Self-stabilizing TDMA Algorithms for Wireless Ad-hoc Networks without External Reference

(Extended Abstract)

Thomas Petig, Elad M. Schiller, Philippas Tsigas
{petig, elad, tsigas}@chalmers.se
Department of Computer Science and Engineering
Chalmers University of Technology, Sweden

***Abstract*—Time division multiple access (TDMA) is a method for sharing communication media. In wireless communications, TDMA algorithms often divide the radio time into timeslots of uniform size, ξ, and then combine them into frames of uniform size, τ. We consider TDMA algorithms that allocate at least one timeslot in every frame to every node. Given a maximal node degree, δ, and no access to external references for collision detection, time or position, we consider the problem of collision-free self-stabilizing TDMA algorithms that use constant frame size.**

We demonstrate that this problem has no solution when the frame size is $\tau < \max\{2\delta, \chi_2\}$, where χ_2 is the chromatic number for distance-2 vertex coloring. As a complement to this lower bound, we focus on proving the existence of collision-free self-stabilizing TDMA algorithms that use constant frame size of τ. We consider basic settings (no hardware support for collision detection and no prior clock synchronization), and the collision of concurrent transmissions from transmitters that are at most two hops apart. In the context of self-stabilizing systems that have no external reference, we are the first to study this problem (to the best of our knowledge), and use simulations to show convergence even with computation time uncertainties.

I. Introduction

Autonomous and cooperative systems will ultimately carry out risk-related tasks, such as piloting driverless cars, and liberate mankind from mundane labor, such as factory and production work. Note that the implementation of these cooperative systems implies the use of wireless ad hoc networks and their critical component – the *medium access control* (MAC) layer. Since cooperative systems operate in the presence of people, their safety requirements include the provision of real-time guarantees, such as constant communication delay. Infrastructure-based wireless networks successfully provide high bandwidth utilization and constant communication delay. They divide the radio into *timeslots* of uniform size, ξ, that are then combined into *frames* of uniform size, τ. Base-stations, access points or wireless network coordinators can schedule the frame in a way that enables each node to transmit during its own timeslot, and arbitrate between nearby nodes that wish to communicate concurrently. We strive to provide the needed MAC protocol properties, using limited radio and clock settings, i.e., no external reference for collision detection, time or position. Note that ad hoc networks often do not consider collision detection mechanisms, and external references are subject to signal loss. For these settings, we demonstrate that there is no solution for the studied problem when the frame size is $\tau < \max\{2\delta, \chi_2\}$, where δ is the maximal node degree, and χ_2 is the chromatic number for distance-2 vertex coloring. The main result is the existence of collision-free self-stabilizing TDMA algorithms that use constant frame size of $\tau > \max\{4\delta, X_2\} + 1$, where $X_2 \geq \chi_2$ is a number that depends on the coloring algorithm in use. To the best of our knowledge, we are the first to study the problem of self-stabilizing TDMA timeslot allocation without external reference. The algorithm simulations demonstrate feasibility in a way that is close to the practical realm.

Wireless ad hoc networks have a dynamic nature that is difficult to predict. This gives rise to many fault-tolerance issues and requires efficient solutions. These networks are also subject to transient faults due to temporal malfunctions in hardware, software and other short-lived violations of the assumed system settings, such as changes to the communication graph topology. We focus on fault-tolerant systems that recover after the occurrence of transient faults, which can cause an arbitrary corruption of the system state (so long as the program's code is still intact). These *self-stabilizing* [9] design criteria simplify the task of the application designer when dealing with low-level complications, and provide an essential level of abstraction. Consequently, the application design can easily focus on its task – and knowledge-driven aspects.

ALOHAnet protocols [1] are pioneering MAC algorithms that let each node select one timeslot per TDMA frame at random. In the Pure Aloha protocol, nodes may transmit at any point in time, whereas in the Slotted Aloha version, the transmissions start at the timeslot beginning. The latter protocol has a shorter period during which packets may collide, because each transmission can collide only with transmissions that occur within its timeslot, rather than with two consecutive timeslots as in the Pure Aloha case. Note that the random access approach of ALOHAnet cannot provide constant communication delay. Distinguished nodes are often

The work of this author was partially supported by the EC, through project FP7-STREP-288195, KARYON (Kernel-based ARchitecture for safetY-critical cONtrol). This work appears as a technical report in [26].

978-1-4799-5258-8/14/$31.00 ©2014 IEEE

used when the application requires bounded communication delays, e.g., IEEE 802.15.4 and deterministic self-stabilizing TDMA [3, 16]. Without such external references, the TDMA algorithms have to align the timeslots while allocating them. Existing algorithms [5] circumvent this challenge by assuming that $\tau/(\Delta+1) \geq 2$, where Δ is an upper bound on the number of nodes with whom any node can communicate with using at most one intermediate node for relaying messages. This guarantees that every node can transmit during at least one timeslot, s, such that no other transmitter that is at most two hops away, also transmits during s. However, the $\tau/(\Delta+1) \geq 2$ assumption implies bandwidth utilization that is up to $\mathcal{O}(\delta)$ times lower than the proposed algorithm, because $\Delta \in \mathcal{O}(\delta^2)$.

As a basic result, we show that $\tau/\delta \geq 2$, and as a complement to this lower bound, we focus on considering the case of $\tau/\delta \geq 4$. We present a collision-free self-stabilizing TDMA algorithm that uses constant frame size of τ. We show that it is sufficient to guarantee that collision freedom for a single timeslot, s, and a *single* receiver, rather than *all* neighbors. This narrow opportunity window allows timeslot alignment and, after convergence, there are no collisions of any kind.

Related work Herman and Zhang [13] assume constant bounds on the communication delay and present self-stabilizing clock synchronization algorithms for wireless ad hoc networks. Herman and Tixeuil [12] assume access to synchronized clocks and present the first self-stabilizing TDMA algorithm for wireless ad hoc networks. They use external reference for dividing the radio time into timeslots and assign them according to the neighborhood topology. The self-stabilization literature often does not answer the causality dilemma of “which came first, synchronization or communication” that resembles Aristotle’s *“which came first, the chicken or the egg?”* dilemma. On one hand, existing clock synchronization algorithms often assume the existence of MAC algorithms that offer bounded communication delay, e.g. [13], but on the other hand, existing MAC algorithms that provide bounded communication delay, often assume access to synchronized clocks, e.g. [12]. We propose a bootstrapping solution to the causality dilemma of “which came first, synchronization or communication”, and discover convergence criteria that depend on τ/δ.

