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

Thomas Petig [†]
petig@chalmers.se

Elad M. Schiller[†]
elad@chalmers.se

Philippas Tsigas[†]
tsigas@chalmers.se

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 reference, we consider the problem of the existence of collision-free self-stabilizing TDMA algorithms that have constant communication delay of τ.

We demonstrate that this problem has no solution when $\tau \leq \max((2-\varepsilon)\delta, \chi_2)$, where $\varepsilon > 0$, and χ_2 is the chromatic number for distance-2 vertex coloring. We observe the bound relevance to the bandwidth utilization when the network topology is a planar graph. As a complement to this lower bound, we focus on proving the existence of probabilistic collision-free self-stabilizing TDMA algorithms that have constant communication delay 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).

1 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

[*]The work of this author was partially supported by the EC, through project FP7-STREP-288195, KARYON (Kernel-based ARchitecture for safetY-critical cONtrol).

[†]Computer Science and Engineering, Chalmers University of Technology, Sweden.

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. For these settings, we demonstrate that there is no solution for the studied problem when $\tau < \max((2-\varepsilon)\delta, \chi_2)$, where $\varepsilon > 0$, δ is a bound on the node degree, and χ_2 is the chromatic number for distance-2 vertex coloring. Note that $\chi_2 = \delta + 1$ and $\chi_2 = 5\delta/3 + \mathcal{O}(1)$ for the cases of tree, and respectively, planar graphs [23]. The main result is the existence of probabilistic collision-free self-stabilizing TDMA algorithms that have constant communication delay 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. 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).

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 *exposure 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 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 relatying message. This guarantees zero exposure period with respect to at least one timeslot, *s*, and *all* transmissions from transmitters that are at most two hops away, where Δ is a bound on their number. 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 $\Delta \in \mathcal{O}(\delta^2)$ (also for planar graphs [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 probabilistic collision-free self-stabilizing TDMA algorithm that has constant communication delay of τ. We show that it is sufficient to guarantee zero exposure period with respect to a single timeslot, s, and a *single* receiver, rather than *all* neighbors. This narrow opportunity window allows control packet exchange, and timeslot alignment. After convergence, there are no collisions of any kind, and each frame includes at most one control packet.

Related work The first proposal for a self-stabilizing TDMA algorithm for wireless ad hoc networks [11] considers external time reference for dividing the radio time. The proposal in [10] 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 [25], a transformation from central demon models to distributed scheduler ones [27], 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, 19, 24], the authors use computer network simulators for evaluating self-$\star$ MAC algorithms. A self-stabilizing TDMA algorithm, that accesses external time references, is presented in [20]. Simulations are used for evaluating the heuristics of MS-ALOHA [26] for dealing with timeslot exhaustion by adjusting the nodes' individual transmission signal strength. We provide analytical proofs and consider basic radio settings. The results presented in [8, 14] do not consider the time it takes the algorithm to convergence, as we do. We mention a number of MAC algorithms that consider onboard hardware support, such as receiver-side collision detection [5, 6, 8, 26, 29]. We consider merely basic radio technology that is commonly used in wireless ad hoc networks. The MAC algorithms in [28, 29] 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 an existing self-stabilizing clock synchronization technique for wireless ad hoc networks [12]. Unlike existing self-stabilizing MAC algorithms, our correctness proof considers the TDMA algorithm together its mechanisms for timeslot alignment and clock synchronization.

We consider minimal transmission duration, uniform packet priority, and a predefined frame size, τ, as in [22]. This assumption holds whenever the node degree is bounded by a known constant, the node distribution is sparse, or when nodes can adjust their transmission power in overpopulated areas. The system designer may decide to abandon real-time guarantees, and prefer the participation of all nodes over a uniform communication delay. Such design alternatives are beyond the scope of this work.

Our contribution Given a maximal node degree, δ, we consider the problem of the existence of collision-free self-stabilizing TDMA algorithms that have constant communication delay 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).

