Autonomous TDMA Alignment for VANETs *

Mohamed Mustafa  Marina Papatriantafilou  Elad M. Schiller  Amir Tohidi  Philippas Tsigas
Chalmers University of Technology, Sweden {mohmus, ptrianta, elad, tohidi, tsigas}@chalmers.se

Abstract—The problem of local clock synchronization is studied in the context of media access control (MAC) protocols, such as time division multiple access (TDMA), for dynamic and wireless ad hoc networks. In the context of TDMA, local pulse synchronization mechanisms let neighboring nodes align the timing of their packet transmissions, and by that avoid transmission interferences between consecutive timeslots. Existing implementations for Vehicular Ad-Hoc Networks (VANETs) assume the availability of common (external) sources of time, such as base-stations or geographical positioning systems (GPS). This work is the first to consider autonomic design criteria, which are imperative when no common time sources are available, or preferred not to be used, due to their cost and signal loss. We present self-⋆ pulse synchronization strategies. Their implementing algorithms consider the effects of communication delays and transmission interferences. We demonstrate the algorithms via extensive simulations in different settings including node mobility. We also validate these simulations in the MicaZ platform, whose native clocks are driven by inexpensive crystal oscillators. The results imply that the studied algorithms can facilitate autonomous TDMA protocols for VANETs.

I. INTRODUCTION

Recent work on vehicular systems explores a promising future for vehicular communications. They consider innovative applications that reduce road fatalities, lead to greener transportation, and improve the driving experience, to name a few. The prospects of these applications depend on the existence of predictable communication infrastructure for dynamic networks. We consider time division multiple access (TDMA) protocols that can divide the radio time regularly and fairly in the presence of node mobility, such as Chameleon-MAC [8]. The studied problem appears when neighboring nodes start their broadcasting timeslots at different times. It is imperative to employ autonomous solutions for timeslot alignment when no common (external) time sources are available, or preferred not to be used, due to their cost and signal loss. We address the timeslot alignment problem by considering the more general problem of (decentralized) local pulse synchronization. Since TDMA alignment is required during the period in which communication links are being established, we consider non-deterministic communication delays, the effect of transmission interferences and local clocks with arbitrary initial offsets, see Section II. We propose autonomous and self-⋆ algorithmic solutions that guarantee robustness and provide an important level of abstraction as they liberate the system designer from dealing with low-level problems, such as availability and cost of common time sources, see Section III. Our contribution also facilitates autonomous TDMA protocols for Vehicular Ad-Hoc Networks (VANETs), see Section IV.

Let us illustrate the problem and the challenges of possible strategies using an example. Consider three neighboring stations that have unique timeslot assignment, but their timeslots are not well-aligned, see Fig. 1. Packet transmissions collide in the presence of such concurrent transmissions. Suppose that the stations act upon the intuition that gradual pairwise adjustments are most preferable. Station $p_k$ is the first to align itself with its closest neighbor, $p_j$, see Fig. 2. Next, $p_j$ aligns itself with $p_i$ and by that it opens a gap between itself and $p_k$. Then, $p_k$ aligns itself with $p_i$ and $p_j$. The end result is an all aligned sequence of timeslots. We call this algorithmic approach the cricket strategy.

Observe that the convergence process includes chain reactions, i.e., node $p_k$ aligns itself before and after $p_j$’s alignment. One can foresee the outcome of such chain reactions and let $p_i$ and $p_k$ to concurrently adjust their clock according to $p_j$. This algorithmic approach, named the grasshopper, is faster than the cricket, see Section IV. This improvement comes at the cost of additional memory and processing requirements. We integrate the proposed algorithms with the Chameleon-MAC [8], which is a self-⋆, mobility resilient, TDMA protocol. After extensive simulations with and without mobility, we observe tight alignment among the timeslots, and high MAC throughput. Additional testbed experiments appear in [12].

Biologically-inspired synchronization mechanisms are proposed in [3, 10, 11]. They, and others such as [14], do not consider wireless communication environments with communication delays or disruptions. More practical communication environments are considered in [5, 15, 16], but they do not have TDMA MAC in mind. In [5], Byzantine-tolerance and self-stabilization properties are considered after communication establishment. We are the first to consider TDMA timeslot alignment during the period in which communication links are being established.