The *converge-to-the-max synchronization* principle assumes that nodes periodically transmit their clock value, $ownClock$. Whenever they receive clock values, $receivedClock > ownClock$, that are greater than their own, they adjust their clocks accordingly, i.e., $ownClock \leftarrow receivedClock$. Herman and Zhang [13] assume constant bounds on the communication delay and demonstrate convergence. Basic radio settings do not include constant bounds on the communication delay. We show that the converge-to-the-max principle works when given bounds on the expected communication delay, rather than constant delay bounds, as in [13].

The proposal in [11] considers shared variable emulation. Several self-stabilizing algorithms adopt this abstraction, e.g., a generalized version of the dining philosophers problem for wireless networks in [7], topology discovery in anonymous networks [21], random distance-k vertex coloring [22], deterministic distance-2 vertex coloring [4], two-hop conflict resolution [27], a transformation from central demon models to distributed scheduler ones [29], to name a few. The aforementioned algorithms assume that if a node transmits infinitely many messages, all of its communication neighbors will receive infinitely many of them. We do not make such assumptions about *(underlying) transmission fairness*. We assume that packets, from transmitters that are at most two hops apart, can collide *every time*.

The authors of [17] present a MAC algorithm that uses convergence from a random starting state (inspired by self-stabilization). In [18, 24], they use network simulators for evaluating self-$\star$ MAC algorithms. A self-stabilizing TDMA algorithm, that accesses external time references, is presented in [19]. Simulations are used for evaluating the heuristics of MS-ALOHA [28] for dealing with timeslot exhaustion by adjusting the nodes’ individual transmission signal strength. The authors of [2, 10] assume timeslot alignment, access to collision detection mechanisms and complete communication graphs and they study the problem of transmission rate control for optimal channel throughput of Slotted ALOHA. We do not require these assumptions and offer, after a bounded convergence period, a deterministic transmission schedule. We provide analytical proofs and consider basic radio settings. The results presented in [8, 15] do not consider the time it takes the algorithm to converge, as we do. We mention a number of MAC algorithms that consider onboard hardware support, such as receiver-side collision detection [5, 6, 8, 28, 31]. We consider merely basic radio technology that is commonly used in wireless ad hoc networks. The MAC algorithms in [30, 31] assumes the accessibility of an external time or geographical references or the node trajectories, e.g., Global Navigation Satellite System (GNSS). We instead integrate the TDMA timeslot alignment with clock synchronization.

Our contribution Given a maximal node degree, δ, we consider the problem of the existence of collision-free self-stabilizing TDMA algorithms that use constant frame size of τ. In the context of self-stabilizing systems that have no external reference, we are the first to study this problem (to the best of our knowledge). The proposed self-stabilizing and bootstrapping algorithm answers the causality dilemma of synchronization and communication.

For settings that have no assumptions about fairness and external reference existence, we establish a basic limit on the bandwidth utilization of TDMA algorithms in wireless ad hoc networks (Section III). Namely, $\tau < \max\{2\delta, \chi_2\}$, where χ_2 is the chromatic number for distance-2 vertex coloring. We note that the result holds for general graphs with a clearer connection to bandwidth utilization for the cases of tree graphs ($\chi_2 = \delta + 1$) and planar graphs [23] ($\chi_2 = 5\delta/3 + \mathcal{O}(1)$).

We prove the existence of collision-free self-stabilizing TDMA algorithms that use constant frame size of τ without assuming the availability of external references (Section IV).

The convergence period is within $\mathcal{O}(\mathsf{diam} \cdot \tau^2\delta + \tau^3\delta)$ steps starting from an arbitrary configuration, where diam is the network diameter. We note that in case the system happens to have access to external time references, i.e., start from a configuration in which clocks are synchronized, the convergence time is within $\mathcal{O}(\tau^3)$, and $\mathcal{O}(\tau^3\delta)$ steps when $\tau > 2\Delta$, and respectively, $\tau > \max\{4\delta, \Delta + 1\}$. We also demonstrate convergence via simulations that take uncertainties into account, such as (local) computation time.

II. System Settings

The system consists of a set, $P := \{p_i\}$, of communicating entities, which we call *nodes*. An upper bound, $\nu > |P|$, on the number of nodes in the system is known. Subscript font is used to point out that X_i is p_i's variable (or constant) X. Node p_i has a unique identifier, id_i, that is known to p_i but not necessarily by $p_j \in P \setminus \{p_i\}$.

Communication graphs At any instance of time, the ability of any pair of nodes to communicate, is defined by the set, $\delta_i \subseteq P$, of *(direct) neighbors* that node $p_i \in P$ can communicate with directly. The system can be represented by an undirected network of directly communicating nodes, $G := (P, E)$, named the *communication graph*, where $E := \{\{p_i, p_j\} \in P \times P : p_j \in \delta_i\}$. We assume that G is connected. For $p_i, p_j \in P$, we define the distance, $d(p_i, p_j)$, as the number of edges in an edge minimum path connecting p_i and p_j. We denote by $\Delta_i := \{p_j \in P : 0 < d(p_i, p_j) \leq 2\}$ the 2-neighborhood of p_i, and the upper bounds on the sizes of δ_i and Δ_i are denoted by $\delta \geq \max_{p_i \in P}(|\delta_i|)$, and respectively, $\Delta \geq \max_{p_i \in P}(|\Delta_i|)$. Note that $p_i \in \delta_i$ and $p_i \in \Delta_i$. We assume that $\mathsf{diam} \geq \max_{p_i, p_j \in P} d(p_i, p_j)$ is an upper bound on the network diameter.