We make no assumptions about (underlying) transmission fairness, or external reference. Yet we assume that transmissions can collide *every time* the transmitters are at most two hops apart and their transmission times intersect (Section 2). For these settings, we establish a basic limit on the bandwidth utilization of TDMA algorithms in wireless ad hoc networks (Section 3). Namely, $\tau < \max(2\delta, \chi_2)$, where $\varepsilon > 0$, and χ_2 is the chromatic number for distance-2 vertex coloring. We note that $\chi_2 = \delta + 1$ holds for tree graphs, as well as $\chi_2 = 5\delta/3 + \mathcal{O}(1)$ for the case of planar graphs [23].

We prove the existence of collision-free self-stabilizing TDMA algorithms that have constant communication delay of τ without assuming the availability of external references (Section 4). The convergence period is within $\mathcal{O}(\mathsf{diam} \cdot \tau^2 + \tau^3\delta)$ 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, $\tau > 4\delta$.

2 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 to $p_j \in P \setminus \{p_i\}$.
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 (although the proposed algorithm borrows a clock synchronization algorithm [12] that does consider skews). We assume that the *clock* value, $C \in [0, c-1]$, and any time stamp 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, as in [12] Section 3.3. We say that the clocks are *synchronized* when $\forall p_i, p_j \in P : C_i = C_j$.

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 pulse, t_{i_y}, where C_i is p_i clock value, $t_{i_y} := C_i \div \xi$ (integer division) and $s := t_{i_y} (\bmod \tau)$ is the *timeslot number*.

Operations The communication allows 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, as in [12]. 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 a 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 during $[t_k, t_k + \xi)$, such that $[t_i, t_i + \xi)$ and $[t_k, t_k + \xi)$ intersect.

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. The environment can also choose to inform the sending node about this error.

The environment can use the operation $omission_{i,j}(m)$ for removing message m from the communication channel, $q_{i,j}$, when p_i 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. The environment uses Property 2 to decide whether to append an error, $TransmissionError$, to the error channel, e_i, which then in turn triggers the event $TransmissionError()$.

Property 2. Suppose that nodes $p_i, p_j \in P : p_i \in \delta_j$, concurrently transmit in $\alpha \in \mathbb{N}$ successive frames. We assume that p_j receives at least one message from p_i or that p_i receives a *(transmission) error*, i.e., invokes $TransmissionError_i()$, at least once.

The implementation of Property 2 can consider a basic technique for fault detection, say, based on the packet acknowledgment timeouts or ratio. Moreover, basic capabilities of the radio unit can also facilitate the implementation. For example, random separation time between timeslots [17, 19, 20]. Furthermore, RTS-CTS techniques can also address such implementation issues, as well as the use of dedicated hardware, for example, when using ultra-wideband for ranging and localization [15].

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, $q_{i,j} : p_j \in$

δ_i. The *(system) configuration* is a tuple $c := (st_1, \dots, st_{|P|})$ 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 $R := (c[0], a[0], c[1], a[1], \dots)$ (Run) of configurations $c[k]$, and applicable steps $a[k]$ that are taken by the algorithm and the environment. The task $\mathscr{T}$ is a set of specifications and LE (legal execution) is the set of all executions that satisfy $\mathscr{T}$. 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 $\mathscr{T}_{\text{TDMA}}$, that requires all nodes, p_i, to have data packet timeslots that are unique within Δ_i. We note that $\mathscr{T}_{\text{TDMA}}$'s requirements obviously hold when the ratio between the extended degree and the frame size is less than one, i.e., there is no timeslot exhaustion when $\forall p_i \in P : 1 < \tau/|\Delta_i|$. Therefore, the studied task also deals with timeslot exhaustion by delivering busy channel indications, $\perp$, to the nodes for which there were no timeslot left, and thus cannot transmit data packets. We define LE_{TDMA} to be the set of legal executions, R, for which $\forall p_i \in P :$ $(((s_i \in [0, \tau-1]) \wedge (p_j \in \Delta_i)) \Rightarrow s_i \neq s_j) \vee (s_i = \perp \Rightarrow \forall t \in [0, \tau-1]\ \exists p_j \in \Delta_i : s_j = t)$ holds in all of R's configurations.