II. PRELIMINARIES

The system consists of a set, $N = \{p_i\}$, of $n$ anonymous communicating entities, which we call nodes. The radio time is divided into fixed size TDMA frames and then into fixed size timeslots [as in 8]. The nodes’ task is to adjust their local clocks so that the starting time of frames and timeslots is aligned. They are to achieve this task in the presence of: (1)
a MAC layer that is in the process of assigning timeslots, (2) network topology changes, and (3) message omission, say, due to topological changes, transmission interferences, unexpected change of the ambient noise level, etc.

**Time, clocks, and synchrony bounds** We consider three notations of time: *real time* is the usual physical notion of continuous time, used for definition and analysis only; *native time* is obtained from a native clock, implemented by the operating system from hardware counters; *local time* builds on native time with an additive adjustment factor in an effort to facilitate a neighborhood-wise clock. Applications require the clock interface to include the **READ** operation, which returns a **timestamp** value of the local clock. Let \( C_i^k \) and \( c_i^k \) denote the value \( p_i \in N \) gets from the \( k^{th} \) **READ** of the native or local clock, respectively. Moreover, let \( r_i^k \) denote the real-time instance associated with that \( k^{th} \) **READ** operation. Pulse synchronization algorithms adjust their local clocks in order to achieve synchronization, but never adjust their native clocks. Namely, the operation **ADJUST**(add) adds a positive integer value to the local clock. This work considers solutions that adjust clocks forward, because such solutions simplify the reasoning about time at the higher layers. We define the native clocks offset \( \delta_{i,j}(k,q) = C_i^k - C_j^q \), and the local clocks offset \( \Delta_{i,j}(k,q) = c_i^k - c_j^q \); where \( \Delta_{i,j}(k,q) = r_i^k - r_j^q = 0 \). Given a real-time instance \( t \), we define the (local) synchrony bound \( \psi(t) = \max(\{\Delta_{i,j}(k,q); p_i, p_j \in N \land \Delta_{i,j}(k,q) = 0\}) \) as the maximal clock offset among the system nodes.

One may consider \( p_i \)'s (clock) skew, \( \rho_i = \lim_{\Delta_{i,j}(k,q) \to 0} \delta_{i,j}(k,q)/\Delta_{i,j}(k,q) \in [\rho_{\min}, \rho_{\max}] \), where \( \rho_{\min} \) and \( \rho_{\max} \) are known constants [4, 6]. The clock skew of MicaZ nodes is bounded by a constant that is significantly smaller than the communication delays. Therefore, our simulations assume a zero skew. We validate these simulations in the MicaZ platform.

**Pulses** Each node has hardware supported timer for generating (periodic) pulses every \( P \) (phase) time units. Denote by \( c_{iq} \) the \( k-th \) time in which node \( p_i \)'s timer triggers a pulse, immediately after performing the READ operation for the \( q_k - th \) time. The term timeslot refers to the period between two consecutive pulses at times \( c_{iq} \) and \( c_{iq+1} \). We say that \( t_i = c_{iq} \mod P \) is \( p_i \)'s (pulse) phase value. Namely, whenever \( t_i = 0 \), node \( p_i \) raises the event **timeslot**(\( t_i \)), where \( s_i = k \mod T \) is \( p_i \)'s (broadcasting) timeslot number and \( T > 1 \) is the **TDMA frame size**.

![Fig. 2. The cricket strategy. Solid and dashed lines stand for transmission, and respectively, idle radio times. The circles above the solid boxes represents the node's view on its neighbors' TDMA alignment at the start of its broadcasting timeslot. Gaps between two solid boxes represent alignment events.](image)

The MAC layer The studied algorithms use packet transmission schemes that employ communication operations for receiving, transmitting and carrier sensing. Our implementation considers merely the latter two operations, as in the Beeps model [2], which also considers the period prior to communication establishment. We denote the operations' time notation (timestamp) in the format \((\text{timeslot}, \text{phase})\), where \( \text{timeslot} \in [0, T - 1] \) and \( \text{phase} \in [0, P - 1] \). We assume the existence of efficient mechanisms for timestamping packets at the MAC layer that are executed by the transmission operations, as in [4]. We assume the existence of an efficient upper-bound, \( \alpha \ll P \), on the communication delay between two neighbors, that, in this work, has no characterized and known distribution.