Synchronization The nodes have fine-grained clock hardware (with arbitrary clock offset upon system start). For the sake of presentation simplicity, our work considers zero clock skews. We assume that the *clock* value, $C \in [0, c-1]$, and any timestamp in the system have c states. The pseudo-code uses the $GetClock()$ function that returns a timestamp of C's current value. Since the clock value can overflow at its maximum, and wrap to the zero value, arithmetic expressions that include timestamp values are module c, e.g., the function $AdvanceClock(x) := C \leftarrow (C + x) \bmod c$ adds x time units to clock value, C, modulo its number of states, c. We assume that the maximum clock value is sufficiently large, $c \gg \mathsf{diam}\tau^2$, to guarantee convergence of the clock synchronization algorithm, before the clock wrap around. We say that the clocks are *synchronized* when $\forall p_i, p_j \in P : C_i = C_j$, where C_i is p_i's clock value.

Periodic pulses invoke the MAC protocol, and divide the radio time into *(broadcasting) timeslots* of ξ time units in a way that provides sufficient time for the transmission of a single packet. We group τ timeslots into *(broadcasting) frames*. The pseudo-code uses the event $timeslot(s)$ that is triggered by the event $0 = C_i \bmod \xi$ and $s := C_i \div \xi \bmod \tau$ is the *timeslot number*, where $\div$ is the integer division.

Operations The communication allows a message exchange between the sender and the receiver. After the sender, p_i, fetches message $m \leftarrow MAC\_fetch_i()$ from the upper layer, and before the receiver, p_j, delivers it to the upper layer in $MAC\_deliver_j(m)$, they exchange m via the operations $transmit_i(m)$, and respectively, $m \leftarrow receive_j()$. We model the communication channel, $q_{i,j}$ (queue), from node p_i to node $p_j \in \delta_i$ as the most recent message that p_i has sent to p_j and that p_j is about to receive, i.e., $|q_{i,j}| \leq 1$. When p_i transmits message m, the operation $transmit_i(m)$ inserts a copy of m to every $q_{i,j}$, such that $p_j \in \delta_i$. Once m arrives, p_j executes $receive()$ and returns the tuple $\langle i, t_i, t_j, m \rangle$, where $t_i = C_i$ and $t_j = C_j$ are the clock values of the associated $transmit_i(m)$, and respectively, $m \leftarrow receive_j()$ calls. We assume zero propagation delay and efficient time-stamping mechanisms for t_i and t_j. Moreover, the timeslot duration, ξ, allows the transmission and reception of at least a single packet, see Property 1.

Property 1. Let $p_i \in P$, $p_j \in \delta_i$. At any point in time t_i in which node p_i transmits message m for duration of ξ, node p_j receives m if there is no node $p_k \in (\delta_i \cup \delta_j) \setminus \{p_i\}$ that transmits starting from time t_k with duration ξ such that $[t_i, t_i + \xi)$ and $[t_k, t_k + \xi)$ are intersecting.

This means a node can receive a message if no node in the neighborhood of the sender and no node in the neighborhood of the receiver is transmitting concurrently.

Interferences Wireless communications are subject to interferences when two or more neighboring nodes transmit *concurrently*, i.e., the packet transmission periods overlap or intersect. We model communication interferences, such as unexpected peaks in ambient noise level and concurrent transmissions of neighboring nodes, by letting the *(communication) environment* to selectively omit messages from the communication channels. We note that we do *not* consider any error (collision) indication from the environment.

The environment can use the operation $omission_{i,j}(m)$ for removing message m from the communication channel, $q_{i,j}$, when p_i's transmission of m to $p_j \in \delta_i$ is concurrent with the one of $p_k \in \Delta_i$. Immediately after $transmit_i(m)$, the environment selects a subset of p_i's neighbors, $Omit_m \subseteq \delta_i$, removes m from $q_{i,j} : p_j \in Omit_m$ and by that it prevents the execution of $m \leftarrow receive_j()$. Note that $Omit_m = \delta_i$ implies that no direct neighbor can receive message m.

Self-stabilization Every node, $p_i \in P$, executes a program that is a sequence of *(atomic) steps*, a_i. The state, st_i, of node $p_i \in P$ includes p_i's variables, including the clocks and the program control variables, and the communication channels, $q_{i,j} : p_j \in \delta_i$. The *(system) configuration* is a tuple $c := (st_1, \ldots, st_{|P|})$ of node states. Given a system configuration, c, we define the set of *applicable steps*, $a = \{a_i\}$, for which p_i's state, st_i, encodes a non-empty communication channel or an expired timer. An *execution* is an unbounded alternating sequence $R := (c[0], a[0], c[1], a[1], \ldots)$ (Run) of configurations $c[k]$, and applicable steps $a[k]$ that are taken by the algorithm and the environment. The task $\mathcal{T}$ is a set

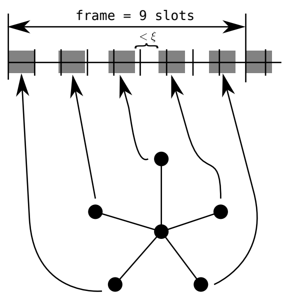

Fig. 1. The outer five nodes are covering nine timeslots. The top horizontal line and its perpendicular marks depict the radio time division according to the central node, p_δ. The gray boxes depict the radio time covered by the leaf nodes, $p_i \in L$.

of specifications and LE (legal execution) is the set of all executions that satisfy $\mathcal{T}$. We say that configuration c is *safe*, when every execution that starts from it is in LE. An algorithm is called *self-stabilizing* if it reaches a safe configuration within a bounded number of steps.

Task definition We consider the task $\mathcal{T}_{\text{TDMA}}$, that requires all nodes, p_i, to have timeslots, s_i, that are uniquely allocated to p_i within Δ_i. We define LE_{TDMA} to be the set of legal executions, R, for which $\forall p_i \in P : (p_j \in P \Rightarrow C_i = C_j) \wedge (((s_i \in [0, \tau - 1]) \wedge (p_j \in \Delta_i)) \Rightarrow s_i \neq s_j)$ holds in all of R's configurations. We note that for a given finite τ, there are communication graphs for which $\mathcal{T}_{\text{TDMA}}$ does not have a solution, e.g., the complete graph, $K_{\tau+1}$, with $\tau + 1$ nodes. In Section III, we show that the task solution can depend on the (arbitrary) starting configuration, rather than just the communication graph.