3 Basic Results

We establish a basic limitation of the bandwidth utilization for TDMA algorithms in wireless ad hoc networks. Before generalizing the limitation, we present an illustrative example (Lemma 1) of a starting configuration for which $\tau < \max(2\delta, \chi_2)$, where $\varepsilon > 0$, and χ_2 is the chromatic number for distance-2 vertex coloring.

Lemma 1. *Let $n \in \mathbb{N}$, $\varepsilon > 0$ and $\tau \leq (2-\varepsilon)\delta$. Suppose that 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. There is a starting configuration, $c[x]$, and an execution, R, which follows $c[x]$, during which not all nodes, but one, are allocated with a timeslot, and yet the ones that are allocated follow a fixed schedule (with constant communication delay).*

Proof. We show by induction on x that all nodes in L are allocated with a timeslot in a fixed schedule, but p_δ is never properly allocated in R. Suppose that in the starting configuration, $c[x]$, all leaf nodes, $p_i \in L$, have a TDMA schedule in which their broadcasting timeslot is $s_i = s \in [0, \tau-1]$ (same value for all), and their clock values, $C_i = i(2-\varepsilon)\xi$, where ξ is the timeslot size. We assume that, in $c[x]$, there is no message in transit and that the communication queues are empty. Thus, p_i can only transmit a packet in the step $a[x]$ that immediately follows $c[x]$. However, as long as p_i's communication channels are empty, node p_i does not receive any packet. In this case, we say that p_i's timeslot does not change, and the schedule is fixed. Similarly, p_i clock values are not adjusted. Suppose that p_δ timeslot is $s_\delta = s \in [0, \tau-1]$. Let p_δ attempt to transmit in $a[x]$. We note that for any clock value that p_n may have in $c[x]$, there exists $p_i \in L$, such that the transmissions of p_i and p_n intersect. By definition, the environment can add the event *TransmissionError*, to p_n error channel, e_n and let the event go unnoticed with respect to p_i error and communication channels. Thus, Property 2 holds. The lemma is proved since $c[x+1]$ follows the definition of $c[x]$. We note that the same proof holds when each leaf node, $p_i \in L$, has a set of neighbors, $\delta_i \setminus \{p_\delta\}$, that do not include p_δ. The clock of each such neighbor, $p_j \in \delta_i$, can synchronize with the one of p_i and its timeslot, $s_j \neq s_i$ can allow communication with p_i once in every TDMA frame. □ □

The proof of Lemma 1 considers that case of $\tau \leq (2-\varepsilon)\delta$ and the star topology graph. We note that the same scenario can be demonstrated in every graph that includes a node with the degree δ. Thus, we can establish a general proof for $\tau < \max((2-\varepsilon)\delta, \chi(G^2))$ using the arguments in the proof of Lemma 2, where χ_2 is the chromatic number when considering distance-2 coloring.

Lemma 2. *Let $\xi \in \mathbb{R}, \tau \in \mathbb{N}$ and $S := \{[a\xi,(a+1)\xi) \colon a \in [0,\tau-1]\}$ be a partition of $[0,\xi\tau)$. The intervals $C := \{[b_i, b_i+\xi) \colon b_i \in \mathbb{R}\}_i$ intersects maximum $2|C|$ elements of S.*

Proof. Suppose that $[b,b+\xi) \in C$ intersects $I := [a\xi,(a+1)\xi) \in S$ for some a. Either $I = [b,b+\xi)$, $b \in I$ or $b+\xi \in I$. Therefore, any element $[b_i,b_i+\xi)$ of C intersects maximum 2 elements of S, one that contains b_i and one that contains $b_i+\xi$. □ □

4 Self-stabilizing TDMA Allocation and Alignment Algorithm

We propose Algorithm 1 as a self-stabilizing algorithm for the $\mathscr{T}_{\text{TDMA}}$ 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 frame information that includes 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. Passive nodes, p_i, can become active by uniformly, at random, selecting timeslots, s_i, that are not used by active nodes, sending 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. For example, when a data packet transmission fails, p_i, concludes that its data packet collided.

