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-$\star$ 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-$\star$ 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.

* This work was partially supported by the EC, through project FP7-STREP-288195, KARYON (Kernel-based ARchitecture for safetY-critical cONtrol).

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_j and p_k to concurrently adjust their clock according to p_i. 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-$\star$, 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].

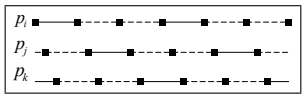

Fig. 1. Unaligned TDMA timeslots. Solid and dashed lines stand for transmission, and respectively, idle radio times.

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)

978-1-4673-1881-5/12/$31.00 ©2012 IEEE

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_k^i and c_k^i denote the value $p_i \in N$ gets from the k^{th} READ of the native or local clock, respectively. Moreover, let r_k^i 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* $\delta_{i,j}(k,q) = C_k^i - C_q^j$, and the local clocks offset $\Lambda_{i,j}(k,q) = c_k^i - c_q^j$; where $\Delta_{i,j}(k,q) = r_k^i - r_q^j = 0$. Given a real-time instance t, we define the *(local clock) synchrony bound* $\psi(t) = \max(\{\Lambda_{i,j}(k,q) : p_i, p_j \in N \wedge \Delta_{i,j}(k,q) = 0\})$ as the maximal clock offset among the system nodes.

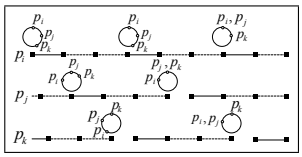

Fig. 2. The cricket strategy. Solid and dashed lines stand for transmission, and respectively, idle radio times. The circles above the solid boxes represents the node's view on its neighbors' TDMA alignment at the start of its broadcasting timeslot. Gaps between two solid boxes represent alignment events.

One may consider p_i's *(clock) skew*, $\rho_i = \lim_{\Delta_{i,i}(k,q)\to 0} \delta_{i,i}(k,q)/\Delta_{i,i}(k,q) \in [\rho_{\min}, \rho_{\max}]$, 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 $c_{q_k}^i$ 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 $c_{q_k}^i$ and $c_{q_{k+1}}^i$. We say that $t_i = c_{q_k}^i \bmod P$ is p_i's *(pulse) phase* value. Namely, whenever $t_i = 0$, node p_i raises the event $timeslot(s_i)$, where $s_i = k \bmod T$ is p_i's (broadcasting) timeslot number and $T > 1$ is the *TDMA frame size*.

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 $\langle timeslot, phase\rangle$, where $timeslot \in [0, T-1]$ and $phase \in [0, P-1]$. 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* $\psi \geq \max(\{\Lambda_{i,j}(k,q) : p_i, p_j \in N \wedge p_j \in \mathcal{N}_i^T \wedge \Delta_{i,j}(k,q) = 0\})$, where $\mathcal{N}_i^T$ 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)*, ℓ_ψ, which is the number of TDMA frames it takes to reach ψ. 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(s_i), when $t_i = 0$, and adjusts its local clock according to Eq. (1). At this time, $PhaseOrder_{\gamma_i}$'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.

$$aim_{\gamma_i} = \begin{cases} head_{\gamma_i} & : head_{\gamma_i} < tail_{\gamma_i} \quad \text{JUMP} \\ 0 & : head_{\gamma_i} > tail_{\gamma_i} \quad \text{WAIT} \\ head_{\gamma_i} \text{or } 0; \text{ each with probability } \frac{1}{2} & : head_{\gamma_i} = tail_{\gamma_i} \quad \text{MIX} \end{cases} \tag{1}$$

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 ($head_{\gamma_i} < tail_{\gamma_i}$), it catches up with its predecessor by adding $head_{\gamma_i}$ 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 ($head_{\gamma_i} > tail_{\gamma_i}$), 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 ($head_{\gamma_i} = tail_{\gamma_i}$). In

Learning about neighbors' clock values 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*, $\mathcal{N}_i^T = \{p_j \in N : \text{starting-time}(s_j) \in [t, t'] \wedge p_i \in R_j(t(s_j))\}$, 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_j'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 $\gamma_i(t) = (\langle s_j, t_j \rangle)_{p_j \in \mathcal{N}_i^T}$, as a list of p_i's recently observed timestamps during the passed T timeslots before t. We sometimes write γ_i, rather than $\gamma_i(t)$, when t refers to the starting time of p_i's timeslot.

Phase orders Let $Order = (p_{i_k})_{k=0}^{n-1}$ be an ordered list of nodes in N, where p_i's predecessor and successor in N are $p_{i_{k-1 \bmod n}}$, and respectively, $p_{i_{k+1 \bmod n}}$. The ordered list, $PhaseOrder_{\gamma_i}$, of the pulse profile, γ_i, is sorted by the phase field of γ_i's timestamp $\langle timeslot_j, phase_j \rangle \in \gamma_i$.