III. Basic Results

We establish a bandwidth utilization limitation for TDMA algorithms in wireless ad hoc networks, i.e., $\tau < \max\{2\delta, \chi_2\}$, where χ_2 is the chromatic number for distance-2 vertex coloring. Suppose that $\delta \in \mathbb{N}$, $\tau < 2\delta$, the communication graph, $G := (\{p_0, \dots p_\delta\}, E)$, has the topology of a star, where the node p_δ is the center (root) node and $E := \{p_\delta\} \times L$, where $L := \{p_0, \dots p_{\delta-1}\}$ are the leaf nodes. The illustrative example in Figure 1 explains why no algorithm can converge. (The proof details appear in [26].) Note that the gap between p_i's transmission and the following transmission by $p_{(i+1) \bmod \delta}$ is less than the slot size ξ. This pattern of a frame repeats, because only p_δ receives these messages transmitted by the leaves and p_δ does not have a timeslot assigned. According to Property 1, all p_δ transmission attempts can fail.

IV. Self-stabilizing TDMA Allocation and Alignment Algorithm

We propose Algorithm 1 as a self-stabilizing algorithm for the $\mathcal{T}_{\text{TDMA}}$ task. The nodes transmit data packets, as well as control packets. Data packets are sent by active nodes during their data packet timeslots. The passive nodes listen to the active ones and do not send data packets. Both active and passive nodes use control packets, which include the reception time and the sender of recently received packets from direct neighbors. Each node aggregates the frame information it receives. It uses this information for avoiding collisions, acknowledging packet transmission and resolving hidden node problems. A passive node, p_i, can become active by selecting random timeslots, s_i, that are not used by active nodes. Then p_i sends a control packet in s_i and waiting for confirmation. Once p_i succeeds, it becomes an active node that uses timeslot s_i for transmitting data packets. Node p_i becomes passive whenever it learns about conflicts with nearby nodes, e.g., due to a transmission failure.

The hidden node problem refers to cases in which node p_i has two neighbors, $p_j, p_k \in \delta_i$, that use intersecting timeslots. The algorithm uses random back off techniques for resolving this problem in a way that assures at least one successful transmission from all active and passive nodes within $\mathcal{O}(\tau)$, and respectively, $\mathcal{O}(1)$ frames in expectation. The passive nodes count a random number of unused timeslots before transmitting a control packet. The active nodes use their clocks for defining frame numbers. They count down only during TDMA frames whose numbers are equal to s_i, where $s_i \in [0, \tau - 1]$ is p_i's data packet timeslot. These back off processes connect all direct neighbors and facilitate clock synchronization, timeslot alignment and timeslot assignment. During legal executions, in which all nodes are active, there are no collisions and each node transmits one control packet once every τ frames.

Algorithm details The node status, $status_i$, is either active or passive. When it is active, variable s_i contains p_i timeslot number. The frame information is the set $FI_i := \{id_k, type_k, occurrence_k, rxTime_k\}_k \subset \mathcal{FI} = \text{ID} \times \{\text{message}, \text{welcome}\} \times \{\text{remote}, \text{local}\} \times \mathbb{N}$ that contains information about recently received packets, where $\text{ID} := \{\perp\} \cup \mathbb{N}$ is the set of possible ids and the null value denoted by $\perp$. An element of the frame information contains the id of the sender id_k. The type $type_k = \text{message}$ indicates that the sender was active. For a passive sender $type_k = \text{welcome}$ indicates that there was no known conflict when this element was added to the local frame information. If $occurrence_k = \text{local}$, the corresponding packet was received by p_i, otherwise it was copied from a neighbor. The reception time $rxTime_k$ is the time when this packet was received, regarding the local clock C_i, i.e., it is updated whenever the local clock is updated. The algorithm considers the frame information to select an unused timeslot. An entry in the frame information with timestamp t covers the time interval $[t, t + \xi)$.

Nodes transmit control packets according to a random back off strategy for collision avoidance. The passive node, p_i, chooses a random back off value, stores it in the variable $wait_i$, and uses $wait_i$ for counting down the number of timeslots that are available for transmissions. When $wait_i = 0$, node p_i uses the next unused timeslot according to its frame information. During back off periods, the algorithm uses the variables $wait_i$ and $waitAdd_i$ for counting down to zero. The process starts when node p_i assigns $wait_i \leftarrow waitAdd_i + r$, where r is a random choice from $[1, 3\Delta]$, and updates

Algorithm 1: Self-stabilizing TDMA Allocation, code for node p_i

$status_i \in \{\text{active}, \text{passive}\}$; /* current node status */
$s_i \in [0, \tau - 1]$; /* current data packet timeslot */
$wait_i, waitAdd_i \in [0, maxWait]$; /* current back off countdown */
$FI_i := \{id_k, type_k, occurrence_k, rxTime_k\}_k \subset \mathcal{FI}$; /* frame information */
$timeOut$; /* constant, age limit of elements in FI_i */
$BackOff() :=$ **let** $(tmp, r) \leftarrow (waitAdd_i,\ random([1,\ 3\Delta]))$; **return** $(r + tmp,\ 3\Delta - r)$; /* reset backoff counter */
$frame() := (GetClock() \div \xi\tau) \bmod \tau$,; /* the current frame number */
$Slot(t) := (t \div \xi \bmod \tau), s() := Slot(GetClock())$; /* convert time to slot numbers */
$Local(set) := \{\langle \bullet, \text{local}, \bullet \rangle \in set\}$; /* dist-1 neighbors in set */
$Used(set) := \bigcup_{\langle \bullet, t_k \rangle \in set} [Slot(t_k), Slot(t_k + \xi - 1)]$; $Unused(set) := [0, \tau - 1] \setminus Used(set)$; /* set of (un)used slots */
$ConflictWithNeighbors(set) := (\nexists_{\langle id_i \bullet \rangle \in set} \vee\ s_i \in [Slot(t_i), Slot(t_i + \xi)] \vee$
$\exists_{\langle k, \bullet, rxTime \rangle \in set, k \neq id_i} : s_i \in [Slot(rxTime - t_j + t_i), Slot(rxTime - t_j + t_i + \xi)])$; /* check for conflicts */
$AddToFI(set, o) := FI_i \leftarrow FI_i \cup \{\langle x, y, \text{remote}, z' \rangle : \langle x, y, \bullet, z \rangle \in set, z' := (z + \max\{0, o\}) \bmod c, z' \leq_{timeOut} C_i\}$; /* $set + FI_i$ */
$IsUnused(s) := s \in Unused(FI_i) \vee (Unused(FI_i) = \emptyset \wedge s \in Unused(Local(FI_i)))$; /* is s an unused slot? */