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 $\mathscr{O}(\tau)$, and respectively, $\mathscr{O}(1)$ 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

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

```
$status_i$: current node status in $\{\mathsf{active}, \mathsf{passive}\}$
$s_i$: current data packet timeslot in $[0, \tau - 1]$
$lastTx_i$: type of the last transmission in $\{passiveCSMA, activeCSMA, activeTDMA\}$
$wait_i, waitAdd_i$: current back off countdown in $[0, maxWait]$
$FI_i := \{id_k, type_k, occurrence_k, rxTime_k\}_k \subset \mathscr{FI}$: frame information
$timeOut$: age limit of an element in a frame information set
$BackOff() :=$ let $(tmp, r) \leftarrow (waitAdd_i, random([1, 3\Delta]))$; return $(r + tmp, 3\Delta - r)$
$frame() :=$ the current frame number $(GetClock() \div \xi\tau) \bmod \tau$,
$Slot(t) := (t \div \xi \bmod \tau), s() := Slot(GetClock())$ convert time to slot numbers
$Local(set) := \{\langle \bullet, \mathsf{local}, \bullet \rangle \in set\}$ $Ids(set) := \{id \in \mathsf{ID} : \langle id, \bullet \rangle \in set\}$
$Used(set) := \bigcup_{\langle \bullet, t_k \rangle \in set} [Slot(t_k), Slot(t_k + \xi - 1)]$; $Unused(set) := [0, \tau - 1] \setminus Used(set)$
$ConflictWithNeighbors(set) := (id_i \notin Ids(set) \vee s_i \in [Slot(t_i), Slot(t_i + \xi)] \vee$
$\exists_{\langle k, \bullet, rxTime \rangle \in set, k \neq id_i} : s_i \in [Slot(rxTime - t_j + t_i), Slot(rxTime - t_j + t_i + \xi)])$
$AddToFI(set, offset) := FI_i \leftarrow FI_i \cup \{\langle x, y, \mathsf{remote}, (z + \max\{0, offset\}) \bmod c \rangle : \langle x, y, \bullet, z \rangle \in set\}$
$IsUnused(s) := s \in Unused(FI_i) \vee (Unused(FI_i) = \emptyset \wedge s \in Unused(Local(FI_i)))$

1  upon $timeslot()$ do
2      if $s() = s_i \wedge status_i = \mathsf{active}$ then
3          transmit$(\langle status_i, Local(FI_i), MAC\_fetch() \rangle)$; $lastTx_i \leftarrow activeTDMA$;
4      else if $\neg(status_i = \mathsf{active} \wedge frame() \neq s_i)$ then
5          if $IsUnused(s()) \wedge wait_i \leq 0$ then
6              transmit$(\langle status_i, Local(FI_i), 0 \rangle)$; $\langle wait_i, waitAdd_i \rangle \leftarrow BackOff()$;
7              if $status_i = \mathsf{active}$ then $lastTx_i \leftarrow activeCSMA$;
8              else $\langle s_i, status_i, lastTx_i \rangle \leftarrow \langle s(), \mathsf{active}, passiveCSMA \rangle$
9          else if $wait_i > 0 \wedge IsUnused((s() - 1) \bmod \tau)$ then $wait_i \leftarrow \max\{0, wait_i - 1\}$
10     $FI_i \leftarrow \{\langle \bullet, rxTime \rangle \in FI_i : GetClock() < (timeOut + rxTime) \bmod c\}$;

11 upon $TransmissionError()$ do
12     if $lastTx_i \neq activeCSMA$ then $\langle \langle wait_i, waitAdd_i \rangle, status_i \rangle \leftarrow \langle BackOff(), \mathsf{passive} \rangle$

13 upon $\langle j, t_j, t_i, \langle status_j, FI_j, m' \rangle \rangle \leftarrow$ receive() do
14     if $ConflictWithNeighbors(FI_j) \wedge status_i = \mathsf{active}$ then
           $\langle \langle wait_i, waitAdd_i \rangle, status \rangle \leftarrow \langle BackOff(), \mathsf{passive} \rangle$;
15     if $status_j = \mathsf{active}$ then
16         if $m' \neq \perp$ then $FI_i \leftarrow \{\langle id_i, \bullet \rangle \in FI_i : id_i \neq j\} \cup \{\langle j, \mathsf{message}, \mathsf{local}, t_i \rangle\}$
17     else if $t_j = t_i \wedge Slot(t_j) \notin Used(FI_i)$ then
18         $FI_i \leftarrow \{\langle id_i, \bullet \rangle \in FI_i : id_i \neq j\} \cup \{\langle j, \mathsf{welcome}, \mathsf{local}, t_i \rangle\}$;
19     if $t_i < t_j$ then
20         $AdvanceClock(t_j - t_i)$; $FI_i \leftarrow \{\langle \bullet, (rxTime + t_j - t_i) \bmod c \rangle : \langle \bullet, rxTime \rangle \in FI_i\}$;
21         if $s_i \in Used(Local(FI_i))$ then
               $\langle \langle wait_i, waitAdd_i \rangle, status_i \rangle \leftarrow \langle BackOff(), \mathsf{passive} \rangle$
22     $AddToFI(FI_j, t_i - t_j)$;
23     if $m' \neq \perp$ then $MAC\_deliver(m')$
```