Predecessors, successors, heads, and tails Given a node, p_i, and its view on the pulse profile, γ_i, define the $predecessor_i$ and the $successor_i$ as p_i's predecessor, and respectively, successor in $PhaseOrder_{\gamma_i}$. Moreover, $head_{\gamma_i} = (t_i - t_{pr}) \bmod P$ and $tail_{\gamma_i} = (t_{su} - t_i) \bmod P$ is the phase difference between p_i's phase value, t_i and $predecessor_i = p_{pr}$, and respectively, $successor_i = p_{su}$. These imply that $predecessor_i$ is pulsed $head_{\gamma_i}$ time units before node p_i and $successor_i$ is pulsed $tail_{\gamma_i}$ time units after p_i.

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

Set up: Given that the broadcasting schedule of nodes $N = \{p_1, \ldots, p_4\}$ is by their index value, the pulse profiles, $\{\gamma_i\}_{p_i \in N}$, encode pulses such that initially p_1 and p_3 have local dominant pulses.
Convergence: In the first TDMA frame, nodes $N(1) = \{p_2, p_4\}$ converge towards their respective local dominant pulses in $p_y \in \{p_1, p_3\}$. In the second TDMA frame, the local dominant pulses are nodes $p_y \in \{p_1, p_2\}$. Note that node p_3's pulse is no longer a local dominant and it adjusts its phase according to Eq. (1), i.e., $N(2) = \{p_3, p_4\}$

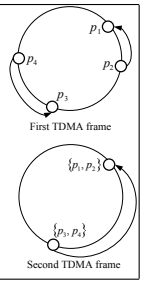

Fig. 4. Cricket strategy convergence during the first two TDMA frames.

this case, p_i randomly chooses between JUMP and WAIT.

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_j's pulse (phase value) *locally dominates* the one of p_i, if $head_{\gamma_i} < tail_{\gamma_i}$ and p_j is p_i's predecessor in γ_i. Observe that clock updates can result in a chain reaction, see Fig. 4. Lengthy chain reactions can prolong the convergence up to $\mathcal{O}(n)$ 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 γ_i. Pulse gaps provide

Set up: For $\xi > 0$ and $p_i \in N$, the pulse profiles $\{\gamma_i\}_{p_i \in N}$ encode pulses in which node p_{i+1}'s pulse occurs $(n-i)\xi$ clock units after p_i's pulse. **Convergence:** In the first TDMA frame, only node p_n can take JUMP action to align with its neighbor local clock, p_{n-1}, because its $head_{\gamma_n} < tail_{\gamma_n}$. In the second TDMA frame, nodes $N(2) = \{p_n, p_{n-1}\}$ adjust their clocks to be aligned with p_{n-2}. Thus, in the $(n-1)$-th TDMA frame, nodes $N(n-1) = \{p_n, p_{n-1}, \ldots, p_2\}$ align with node p_1. Therefore, $n-1$ TDMA frames are needed before convergence.

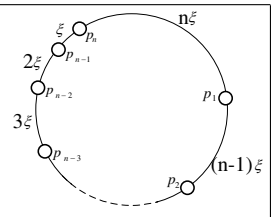

Fig. 5. Chain reactions of pulse updates: An example with $\mathcal{O}(n)$ TDMA frames before convergence.

useful insights into the cricket strategy convergence. Given node $p_i \in N$, its pulse profile γ_i and $k \in [1, |\mathcal{N}_i^T|]$, we obtain the *(pulse) gaps* between γ_i's consecutive pulses, $Gap_{\gamma_i}(k) = (PhaseOrder_{\gamma_i}[k].phase - PhaseOrder_{\gamma_i}[k-1].phase)$, see Fig. 3 for definitions. For the case of $k = 0$, we define $Gap_{\gamma_i}(0) = (P - PhaseOrder_{\gamma_i}[|\mathcal{N}_i^T|].phase)$. The set, $MaxGap_{\gamma_i}$, of pulses that immediately follow the maximal gap in γ_i are named *global dominates*.

$$MaxGap_{\gamma_i} = \underset{k \in [0, |\mathcal{N}_i^T|]}{\operatorname{argmax}} (Gap_{\gamma_i}(k)) \tag{2}$$

Given three nodes, $p_i, p_j, p_\ell \in N$, p_i's locally observed pulse profile, $\gamma_i(t)$, $j \in MaxGap_{\gamma_i}$ and $i, \ell \notin MaxGap_{\gamma_i}$, we say that p_j's pulse *globally dominates the one of* p_i, if at least one of the following holds: (1) $i = j$ (2) p_j's pulse locally dominates the one of p_i, or (3) p_j's pulse globally dominates the one of p_ℓ and p_ℓ's pulse locally dominates the one of p_i. We define p_i's clock offset towards its preceding global dominant pulse as $DomPulse_i = P - PhaseOrder_{\gamma_i}[k].phase$, where $k \in MaxGap_{\gamma_i}$ is p_i's global dominant pulse, see Eq. (2).