1 **upon** $timeslot()$ **do**
2 **if** $s() = s_i \wedge status_i =$ active **then transmit**$(\langle status_i, Local(FI_i), MAC\_fetch() \rangle)$; /* send data packet */
3 **else if** $\neg(status_i =$ active $\wedge frame() \neq s_i)$ **then** /* check if our frame */
4 **if** $IsUnused(s()) \wedge wait_i \leq 0$ **then** /* send control packet */
5 **transmit**$(\langle status_i, Local(FI_i), 0 \rangle)$;
$\langle wait_i, waitAdd_i \rangle \leftarrow BackOff()$; /* prepare for next control packet */
6 **if** $status_i \neq$ active **then** $\langle s_i, status_i \rangle \leftarrow \langle s(), \text{active} \rangle$;
7 **else if** $wait_i > 0 \wedge IsUnused((s() - 1) \bmod \tau)$ **then** $wait_i \leftarrow \max\{0, wait_i - 1\}$; /* count down */
8 $FI_i \leftarrow \{\langle \bullet, rxTime \rangle \in FI_i : rxTime \leq_{timeOut} GetClock()\}$; /* remove old entries from FI_i */

9 **upon** $\langle j, t_j, t_i, \langle status_j, FI_j, m' \rangle \rangle \leftarrow$ **receive**() **do**
10 **if** $ConflictWithNeighbors(FI_j) \wedge status_i =$ active **then** /* conflicts? */
$\langle \langle wait_i, waitAdd_i \rangle, status \rangle \leftarrow \langle BackOff(), \text{passive} \rangle$; /* get passive */
11 **if** $status_j =$ active **then** /* active node acknowledge */
12 **if** $m' \neq \perp$ **then** $FI_i \leftarrow \{\langle id_i, \bullet \rangle \in FI_i : id_i \neq j\} \cup \{\langle j, \text{message}, \text{local}, t_i \rangle\}$;
13 **else if** $t_j = t_i \wedge Slot(t_j) \notin Used(FI_i)$ **then** /* passive node acknowledge */
14 $FI_i \leftarrow \{\langle id_i, \bullet \rangle \in FI_i : id_i \neq j\} \cup \{\langle j, \text{welcome}, \text{local}, t_i \rangle\}$;
15 **if** $t_i < t_j$ **then** /* converge-to-the-max */
16 $AdvanceClock(t_j - t_i)$; /* adjust clock */
17 $FI_i \leftarrow \{\langle \bullet, (rxTime + t_j - t_i) \bmod c \rangle : \langle \bullet, rxTime \rangle \in FI_i\}$; /* shift timestamps in FI_i */
18 $\langle \langle wait_i, waitAdd_i \rangle, status_i \rangle \leftarrow \langle BackOff(), \text{passive} \rangle$; /* get passive */
19 $AddToFI(FI_j, t_i - t_j)$; /* Aggregate information on used timeslots */
20 **if** $m' \neq \perp$ **then** $MAC\_deliver(m')$;

$waitAdd_i \leftarrow 3\Delta - r$, cf. $BackOff()$.

The node clock is the basis for the frame and timeslot starting times, cf. $frame()$, and respectively, $s()$, and also for a given timeslot number, cf. $Slot(t)$. When working with the frame information, set, it is useful to have restriction by local occupancies, cf. $Local(set)$ and to list the sets of used and unused timeslots, cf. $Used(set)$, and respectively, $Unused(set)$. We check whether an arriving frame information, set, conflicts with the local frame information that is stored in FI_i, cf. $ConflictWithNeighbors(set)$, before merging them together, cf. $AddToFI(set, offset)$, after updating the timestamps in set, which follow the sender's clock.

Node p_i can test whether the timeslot number s is available according to the frame information in FI_i and p_i's clock. Since Algorithm 1 complements the studied lower bound (Section III), the test in $IsUnused(s)$ checks whether FI_i encodes a situation in which there are no unused timeslots. In that case, $IsUnused(s)$ tests whether we can say that s is unused when considering only transmissions of direct neighbors. The correctness proof considers the cases in which $\tau > 2\Delta$ and $\tau > \max\{4\delta, \Delta + 1\}$. For the former case, we show that there is always an unused timeslot s' that is not used by any neighbor $p_j \in \Delta_i$. For the latter case, the proof shows that for any neighbor $p_j \in \delta_i$, there is a timeslot s'' for which there is no node $p_k \in \delta_i \cup \delta_j \cup \{p_j, p_i\}$ that transmits during s''. The code of Algorithm 1 considers two events: (1) periodic timeslots (line 1) and (2) reception of a packet (line 9).

(1) *$timeslot()$, line 1:* Actives nodes transmit their data packets upon their timeslot (line 2). Passive nodes transmit control packets when the back off counter, $wait_i$, reaches zero (line 5). Note that passive nodes count only when the local frame information says that the previous timeslot was unused (line 7). Active nodes also send control packets, but rather than counting all unused timeslots, they count only the unused timeslots that belong to frames with a number that matches the timeslot number, i.e., $frame() = s_i$ (line 3).

(2) ***receive**(), line 9:* Active nodes, p_i, become passive when they identify conflicts in FI_j between their data packet timeslots, s_i, and data packet timeslots, s_j of other nodes $p_j \in \Delta_i$ (line 10). When the sender is active, the receiver records the related frame information. Note that the payload of data packets is not empty in line 12, c.f., $m' \neq \perp$. Passive nodes, p_j, aim to become active. In order to do that, they need to send a control packet during a timeslot

that all nearby nodes, p_i, view as unused, i.e., $Slot(t) \notin Used(FI_i)$, where t is the packet sending time. Therefore, when the sender is passive, and its data packet timeslots are aligned, i.e., $t_i = t_j$, node p_i welcomes p_i's control packet whenever $Slot(t_j) \notin Used(FI_i)$. Algorithm 1 uses a self-stabilizing clock synchronization algorithm that is based on the converge-to-the-max principle. When the sender clock value is higher (line 15), the receiver adjusts its clock value and the timestamps in the frame information set, before validating its timeslot, s_i, (lines 16 to 18). The receiver can now use the sender's frame information and payload (lines 19 to 20).