**Task definition** The problem of (decentralized) local pulse synchronization considers the rapid reduction of all local synchrony bounds \( \psi \geq \max(\{\Lambda_{i,j}(k,q); p_i, p_j \in N \land \Delta_{i,j}(k,q) = 0\}) \), where \( N_i^P \) refers to \( p_i \)'s recent neighbors, see Fig. 3 for definition. Given the synchrony bound \( \psi \geq 0 \), we look at the convergence (rate bound), \( t_\psi \), which is the number of TDMA frames it takes to reach \( \psi \). Recall that we consider only forward clock adjustments. We also study local pulse synchronization’s relation to MAC-layer, network scalability and topological changes.

**III. PULSE SYNCHRONIZATION STRATEGIES**

Pulse synchronization solutions require many considerations, e.g., non-deterministic delays and transmission interferences. Before addressing the implementation details, we simplify the presentation by first presenting (algorithmic) strategies in which the nodes learn about their neighbors’ clock values without delays and interferences.

We present two strategies that align the TDMA timeslots by calling the function **ADJUST**(aim) immediately before their broadcasting timeslot, see Fig. 2. The first strategy, named Cricket, sets **aim**’s value according to neighbors that have the most similar phase values. The second strategy, named Grasshopper, looks into a greater set of neighbors before deciding on **aim**’s value. Both strategies are based on the relations among nodes’ phase values, see Fig. 3 for definitions.

**Cricket strategy** This strategy acts upon the intuition that gradual pairwise adjustments are most preferable. Node \( p_i \) raises the event **timeslot**(\( t_i \)), when \( t_i = 0 \), and adjusts its local clock according to Eq. (1). At this time, **PhaseOrder**\(_{p_i}\)'s first item has zero value, because it refers to \( p_i \)'s own pulse, the second item refers to \( p_i \)'s successor and the last item refers to \( p_i \)'s predecessor.

\[
\text{aim}_{p_i} = \begin{cases} \text{head}_{p_i} & : \text{head}_{p_i} < \text{tail}_{p_i} \quad \text{JUMP} \\ 0 & : \text{head}_{p_i} > \text{tail}_{p_i} \quad \text{WAIT} \\ \text{head}_{p_i} = \text{tail}_{p_i} & : \text{head}_{p_i} = \text{tail}_{p_i} \quad \text{MIX} \\ \end{cases}
\]

The cricket strategy considers both pure deterministic actions (JUMP and WAIT) and a non-deterministic one (MIX).

- **JUMP:** Whenever node \( p_i \) is closer to its predecessor than to its successor (\( \text{head}_{p_i} < \text{tail}_{p_i} \)), it catches up with its predecessor by adding \( \text{head}_{p_i} \) to its clock value, which is the phase difference between itself and its predecessor.

- **WAIT:** Whenever \( p_i \) is closer to its successor than to its predecessor (\( \text{head}_{p_i} > \text{tail}_{p_i} \)), \( p_i \) simply waits for its successor.

- **MIX:** Node \( p_i \) breaks symmetry whenever it is as close to its predecessor as it is to its successor (\( \text{head}_{p_i} = \text{tail}_{p_i} \)).
At any real-time instance $t$, $p_i$’s reach set $R_i(t) = \{p_j\} \subseteq N$, represents the set of nodes, $p_j$, that receive $p_i$’s transmissions. At the MAC layer, the real-time instance $t$ refers to the time in which $p_j$ raises the carrier sense event. The set recent neighbors, $N^t = \{p_j \in N : \text{starting-time}(s_j) \leq t \}$, refers to nodes whose broadcast in timeslot $s_j$, arrive to node $p_i$, where $t$ is a real-time instance that happens $T$ timeslots before the real-time instance $t'$ and starting-time$(s_j) \in [t, t')$ refers to the starting time of $p_i$’s timeslot.