We define $OneGlobal(\gamma_i) = (|\ MaxGap_{\gamma_i}\ | = 1)$ to be true whenever γ_i encodes a single global dominant pulse. For the cases in which there is more than one, we define the term next (global) dominant pulse for node p_i, where $i \in MaxGap_{\gamma_i}$ refer to p_i's global dominant pulse. In this case, p_i's *next (global) dominant pulse*, $NextDomPulse_i = DomPulse_{pr}$, is p_i's predecessor's global dominant pulse, where $predecessor_i = p_{pr}$.

Next we present the grasshopper strategy, which uses the notion of global dominant pulses to avoid lengthy chain reactions in order to achieve a faster convergence.

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, $aim(\gamma_i)$, 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. Thus, 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

Set up: Given nodes $N = \{p_1, \ldots, p_9\}$, the pulse profiles, $\{\gamma_i\}_{p_i \in N}$, encode pulses, such that initially p_1 and p_6 are global dominants.
Convergence: During the first TDMA frame, nodes $p_x \in \{p_2, \ldots, p_5\}$ and $p_y \in \{p_7, p_8, p_9\}$ take the JUMP action to align their local clocks with their respective preceding global dominant pulse, p_1 and p_6 (solid lines). Whereas, node p_1 and p_6 take the MIX action to either stay or align to next dominant pulse (dotted lines).

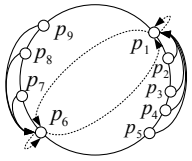

Fig. 6. Typical convergence process of the grasshopper strategy.

step. For the possible case of many global dominant pulses, we take the mixed strategy approach, see the MIX step. Here, chain reactions of clock updates can occur (Fig. 6). They occur only among the nodes whose clock phase values are global dominants.

$$aim_{\gamma_i} = \begin{cases} DomPulse_i & : DomPulse_i < P \quad \text{JUMP} \\ 0 & : Dom_i = P \\ & \wedge\ OneGlobal(\gamma_i) \quad \text{WAIT} \\ NextDomPulse_i \text{ or } 0; & : \textbf{else} \quad \text{MIX} \\ \text{each with probability } \frac{1}{2} & \end{cases} \tag{3}$$

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, ψ, and convergence time, ℓ_ψ, using both: (1) a MAC protocol that uses preassigned timeslots, and (2) Chameleon-MAC, a self-$\star$ 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, $P = 5\ msec$, 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

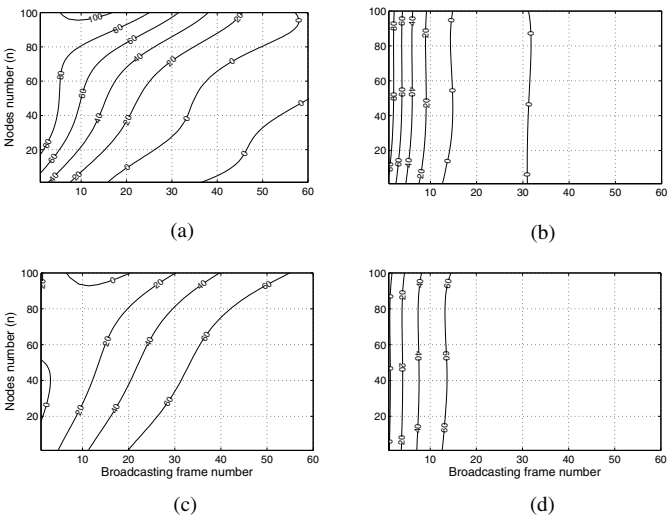

Fig. 8. Synchrony bounds (as timeslot percentage) and throughput levels (as radio time utilized percentage) for single-hop networks of $n \in \{10, 20, 30, \cdots, 100\}$ nodes. Top and bottom contour plots show the synchrony bounds, and respectively, throughput levels for cricket (a) and (c), and grasshopper (b) and (d). Given these plots, the number of TDMA frames needed to reach to a particular synchrony bound (or throughput) by a given number of nodes can be estimated. E.g., 60 nodes reach 20% synchrony within 25 frames using the cricket strategy.

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, $n \in \{10, 20, 30, \ldots, 100\}$. 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%.

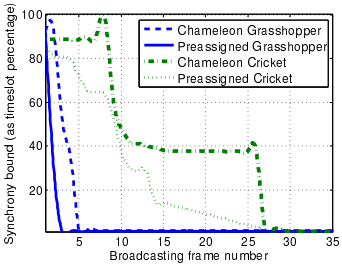

Fig. 7. 20 node single-hop network with two kind of MACs.