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. We use $lastTx_i$ for distinguishing between data and control packet transmissions.

The frame information is the set $FI_i := \{id_k, type_k, occurrence_k, rxTime_k\}_k \subset \mathscr{F}\mathscr{I} = \mathsf{ID} \times \{\mathsf{message}, \mathsf{welcome}\} \times \{\mathsf{remote}, \mathsf{local}\} \times \mathbb{N}$ that contains information about recently received packets, where $\mathsf{ID} := \{\perp\} \cup \mathbb{N}$ 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 $type_k = \mathsf{message}$ indicates that the sender was active. For a passive sender $type_k = \mathsf{welcome}$ indicates that there was no known conflict when this element was added to the local frame information. If $occurrence_k = \mathsf{local}$, 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 $waitAdd_i \leftarrow 3\Delta - r$, 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. $Local(set)$, to retrieve its ids, cf. $Ids(set)$, 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 3), 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 $\tau > 4\delta$. For the former case, Lemma 4 shows that there is always an unused timeslot s' that is not used by any neighbor $p_j \in \Delta_i$, whereas for the latter case, Lemma 5 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 three events: (1) periodic timeslots (line 1), (2) transmission errors (line 11) and (3) reception of a packet (line 13).
(1) $timeslot()$, *line 1:* Actives nodes transmit their data packets upon their timeslot and update their $lastTx_i$ (line 3). Passive nodes transmit control packets when the back off counter, $wait_i$, reaches zero (line 6) and updates $lastTx_i$ afterwards. Note that $\mathsf{passive}$ nodes count only when the local frame information says that the previous timeslot was unused (line 9). 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 4).
(2) $TransmissionError()$, *line11:* Transmission errors indicate failure of the previous attempt to transmit, i.e., due to concurrent transmissions or transient faults. Active nodes become passive when they learn whose data packets collide (line 12). Passive nodes always strive to become active by sending control packets during timeslots that they view as unused. This implies that, during convergence, control packets can collide with other control packets or even with data packets. We note that passive do not become active when they receive transmission errors for their control packets. Moreover, the correctness proof shows that no collision occurs during a legal execution.
(3) ***receive***$()$, *line 13:* 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 14). When the sender is active, the receiver records the related frame information. Note that the payload of data packets is not empty in line 16, 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 $Slot(t) \notin Used(FI_i)$, 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 [12]. When the sender clock value is higher (line 19), the receiver adjusts its clock value and the timestamps in the frame information set, before validating its timeslot, s_i, (lines 20 to 21). The receiver can now use the sender's frame information and payload (lines 22 to 23).
Correctness The proof of Theorem 1 starts by showing that p_i's active neighbors in δ_i stop interfering with each other's communications (Lemma 3). Then the proof shows the existence of unused timeslots by considering the cases in which $\tau > 2\Delta$ and $\tau > 4\delta$ (Lemmas 4, and respectively, 5). This facilitates the proof of network connectivity (Lemma 6), clock synchronization (Theorem 2) and bandwidth allocation (Theorem 3).