Locally observed pulse profiles Given a real-time instance $t$ and node $p_i \in N$, we denote the locally observed pulse profile by $\gamma_i(t) = (\{s_j, t_j\})_{p_j \in N^t}$, as a list of $p_i$’s recently observed timestamps during the passed $T$ timeslots before $t$. We sometimes write $\gamma_i$, rather than $\gamma_i(t)$, when $t$ refers to the starting time of $p_i$’s timeslot.

Phase orders Let $Order = (p_k)_{k=0}^{N-1}$ be an ordered list of nodes in $N$, where $p_i$’s predecessor and successor in $N$ are $p_{i+k+1} \mod_n$ and respectively, $p_{i+k} \mod_n$. The ordered list, $\text{PhaseOrder}_i$, of the pulse profile, $\gamma_i$, is sorted by the phase field of $\gamma_i$’s timestamp (timeslot,$j$,phase,$j$) $\in \gamma_i$. Predecessors, successors, heads, and tails Given a node, $p_i$, and its view on the pulse profile, $\gamma_i$, define the predecessor, and the successor, as $p_i$’s predecessor, and respectively, successor in $\text{PhaseOrder}_i$. Moreover, $\text{head}_i = (t_i - t_{pr}) \mod P$ and $\text{tail}_i = (t_{su} - t_i) \mod P$ is the phase difference between $p_i$’s phase value, $t_i$ and predecessor $\text{head}_i = p_{pr}$, and respectively, successor $\text{tail}_i = p_{su}$. These imply that $\text{predecessor}_i$ is pulsed head$_i$ time units before node $p_i$ and successor$_i$ is pulsed tail$_i$ time units after node $p_i$.

**Set up:** Given that the broadcasting schedule of nodes $N = \{p_1, \ldots, p_4\}$ is by their index value, the pulse profiles $\gamma_i(t), p_i \in N$, encode pulses such that initially $p_1$ and $p_2$ have local dominant pulses. **Convergence:** In the first TDMA frame, nodes $N(1) = \{p_3, p_4\}$ converge towards their respective local dominant pulses in $p_3 \in \{p_1, p_3\}$. In the second TDMA frame, the local dominant pulses are nodes $p_2 \in \{p_1, p_2\}$. Note that node $p_2$’s pulse is no longer a local dominant and it adjusts its phase according to Eq. (1), i.e., $N(2) = \{p_3, p_4\}$.

**Local dominant pulses** Let us look into a typical convergence of the cricket strategy, see Fig. 4. Given two nodes, $p_i, p_j \in N$, and $p_i$’s locally observed pulse profile, $\gamma_i(t)$, we say that $p_i$’s pulse (phase value) locally dominates the one of $p_j$, if $\text{head}_j < \text{tail}_j$. and $p_j$ is $p_i$’s predecessor in $\gamma_i$. Observe that clock updates can result in a chain reaction, see Fig. 4. Lengthy chain reactions can prolong the convergence up to $O(n)$ TDMA frames, see Fig. 5.

**Global dominant pulses** In Fig. 5, all nodes eventually align their timeslots with the one of $p_1$, because $p_1$’s pulse immediately follows the maximal gap in $\gamma_i$. Pulse gaps provide useful insights into the cricket strategy convergence. Given node $p_i \in N$, its pulse profile $\gamma_i$ and $k \in [1, |N|^T]$, we obtain the (pulse) gaps between $\gamma_i$’s consecutive pulses, $\gamma_{i,j}(k) = (\text{PhaseOrder}_i[k], \gamma) - \text{PhaseOrder}_i[k-1], \gamma$), see Fig. 3 for definitions. For the case of $k = 0$, we define $\gamma_{i,j}(0) = (P - \text{PhaseOrder}_i[|N|^T], \gamma)$. The set, $\gamma_{i,j}$, of pulses that immediately follow the maximal gap in $\gamma_i$ are named global dominants.

\[
\gamma_{i,j} = \text{argmax}_{k \in [1, |N|^T]} (\gamma_{i,j}(k))
\]

This case, $p_i$ randomly chooses between JUMP and WAIT.