V. Correctness

The existence proof of a collision-free self-stabilizing TDMA algorithms is given by this section (Theorem 3), and it starts by showing the existence of unused timeslots considering the cases in which $\tau > 2\Delta$ and $\tau > \max\{4\delta, \Delta + 1\}$. This facilitates the proof of network connectivity, clock synchronization (Theorem 1) and bandwidth allocation (Theorem 2). Due to the space limit, some parts of the proofs appear in [26].

Communication among neighbors is possible only when there are timeslots that are free from transmissions by nodes in the local neighborhood. We start by assuming that $\tau > 2\Delta$ and show that every node, $p_i \in P$, has an unused timeslot, s, with respect to p_i's clock. This satisfies the conditions of Property 1 with respect to *all* of p_i's neighbors $p_j \in \delta_i$. We then continue by assuming that $\tau > \max\{4\delta, \Delta+1\}$ and showing that every node, $p_i \in P$, has an unused timeslot, s, with respect to p_i's clock. This satisfies the conditions of Property 1 with respect to *one* of p_i's neighbors $p_j \in \delta_i$, rather than all p_i's neighbors $p_j \in \delta_i$. This implies that there is a single timeslot, s, that is unused with respect to the clocks of node p_i and *all*, respectively, *one* of p_i's neighbors. Lemma 1 shows that the control packet exchange provides network connectivity. The proof shows that we can apply the analysis of [14], because the back off process of a passive node counts r unused timeslots, where r is a random choice in $[1, 3\Delta]$. The lemma statement denotes the latency period by $\ell := (1 - e^{-1})^{-1}$.

A frame information set FI_i is locally consistent if all local entries can be used to predict a transmission of a node in δ_i. A locally consistent frame information set FI_j is consistent if all remote entries can be used to predict transmissions of nodes in $\Delta_i \setminus \delta_i$.

Lemma 1 (The proof appears in [26])**.** *Let R be an execution of Algorithm 1 that starts from an arbitrary configuration $c[x]$. Then there is a suffix R' of R that starts from a configuration $c[x + \mathcal{O}(timeOut)]$ such that a passive node p_i receives a message from all nodes in δ_i within finite time.*

Proof Sketch. We show that every execution reaches a configuration c'' such that in the suffix R' of R starting from c'' neighbors can exchange packets within finite time. The proof shows the existence of a free timeslot for all nodes and their clocks. Then we show that it is also marked as free in the node's frame information, since stale entries, i.e., entries for which there is no neighbor that has corresponding id and timeslot, might block it. Therefore, we study the behavior of elements in the frame information set FI_i during the different clock update steps. Furthermore, we show how elements in the frame information set are propagated between neighboring nodes. We see how the age of each entry is either increasing monotonically, or updated when new packets are received. Thus all stale entries are deleted after C_i advances by $timeOut$ clock steps. Additionally, after all stale entries are deleted there is maximal one entry in FI_i for each neighbor $p_j \in \Delta_i$. Thus, p_i sees the free timeslot. □

After showing network connectivity (Lemma 1), we give an expectation for the communication delay, i.e., how long does it take to successfully send a control packet to a neighbor. Namely, every expected γ frames, p_i receives at least one message from all active neighbor, $p_k \in \delta_i$, where γ is $3\Delta\tau\ell$ frames if $\tau > 2\Delta$, or $3\Delta\tau\ell\delta$ frames if $\tau > \max\{4\delta, \Delta + 1\}$.

Theorem 1 shows that the converge-to-the-max principle works when given bounds on the expectation of the communication delay, rather than constant delay bounds, as in [13].

Theorem 1 (The proof appears in [26])**.** *Let R be an execution of Algorithm 1 that starts from an arbitrary configuration $c[x]$. Within expected $\Phi := \gamma \cdot \mathsf{diam}$ frames, a configuration $c[x^{synchro}]$ is reached after which all clocks are synchronized.*

Once the clocks are synchronized, and the TDMA timeslots are aligned, Algorithm 1 allocates the bandwidth within $\mathcal{O}(\gamma^2)$ frames using distance-2 coloring, see Theorem 2.

Theorem 2. *Let R be an execution that starts from an arbitrary configuration $c[x^{synchro}]$ in which all clocks are synchronized. Within expected γ^2 frames from $c[x^{synchro}]$, the system reaches a configuration, $c[x^{alloc}]$, in which each node $p_i \in P$ has a timeslot that is unique in Δ_i.*

Proof Sketch. We have showed that active nodes get feedback within an expected time. Active nodes with positive feedback stay active, but conflicting active nodes get a negative feedback and change their status to passive. Passive nodes are transmitting from time control packets. They are successful and stay active on this timeslot with probability $1 - 1/e$. Otherwise, they get a negative feedback within expected γ frames. Hence, the number of active nodes without conflicts is monotonically increasing until every node is active.

The convergence time of this timeslot assignment is dominated by the time for a successful transmission and the time for a negative feedback in case of a unsuccessful transmission. This leads to an expected convergence time of γ^2. □

The proof of Theorem 3 is concluded by showing that configuration, $c[x^{alloc}]$ (Theorem 2), is a safe configuration with respect to LE_{TDMA}, see Lemma 2.

Lemma 2. *Configuration $c[x]$ is a safe configuration with respect to LE_{TDMA}, when (1) $\forall_{p_i, p_j \in P} : C_i = C_j$, (2) $\forall_{p_i \in P} : status_i = \mathsf{active}$, (3) $\forall_{p_i \in P} \forall_{p_j \in \Delta_i} : s_i \neq s_j$, (4) $\forall_{p_i \in P}$:*

$\forall_{p_j \in \Delta_i \cup \{p_i\}} \exists!_{\langle id, \mathsf{message}, \bullet \rangle \in FI} : id = id_j$.

