfid anti-collision. IEEE Trans Serv Comput 2014;7(4):741–8.
[14] Zhang Degan, Zhao Chen-peng, Liang Yan-pin, Liu Zhao-jing. A new medium access control protocol based on perceived data reliability and spatial correlation in wireless sensor network. Comput Electr Eng 2012;38(3):694–702.
[15] Zhang De-gan, Zheng Ke, Zhao De-xin, Song Xiao-dong, Wang Xiang. Novel quick start (QS) method for optimization of TCP. Wirel Netw 2016;22(1):211–22.
[16] Zhang De-gan, Zhang Xiao-dan. Design and implementation of embedded uninterruptible power supply system (eupss) for web-based mobile application. Enter Inform Syst 2012;6(4):473–89.
[17] Zhang De-Gan. A new approach and system for attentive mobile learning based on seamless migration. Appl Intell 2012;36(1):75–89.
[18] Zhang De-gan, Liang Yan-pin. A kind of novel method of service-aware computing for uncertain mobile applications. Math Comput Model 2013;57(3):344–56.
[19] Zhang De-gan, Song Xiao-dong, Wang Xiang, Li Ke, Li Wen-bin, Ma Zhen. New agent-based proactive migration method and system for big data environment (BDE). Eng Comput 2015;32(8):2443–66.
[20] Zhang Degan, Kang Xuejing, Wang Jinghui. A novel image de-noising method based on spherical coordinates system. EURASIP J Adv Signal Process 2012;2012(1):110.
[21] Wang Shizhun, Pervez Anjum, Nekovee Maziar. Converging time synchronization algorithm for highly dynamic vehicular ad hoc networks (vanets). In: 2010 7th International symposium on communication systems networks and digital signal processing (CSNDSP). IEEE; 2010. p. 443–8.
[22] Sam Dahlia, Velanganni Cyrilraj, Esther Evangelin T. A vehicle control system using a time synchronized hybrid vanet to reduce road accidents caused by human error. Vehic Commun 2016;6:17–28.
[23] Liu Jun, Wang Zhaohui, Cui Jun-Hong, Zhou Shengli, Yang Bo. A joint time synchronization and localization design for mobile underwater sensor networks. IEEE Trans Mob Comput 2016;15(3):530–43.
[24] Elson Jeremy, Girod Lewis, Estrin Deborah. Fine-grained network time synchronization using reference broadcasts. ACM SIGOPS Oper Syst Rev 2002;36(SI):147–63.

[25] Sommer Philipp, Wattenhofer Roger. Gradient clock synchronization in wireless sensor networks. In: International conference on information processing in sensor networks, 2009. IPSN 2009. IEEE; 2009. p. 37–48.
[26] Rentel Carlos H. Network time synchronization and code-based scheduling for wireless ad hoc networks PhD thesis. Carleton University Ottawa; 2006.
[27] Ebner André, Rohling Hermann, Lott Matthias, Halfmann W. Decentralized slot synchronization in highly dynamic ad hoc networks. The 5th international symposium on wireless personal multimedia communications, 2002, vol. 2. IEEE; 2002. p. 494–8.
[28] Ebner André, Wischhof Lars, Rohling Hermann. Aspects of decentralized time synchronization in vehicular ad hoc networks. In: Proc 1st int WIT. p. 67–72.
[29] Kim Young An, Hong Choong Seon. Autonomous decentralized synchronization system for inter-vehicle communication in ad-hoc network. In: International conference on multimedia and ubiquitous engineering, 2007. MUE'07. IEEE; 2007. p. 512–7.
[30] Függer Matthias, Nowak Thomas, Charron-Bost Bernadette. Diffusive clock synchronization in highly dynamic networks. In: 2015 49th annual conference on information sciences and systems (CISS). IEEE; 2015. p. 1–6.
[31] Ganeriwal Saurabh, Kumar Ram, Srivastava Mani B. Timing-sync protocol for sensor networks. In: Proceedings of the 1st international conference on embedded networked sensor systems. ACM; 2003. p. 138–49.
[32] Maróti Miklós, Kusy Branislav, Simon Gyula, Lédeczi Ákos. The flooding time synchronization protocol. In: Proceedings of the 2nd international conference on embedded networked sensor systems. ACM; 2004. p. 39–49.
[33] Khoshdelniat Reza, Sim Moh Lim, Ewe Hong Tat, Wei Tan Su. Time table transfer time syncronization in mobile wireless sensor networks. In: PIERS proceedings. p. 1724–8.
[34] Medani Khedidja, Aliouat Makhlouf, Aliouat Zibouda. High velocity aware clocks synchronization approach in vehicular ad hoc networks. In: IFIP international conference on computer science and its applications_x000D_. Springer; 2015. p. 479–90.
[35] Függer Matthias, Nowak Thomas, Charron-Bost Bernadette. Diffusive clock synchronization in highly dynamic networks. In: 2015 49th annual conference on information sciences and systems (CISS). IEEE; 2015. p. 1–6.
[36] Aissaoua Habib, Aliouat Makhlouf, Bounceur Ahcène, Euler Reinhardt. A distributed consensus-based clock synchronization protocol for wireless sensor networks. Wirel Person Commun 2017:1–22.
[37] Wang Zhigang, Liu Lichuan, Zhou MengChu, Ansari Nirwan. A position-based clustering technique for ad hoc intervehicle communication. IEEE Trans Syst Man Cybern Part C (Appl Rev) 2008;38(2):201–8.
[38] Caballeros Morales Mildred M, Hong Choong Seon, Bang Young-Cheol. An adaptable mobility-aware clustering algorithm in vehicular networks. In: 2011 13th Asia-Pacific network operations and management symposium (APNOMS). IEEE; 2011. p. 1–6.
[39] Almalag Mohammad S, Weigle Michele C. Using traffic flow for cluster formation in vehicular ad-hoc networks. In: 2010 IEEE 35th conference on local computer networks (LCN). IEEE; 2010. p. 631–6.
[40] Li Weiwei, Tizghadam Ali, Leon-Garcia Alberto. Robust clustering for connected vehicles using local network criticality. In: 2012 IEEE international conference on communications (ICC). IEEE; 2012. p. 7157–61.
[41] Wang Sheng-Shih, Lin Yi-Shiun. Passcar: a passive clustering aided routing protocol for vehicular ad hoc networks. Comput Commun 2013;36(2):170–9.
[42] Zhang De-gan, Zhu Ya-nan, Zhao Chen-peng, Dai Wen-Bo. A new constructing approach for a weighted topology of wireless sensor networks based on local-world theory for the internet of things (iot). Comput Math Appl 2012;64 (5):1044–55.
[43] Mezher Ahmad Mohamad, Oltra Juan Jurado, Aguar Luis Urquiza, Paredes Cristhian Iza, Barba Carolina Tripp, Aguilar-Igartua Mónica. Realistic environment for vanet simulations to detect the prsence of obstacles in vehicular ad hoc networks. In: Proceedings of the 11th ACM symposium on performance evaluation of wireless ad hoc, sensor, and ubiquitous networks, PE-WASUN 2014, Monterial, QC, Canada, September 21–26, 2014. ACM; 2014. p. 77–84.
[44] Lombardi MA. Frequency measurement. The measurement, instrumentation and sensors handbook. CRC Press; 1999.