**Grasshopper strategy** This strategy is based on the ability to see beyond the immediate predecessor and local dominant pulses. The nodes converge by adjusting their local clocks according to the phase value of their global dominant pulses, and by that avoid lengthy chain reactions of clock updates. Eq. (3) defines the adjustment value, $\gamma_{i,j}$, for the grasshopper strategy. Whenever node $p_i \in N$ notices that its clock phase value is dominated by the one of node $p_j \in N$, node $p_i$ aligns its clock phase value with the one of $p_j$, see the JUMP step. This, whenever a single global dominant pulse exists, the convergence speed-up is made simple, because all nodes adjust their clock values according to the dominant pulse of $p_j$. Thus, there are no chain reactions of clock updates. Note that node $p_j$ does not adjust its clock, see the WAIT.

Fig. 3. Pulse profiles and the relations among nodes’ phase values

Fig. 4. Cricket strategy convergence during the first two TDMA frames.

Fig. 5. Chain reactions of pulse updates: An example with $O(n)$ TDMA frames before convergence.
Both algorithms reduce the synchrony bound to 1% of the timeslot size, see Fig. 7. Moreover, the synchrony bounds of the cricket and grasshopper are 24% and, respectively, 62% lower when using preassigned TDMA rather than Chameleon-MAC [8]. However, these values drop to 0.04%, and respectively, 0.4% after convergence. Furthermore, the grasshopper convergence is 5.4 times faster than of the cricket. In addition, the cricket and grasshopper converge 6.8%, and respectively, 40% times faster when using preassigned TDMA rather than Chameleon-MAC, see the cricket’s lengthy chain reactions explained in Section III.

We also study the algorithms’ scalability by considering a variable number of nodes, n ∈ {10, 20, 30, ..., 100}. The grasshopper converges faster than the cricket as the number of nodes increases, cf. Fig. 8 (a) and (b). The convergence depends on the number of nodes. E.g., for 10% synchrony bound, 0.3n + 6.4 and 0.0062n + 10.86 are linear interpolations of the convergence time for the cricket, and respectively, grasshopper strategies. Moreover, 2(\log_2(n) + 0.1) is a logarithmic interpolation of the grasshopper convergence time. This suggests that the grasshopper has lower dependency on the network size than the cricket.

During the grasshopper executions, we often observed a single global dominant pulse that facilitates rapid TDMA alignment, rather than a lengthy chain reactions, see Section III. The proposed algorithms affect the MAC throughput, which is the radio time utilization percentage, cf. Fig. 8 (c) and (d). Both algorithms eventually reach a throughput of 70%.

### IV. EXPERIMENTAL EVALUATION

Computer simulations and the MicaZ platform are used for showing that: (1) both proposed algorithms achieve a small synchrony bound, and (2) the grasshopper, which has a higher resource consumption cost, converges faster than the cricket.

**Experiments design** The proposed algorithms aim at aligning the TDMA timeslots during the MAC’s timeslot assignment period. Since communication interruptions can occur, the nodes might not correctly observe their local pulse profiles. Therefore, we compare the result parameters, synchrony bound, \( \psi \), and convergence time, \( \ell_\psi \), using both: (1) a MAC protocol that uses preassigned timeslots, and (2) Chameleon-MAC, a self-* TDMA protocol [8]. Moreover, the platform validation considers a control experiment using a centralized pulse synchronizer. This external time source is provided by a (base-station) node that periodically broadcasts. The centralized pulse synchronizer serves as a baseline for estimating the overheads imposed by the autonomous design.

The analysis considers the average over 8 experiments in which the simulations consider a timeslot size of, \( P = 5 \text{ msec} \), and a communication delay bound of, \( \alpha = 5\% \) of \( P \).

**Simulation experiments** The proposed pulse synchronization algorithms are simulated using TOSSIM [9] on single-hop, multi-hop and mobile ad hoc networks. We observe the synchrony bound and convergence time, and study the proposed algorithms’ relation to MAC-layer, network scalability and topological changes.

**Single-hop Ad Hoc Network** Both algorithms reduce the synchrony bound down to 1% of the timeslot size, see Fig. 7. Moreover, the synchrony bounds of the cricket and grasshopper are 24%, and respectively, 62% lower when using preassigned TDMA rather than Chameleon-MAC [8]. However, these values drop to 0.04%, and respectively, 0.4% after convergence. Furthermore, the grasshopper convergence is 5.4 times faster than of the cricket. In addition, the cricket and grasshopper converge 6.8%, and respectively, 40% times faster when using preassigned TDMA rather than Chameleon-MAC, see the cricket’s lengthy chain reactions explained in Section III.