Proof. First we check that $c[x]$ is legal regarding LE_{TDMA}.The conditions (1) and (3) are coinciding with the conditions of a legal execution LE_{TDMA}. Condition (2) is necessary in combination to (3) to ensure that the TDMA slot stored in s_i is valid. Condition (4) restricts configurations in LE_{TDMA} by ensuring that nodes are aware of their neighbors.

We conclude by proofing that the conditions of this Lemma hold for all configurations following $c[x]$ in an execution R of Algorithm 1. Since (1) holds in $c[x]$, this means all clocks are synchronized, a node can never receive a clock value that is larger than its own, so the clock update step in line 16 is never executed. Thus for all following configurations to $c[x]$ in R condition (1) holds. A node changes its $status_i$ to passive when it updates its clock (line 16), or when it detects a conflict with the slot assignment (line 10). From (1) follows that there is no clock update. From (3) follows that in $c[x]$ there is no conflict with the slot assignment and from (4) that everyone is aware that there is no conflict. Furthermore, (4) implies that every node selects only slots for control packets that are free within the distance-2 neighborhood. Thus, in R there is never a message transmitted from that the receiver can detect a conflict and no conflict is introduced by sending a control packet on a data packet slot of a distance-2 neighbor. This proofs that $c[x]$ is a safe configuration for LE_{TDMA}. □

Corollary 1. *The configuration $c[x^{alloc}]$ is a safe configuration for the task $\mathcal{T}_{\text{TDMA}}$.*

Proof. We check that $c[x^{alloc}]$ fulfills the conditions of Lemma 2. Condition (1) follows from Theorem 1 and conditions (2), (3) and (4) are following from Theorem 2. □

Theorem 3. *Algorithm 1 is a self-stabilizing implementation of task $\mathcal{T}_{\text{TDMA}}$ that converges within $\mathcal{O}(\mathsf{diam}\tau\Delta\delta + (\Delta\tau\delta)^2)$ frames starting from an arbitrary configuration. In case the system happens to have access to external time references, i.e., start from a configuration in which clocks are synchronized, the convergence time is within $\mathcal{O}((\Delta\tau\delta)^2)$ frames.*

Proof Sketch. The proof shows that communication is possible after a constant number of clock steps. It bounds the expected communication delay to γ. Theorem 1 shows that after $\mathcal{O}(\gamma \cdot \mathsf{diam})$ frames a configuration $c[x^{synchro}]$ is reached where the clocks are synchronized. Theorem 2 shows that in $\mathcal{O}(\gamma^2)$ frames after $c[x^{synchro}]$, a configuration $c[x^{alloc}]$ is reached in which all nodes have an allocated timeslot. Corollary 1 shows that Algorithm 1 solves $\mathcal{T}_{\text{TDMA}}$ by reaching $c[x^{alloc}]$. □

VI. Experimental results

We demonstrate the implementation feasibility. We study the behavior of the proposed algorithm in a simulation model that takes into account timing uncertainties. Thus, we demonstrate feasibility in a way that is close to the practical realm.

The system settings (Section II) that we use for the correctness proof (Section V) assumes that any (local) computation can be done in zero time. In contrast to this, the simulations use the TinyOS embedded operating systems [20] and the Cooja simulation framework [25] for emulating wireless sensor nodes together with their processors. This way Cooja simulates the code execution on the nodes, by taking into account the computation time of each step. We implemented the proposed algorithm for sensor nodes that use IEEE 802.15.4 compatible radio transceivers. The wireless network simulation is according to the system settings (Section II) is based on a grid graph with $4 \geq \delta$ as an upper bound on the node degree and a random graph with $16 \geq \delta$ as an upper bound on the node degree. The implementation uses clock steps of 1 millisecond. We use a timeslot size of $\xi = 20$ clock steps, where almost all of this period is spent on transmission, packet loading and offloading. The frame size is $\tau = 16 \geq 4\delta$ timeslots for the grid graph and $\tau = 64 \geq 4\delta$ for the random graphs. For these settings, all experiments showed convergence, see Figure 2.

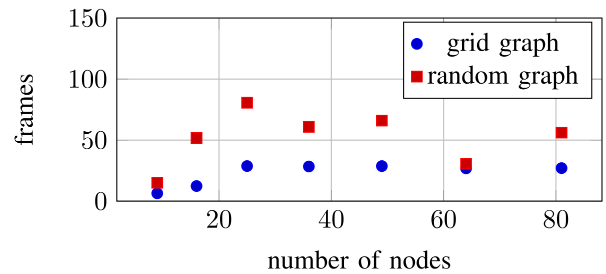

Fig. 2. The converges time in frames for different graphs. In the grid graph, nodes are placed on a lattice and connected to their four neighbors. The convergence times are the average over 16 runs that start each with random clock offsets. The random node graph is a unified disk graph with random node placement with maximal 16 neighbors pair node.

VII. Conclusions

This work considers fault-tolerant systems that have basic radio and clock settings without access to external references for collision detection, time or position, and yet require constant communication delay. We study collision-free TDMA algorithms that have uniform frame size and uniform timeslots and require convergence to a data packet schedule that does not change. By taking into account (local) computation time uncertainties, we observe that the algorithm is close to the practical realm. Our analysis considers the timeslot allocation aspects of the studied problem, together with transmission timing aspects. Interestingly, we show that the existence of the problem's solution depends on convergence criteria that include the ratio, τ/δ, between the frame size and the node degree. We establish that $\tau/\delta \geq 2$ as a general convergence criterion, and prove the existence of collision-free TDMA algorithms for which $\tau/\delta \geq 4$. This works shows how that self-stabilizing algorithms provide bootstrap solutions to the causality dilemma of "which came first, synchronization or communication". As future work, we suggest the study of this causality dilemma in communication networks that do not share a common reference for time collision arbitration.

REFERENCES

[1] N. Abramson. Development of the ALOHANET. *Info. Theory, IEEE Trans. on*, 31(2):119–123, 1985.

[2] B. T. An and E. Gelenbe. Control and improvement of the broadcast channel. In *Modelle für Rechensysteme*, pages 256–276, 1977.