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

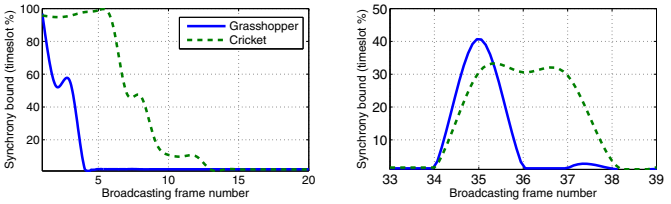

Fig. 9. (Left) Multi-hop network synchrony bound and convergence time the algorithms using Chameleon-MAC. (Right) Cricket and grasshopper convergence with mobile clusters.

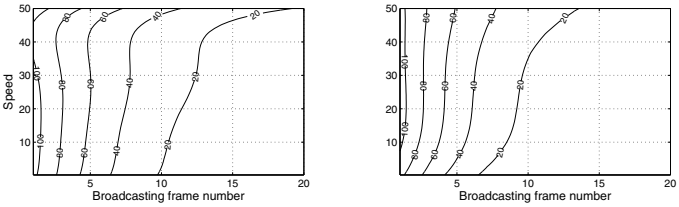

Fig. 10. The synchrony bound (as timeslot percentage) for the cricket (left) and grasshopper (right) using Chameleon-MAC and considering regular interferences. The neighborhood change rate increases with speed, causing the algorithms to spend longer time for convergence.

size and the grasshopper converged 3.25 times faster than the cricket.

Mobile 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-$\star$ 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 implantations 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

[1] K. Bilstrup, E. Uhlemann, E. G. Ström, and U. Bilstrup. "Evaluation of the IEEE 802.11p MAC method for vehicle-to-vehicle communication," *IEEE VTC Fall*, pp. 1–5, 2008.

[2] A. Cornejo and F. Kuhn. "Deploying wireless networks with beeps," *Distributed Systems and Networks*, pp. 148–162, 2010.

[3] A. Daliot, D. Dolev, and H. Parnas. "Self-stabilizing pulse synchronization inspired by biological pacemaker networks," *Stabilization, Safety, and Security of Distributed Systems*, pp. 32–48, 2003.

[4] T. Herman and C. Zhang. "Best paper: Stabilizing clock synchronization for wireless sensor networks," *Stabilization, Safety, and Security of Distributed Systems*, pp. 335–349, 2006.

[5] E. N. Hoch. "Self-stabilizing byzantine pulse and clock synchronization," Master's thesis, CSE Hebrew Univ. of Jerusalem, 2007.

[6] J.-H. Hoepman, A. Larsson, E. M. Schiller, and P. Tsigas. "Secure and self-stabilizing clock synchronization in sensor networks," *Stabilization, Safety, and Security of Distributed Systems*, v. 4838 of *LNCS*, pp. 340–356. Springer, 2007.

[7] C. Lenzen, T. Locher, and R. Wattenhofer. "Tight bounds for clock synchronization," *J. ACM*, 57(2), 2010.

[8] P. Leone, M. Papatriantafilou, E. M. Schiller, and G. Zhu. "Chameleon-mac: Adaptive and self-* algorithms for media access control in mobile ad hoc networks," *Stabilization, Safety, and Security of Distributed Systems*, pp. 468–488, 2010.

[9] P. Levis, N. Lee, M. Welsh, and D. E. Culler. "TOSSIM: accurate and scalable simulation of entire tinyos applications," *ACM SenSys*, pp. 126–137, 2003.

[10] D. Lucarelli and I.-J. Wang. "Decentralized synchronization protocols with nearest neighbor communication," *ACM SenSys*, pp. 62–68, 2004.

[11] R. E. Mirollo, Steven, and H. Strogatz. "Synchronization of pulse-coupled biological oscillators," *SIAM*, 50:1645–1662, 1990.

[12] M. H. Mustafa. "Self-$\star$ Pulse Synchronization for Autonomous TDMA MAC in VANETs," *CSE, Chalmers Univ. of Tech.*, 2012.

[13] K. Sjöberg, E. Uhlemann, and E. G. Ström. "Delay and interference comparison of CSMA and self-organizing TDMA when used in VANETs," *Wireless Comm. & Mobile Comp.*, pp. 1488–1493, 2011.

[14] R. Solis, V.S. Borkar, and P.R. Kumar. "A New Distributed Time Synchronization Protocol for Multihop Wireless Networks," IEEE Decision and Control, pp. 2734–2739. 2006.

[15] A. Tyrrell, G. Auer, and C. Bettstetter. "Fireflies as role models for synchronization in ad hoc networks," *ACM ICST BIONETICS*. 2006.

[16] G. Werner-Allen, G. Tewari, A. Patel, M. Welsh, and R. Nagpal. "Firefly-inspired sensor network synchronicity with realistic radio effects," *ACM SenSys*, pp. 142–153. 2005.