We also study the algorithms’ scalability by considering a variable number of nodes, n ∈ {10, 20, 30, ..., 100}. The grasshopper converges faster than the cricket as the number of nodes increases, cf. Fig. 8 (a) and (b). The convergence depends on the number of nodes. E.g., for 10% synchrony bound, 0.3n + 6.4 and 0.0062n + 10.86 are linear interpolations of the convergence time for the cricket, and respectively, grasshopper strategies. Moreover, 2(\log_2(n) + 0.1) is a logarithmic interpolation of the grasshopper convergence time. This suggests that the grasshopper has lower dependency on the network size than the cricket.

During the grasshopper executions, we often observed a single global dominant pulse that facilitates rapid TDMA alignment, rather than a lengthy chain reactions, see Section III. The proposed algorithms affect the MAC throughput, which is the radio time utilization percentage, cf. Fig. 8 (c) and (d). Both algorithms eventually reach a throughput of 70%.

**Multi-hop Ad Hoc Network** Fig. 9-(left) considers networks with 45 nodes, and diameters of 6 hops. Often, synchrony bounds depend on the network diameter [7]. The observed synchrony bound increased to 3% of the timeslot...
and a greater resiliency degree. That the grasshopper was able to show a shorter recovery time. We observed misalignment and an increase in synchrony bound when the synchronized phase value. This difference results in timeslot other and thus they experience transient radio interferences, see Fig. 10, where the transmission 10% (interference) radius was diameter of 12 2. Interferences follow regular patterns when the nodes are placed from [8] for studying the algorithms. One in which radio size and the grasshopper converged 3.25 times faster than the cricket.

Virtual Ad Hoc Network We borrow two mobility models from [8] for studying the algorithms. One in which radio interferences follow regular patterns when the nodes are placed in parallel lanes and move in opposite directions (72 node and diameter of 12). Both algorithms have quickly reached to a 10% synchrony bound, see Fig. 10, where the transmission (interference) radius was 22 distance units. The second model considers 2 clusters of 50 nodes each that pass by each other and thus they experience transient radio interferences, see Fig. 9(right). Initially, the two clusters differ in their synchronized phase value. This difference results in timeslot misalignment and an increase in synchrony bound when the clusters are within each others interference range. We observed that the grasshopper was able to show a shorter recovery time and a greater resiliency degree.

V. DISCUSSIONS

The prospects of safety-critical vehicular systems depend on the existence of predictable communication protocols that divide the radio time regularly and fairly. This paper presents autonomous and self-* algorithmic solutions for the problem of TDMA timeslot alignment by considering the more general problem of (decentralized) local pulse synchronization. The studied algorithms facilitate autonomous TDMA-based MAC protocols that are robust to transient faults, have high throughput and offer a greater predictability degree with respect to the transmission schedule. These properties are often absent from current MAC protocol implementations for VANETs, see [1, 13]. We saw that avoiding clock update dependencies can significantly speed up the convergence and recovery processes. In particular, the grasshopper algorithm foresees dependencies among the clock updates, which the cricket cannot. However, dependency avoidance requires additional resources.

Existing vehicular systems often assume the availability of common time sources, e.g., GPS. Autonomous systems cannot depend on GPS services, because they are not always available, or preferred not to be used, due to their cost. Arbitrarily long failure of signal loss can occur in underground parking lots and road tunnels. Moreover, some vehicular applications cannot afford accurate clock oscillators that would allow them to maintain the required precision during these failure periods.

By demonstrating the studied algorithms on inexpensive MicaZ motes, we have opened up the door for hybrid-autonomous designs (cf. centralized pulse synchronizer in Section IV). Namely, nodes that have access to GPS, use this time source for aligning their TDMA timeslots, whereas nodes that have no access to GPS, use the studied strategies as dependable fallback for catching up with nodes that have access to GPS.

We expect applicability of the hybrid-autonomous design criteria to other areas of VANETs. E.g., spatial TDMA [13] protocols base their timeslot allocation on GPS availability. As future work, we propose dealing with such dependencies by adopting the hybrid-autonomous design criteria.

REFERENCES