[3] M. Arumugam and S. Kulkarni. Self-stabilizing deterministic time division multiple access for sensor networks. *AIAA Journal of Aerospace Computing, Info., and Comm. (JACIC)*, 3:403–419, 2006.

[4] J. R. S. Blair and F. Manne. An efficient self-stabilizing distance-2 coloring algorithm. *Theor. Comput. Sci.*, 444:28–39, 2012.

[5] C. Busch, M. Magdon-Ismail, F. Sivrikaya, and B. Yener. Contention-free MAC protocols for asynchronous wireless sensor networks. *Distrib. Comp.*, 21(1):23–42, 2008.

[6] H. A. Cozzetti and R. Scopigno. RR-Aloha+: a slotted and distributed MAC protocol for vehicular communications. In *Vehicular Networking Conference (VNC), 2009 IEEE*, pages 1 –8, Oct. 2009.

[7] P. Danturi, M. Nesterenko, and S. Tixeuil. Self-stabilizing philosophers with generic conflicts. *ACM Tran. Autonomous & Adaptive Systems (TAAS)*, 4(1), 2009.

[8] M. Demirbas and M. Hussain. A MAC layer protocol for priority-based reliable multicast in wireless ad hoc networks. In *BROADNETS*. IEEE, 2006.

[9] S. Dolev. *Self-Stabilization*. MIT Press, 2000.

[10] G. Fayolle, E. Gelenbe, and J. Labetoulle. Stability and optimal control of the packet switching broadcast channel. *J. ACM*, 24(3):375–386, 1977.

[11] T. Herman. Models of self-stabilization and sensor networks. In *5th International Workshop on Distributed Computing, IWDC 2003*, pages 205–214, 2003.

[12] T. Herman and S. Tixeuil. A distributed TDMA slot assignment algorithm for wireless sensor networks. In *ALGOSENSORS*, volume 3121 of *LNCS*, pages 45–58. Springer, 2004.

[13] T. Herman and C. Zhang. Best paper: Stabilizing clock synchronization for wireless sensor networks. In *8th Inter. Symp. on Stabilization, Safety, and Security of Distributed Systems (SSS'06)*, pages 335–349, 2006.

[14] J.-H. Hoepman, A. Larsson, E. M. Schiller, and P. Tsigas. Secure and self-stabilizing clock synchronization in sensor networks. *Theor. Comput. Sci.*, 412(40):5631–5647, 2011.

[15] A. Jhumka and S. S. Kulkarni. On the design of mobility-tolerant TDMA-based media access control (MAC) protocol for mobile sensor networks. In *4th International Conference on Distributed Computing and Internet Technology, ICDCIT 2007*, pages 42–53, 2007.

[16] S. S. Kulkarni and M. Arumugam. Transformations for write-all-with-collision model' . *Computer Communications*, 29(2):183–199, 2006.

[17] P. Leone, M. Papatriantafilou, and E. M. Schiller. Relocation analysis of stabilizing MAC algorithms for large-scale mobile ad hoc networks. In *5th Inter. Workshop Algo. Wireless Sensor Net. (ALGOSENSORS)*, pages 203–217, 2009.

[18] 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. In *12th Inter. Symp. on Stabilization, Safety, and Security of Distributed Systems (SSS'10)*, pages 468–488, 2010.

[19] P. Leone and E. M. Schiller. Self-stabilizing TDMA algorithms for dynamic wireless ad-hoc networks. *Int. J. Distributed Sensor Networks*, 639761, 2013.

[20] P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer, et al. Tinyos: An operating system for sensor networks. In *Ambient intelligence*, pages 115–148. Springer, 2005.

[21] T. Masuzawa and S. Tixeuil. On bootstrapping topology knowledge in anonymous networks. *ACM Trans. Auton. Adapt. Syst.*, 4(1):8:1–8:27, Feb. 2009.

[22] N. Mitton, E. Fleury, I. G. Lassous, B. Sericola, and S. Tixeuil. Fast convergence in self-stabilizing wireless networks. In *12th Int. Conf. Parallel and Distributed Systems (ICPADS'06)*, pages 31–38, 2006.

[23] M. Molloy and M. R. Salavatipour. A bound on the chromatic number of the square of a planar graph. *J. Comb. Theory, Ser. B*, 94(2):189–213, 2005.

[24] M. Mustafa, M. Papatriantafilou, E. M. Schiller, A. Tohidi, and P. Tsigas. Autonomous TDMA alignment for VANETs. In *76th IEEE Vehicular Technology Conf. (VTC-Fall'12)*, pages 1–5. IEEE, 2012.

[25] F. Osterlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt. Cross-level sensor network simulation with cooja. In *Local Computer Networks, Proceedings 2006 31st IEEE Conference on*, pages 641–648. IEEE, 2006.

[26] T. Petig, E. M. Schiller, and P. Tsigas. Self-stabilizing TDMA algorithms for wireless ad-hoc networks without external reference. *CoRR*, abs/1308.6475, 2013.

[27] S. Pomportes, J. Tomasik, A. Busson, and V. Vèque. Self-stabilizing algorithm of two-hop conflict resolution. In *12th Inter. Symp. on Stabilization, Safety, and Security of Distributed Systems (SSS'10)*, pages 288–302, 2010.

[28] R. Scopigno and H. A. Cozzetti. Mobile slotted aloha for VANETs. In *70th IEEE Vehicular Technology Conf. (VTC-Fall'09)*, pages 1 – 5, 2009.

[29] V. Turau and C. Weyer. Randomized self-stabilizing algorithms for wireless sensor networks. In H. de Meer and J. P. G. Sterbenz, editors, *IWSOS/EuroNGI*, volume 4124 of *LNCS*, pages 74–89. Springer, 2006.

[30] S. Viqar and J. L. Welch. Deterministic collision free communication despite continuous motion. In *5th Inter. Workshop Algo. Wireless Sensor Net. (ALGOSENSORS)*, pages 218–229, 2009.

[31] F. Yu and S. Biswas. Self-configuring TDMA protocols for enhancing vehicle safety with dsrc based vehicle-to-vehicle communications. *Selected Areas in Communications, IEEE Journal on*, 25(8):1526 –1537, oct. 2007.