Theorem 1. *Algorithm 1 is a self-stabilizing implementation of task $\mathscr{T}_{\text{TDMA}}$ that converges within $\mathscr{O}(\mathsf{diam} \cdot \tau^2 + \tau^3\delta)$ 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 $\mathscr{O}(\tau^3)$, and $\mathscr{O}(\tau^3\delta)$ steps when $\tau > 2\Delta$, and respectively, $\tau > 4\delta$.*

We bound the duration in which direct and active neighbors interfere with each other's communications (Lemma 3). The proof argues that nodes, $p_i \in P$, can invoke $TransmissionError()$ within a bounded time and set $status_i \leftarrow$ passive. The proof considers the following definitions. Given a configuration c, we denote by $\mathscr{A}(c) = \{p_i \in P : status_i = \mathsf{active} \wedge wait_i = 0\}$ for the set of active nodes, and by $\mathscr{P}(c) = P \setminus \mathscr{A}(c)$ the set of passive ones. Let R be an execution, $a_i[x]$ and $a_j[x']$ two steps that include the $transmit_i(m)$, and respectively, $transmit_j(m')$ operations, and $a_k[x'']$ is a step that includes either the $obmission_k(m)$ or $receive(m)$ operations, where $p_i, p_j, p_k \in P$ and $x \leq x' < x''$. We note that $a_i[x]$ and $a_j[x']$ include concurrent transmissions. A *frame* for a node $p_i \in P$ is the time between two successive events of $timeSlot(0)$. We say that the step sequence $(a_i[x_\ell])_\ell : p_i \in P$ has ℓ *successive transmissions* if $a_i[x_k]$ includes a transmission operation, and each of the pairs $a_i[x_k], a_i[x_{k+1}] \in (a_i[x_\ell])_\ell$ occurs in successive frames with respect to p_i's clock. We say that $(a_i[x_\ell])_\ell$ and $(a_j[x'_\ell])_\ell$ are *sequences of pairwise concurrent successive transmissions* of $p_i \in P$ and, respectively, $p_j \in P$ when $(a_i[x_\ell])_\ell$ and $(a_j[x'_\ell])_\ell$ are successive transmission sequences, and for all l, the step $a_i[x_\ell]$ includes a transmission operation that is concurrent to the one included in $a_j[x'_\ell]$.

Lemma 3. *Let R be an execution of Algorithm 1 that starts from an arbitrary configuration $c[x]$. Let $p_i \in \mathscr{A}(c[x])$ be an active node and $p_j \in \delta_i \cap \mathscr{A}(c[x])$ be p_i's neighbor. Let $(a_i[x_\ell])_\ell$ and $(a_j[x'_\ell])_\ell$ maximal sequences of pairwise concurrent successive transmission of p_i and, respectively, p_j. We have that $|((a_i[x_\ell])_\ell| \leq \alpha$.*

Proof. Suppose, by the way of a proof by contradiction, that R includes sequences, $(a_i[x_\ell])_\ell$ and $(a_j[x'_\ell])_\ell$, of pairwise concurrent successive transmissions that are longer than $\alpha < |(a_i[x_\ell])_\ell|$. We show that within $|(a_i[x_\ell])_\ell|$ pairwise concurrent successive transmissions, either p_i, p_j, or both become passive; $\mathscr{P}(c[x_{l'}]) : l' \leq \alpha$. Thus, a contradiction since the sequence is interrupted. Note that by the definition of sequences of pairwise concurrent successive transmission, nodes p_i and p_j, concurrently transmit in successive frames. By Property 2, within α such frames, $p_k : k \in \{i, j\}$ execute a step, a_k, that invokes $TransmissionError_k()$ and by line 12 holds $status_k = \mathsf{passive}$ in the configuration that immediately follows a_k. So, the transmit operation in line 3 is not invoked until the node is reactivated. Hence, the lemma. □ □

Communication among neighbors is possible only when there are timeslots that are free from transmissions in the local neighborhood. Lemma 4 assumes that $\tau > 2\Delta$ and shows that every node, $p_i \in P$, has an unused timeslot, s, with respect to p_i's clock, that satisfies the conditions of Property 1 with respect to *all* of p_i's neighbors $p_j \in \delta_i$. The proof considers the definitions of the start and the end of frames and timeslots, as well as unused timeslots. A configuration, $c_i^{FrameStart}[x_\ell] = c[x]$, in which $GetClock_i() \bmod (\xi\tau) = 0$ holds, marks the start of one of p_i's frames. This frame ends when the next frame starts, i.e., the next configuration $c_i^{FrameStart}[x_{\ell+1}]$. A timeslot of p_i is, respectively, bounded by two successive configurations $c_i^{TimeSlot}[x_\ell]$ and $c_i^{TimeSlot}[x_{\ell+1}]$, such that in those configurations $GetClock_i() \bmod \xi = 0$ holds. The slot number for this timeslot is given as $GetClock_i() \div \xi \bmod \tau$ at configuration $c_i^{TimeSlot}[x_\ell]$. Given execution R, we denote a timeslot starting at $c_i^{TimeSlot}[x_\ell]$ as unused if there is no active node $p_j \in \Delta_i$ exists such that it has in R an intersecting data packet timeslot. Namely, there is no configuration $c_j^{TimeSlot}[x'_\ell]$ in R with slot number $GetClock_i() \div \xi \bmod \tau = s_j$ occurs before $c_i^{TimeSlot}[x_{\ell+1}]$ and a configuration $c_j^{TimeSlot}[x'_{\ell+1}]$ occurs after $c_i^{TimeSlot}[x_\ell]$.

Lemma 4. *Suppose that $\tau > 2\Delta$ and $p_i \in P$. Let R be an execution of Algorithm 1 that includes a complete frame start with respect to p_i's clock. Between any two successive frame starts, $c_i^{FrameStart}[x_\ell]$ and $c_i^{FrameStart}[x_{\ell+1}]$, there is at least one unused timeslot.*

Proof. Let us consider all the configurations, c', that are between $c_i^{FrameStart}[x_\ell]$, and $c_i^{FrameStart}[x_{\ell+1}]$. Let S be the partition of p_i's frame in τ timeslots of length ξ and C be the maximal set of data packet timeslots of active nodes $p_j \in \Delta_i$ (and their respective clocks, C_j). We show that $\tau > 2\Delta$ implies the existence of at least one unused timeslot between $c_i^{FrameStart}[x_\ell]$ and $c_i^{FrameStart}[x_{\ell+1}]$, by requiring that C's elements cannot interest all τ elements in S. The nodes p_j periodically transmit a data packet once every τ timeslots (line 2). Note that there are at most Δ active nodes, p_j, in all (possibly arbitrary) configurations c'. Namely, every p_j has a single data packet timeslot, s_j, but s_j's timing is arbitrary with respect to p_i's clock. By the proof of Lemma 2, C interests maximum $2|C|$ elements of the set S. Since $|S| = \tau$, $|C| \leq \Delta$, and the assumption that C's elements cannot interest all elements in S, we have $\tau > 2\Delta$ implies the existence of at least one unused timeslot between $c_i^{FrameStart}[x_\ell]$ and $c_i^{FrameStart}[x_{\ell+1}]$. □ □

Lemma 5 extends Lemma 4 by assuming that $\tau > 4\delta$ and showing that every node, $p_i \in P$, has an unused timeslot, s, with respect to p_i's clock, that 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 $p_j \in \delta_i$, as in the proof of Lemma 4.

Lemma 5. *Suppose $\tau > 4\delta$, $p_i \in P$ and $p_j \in \delta_i$. Let R be an execution of Algorithm 1 that includes a complete frame with respect to p_i's clock. With respect to p_i's clock, between any two successive frame starts, $c_i^{FrameStart}[x_\ell]$ and $c_i^{FrameStart}[x_{\ell+1}]$, there is at least one timeslot that is unused by any of the nodes $p_k \in \delta_i \cup \delta_j \cup \{p_j, p_i\}$.*

Proof. Let C be the maximum set of data packet timeslots of active nodes $p_j \in \delta_i \cap \delta_j \cap \{p_j\}$ (and their respective clocks, C_j). The proof follows by arguments similar to those of Lemma 4. We show that $\tau > 4\delta$ implies the existence of at least one timeslot that is unused by any of the nodes $p_k \in \delta_i \cup \delta_j \cup \{p_j, p_i\}$ between $c_i^{FrameStart}[x_\ell]$ and $c_i^{FrameStart}[x_{\ell+1}]$, by requiring that C's elements cannot interest all τ elements in S, c.f., proof of Lemma 4 for S's definition. By the proof of Lemma 2, C interests maximum $2|C|$ elements of the set S. Since $|S| = \tau > 4\delta$, $|C| \leq 2\delta$, and the assumption that C's elements cannot interest all elements in S, we have $\tau > 4\delta$ implies the existence of at least one unused timeslot between $c_i^{FrameStart}[x_\ell]$ and $c_i^{FrameStart}[x_{\ell+1}]$. □ □

Lemma 6 shows that the control packet exchange provides network connectivity. Recall that lemmas 4 and 5 imply 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. Lemmas 4 and 5 refer to the cases when $\tau > 2\Delta$ and $\tau > 4\delta$ for which Lemma 6 shows that the communication delay (during convergence) of the former case is δ times shorter than the latter. The proof shows that we can apply the analysis of [13], 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 $\ell := (1 - e^{-1})^{-1}$.

Lemma 6. *Let R be an execution of Algorithm 1 that starts from an arbitrary configuration $c[x]$. Let R' be a suffix of R that starts from a configuration $c[x + \mathcal{O}(timeOut)]$. Every expected $E(\beta)$ frames in R', node $p_i \in P$ receives at least one message from all direct* passive *neighbors, $p_j \in \delta_i$, where $E(\beta)$ is $(3/2)\ell$ frames if $2\Delta < \tau$, and $(3/2)\ell\delta$ frames if $\tau > 4\delta$. Moreover, every expected $E(\gamma)$ frames, p_i receives at least one message from all* active *neighbor, $p_k \in \delta_i$, where $E(\gamma)$ is $(3/2)\tau\ell$ frames if $2\Delta < \tau$, or $(3/2)\tau\ell\delta$ frames if $\tau > 4\delta$.*

Proof. Starting from an arbitrary configuration $c[x]$, within $\mathcal{O}(timeOut)$ steps a configuration $c[x+\mathcal{O}(timeOut)]$ is reached, such that in any frame information set, FI_i from any node $p_i \in P$, contains only elements added by an actual received packet. Lemmas 4 and 5, together with Property 1, are proving that there is for every $p_k \in \delta_i$ at least one timeslot such that p_k can use it to send a packet to p_i. Assume node p_k is passive. Node p_k counts down a random number of unused timeslots regarding FI_k. Since FI_k does not necessarily contain information about conflicting neighbors, i.e., $p_\ell, p_m \in \delta_k \cap \mathcal{A}$ whose data packet timeslots are intersecting, p_k could use a slot which is used by some neighbors. Suppose that some nodes in $\delta_k \cap \mathcal{A}$ are not in FI_k due to conflicts. Then they intersect maximum $2/3$ of the unused timeslots in FI_k, since two conflicting nodes can only intersect with three timeslots, but for each node we add two slots to our frame ($\tau > 2\Delta$). Therefore, a factor of $3/2$ is added to our expected value ℓ. The same holds if p_k is active. In the case of $\tau \in [4\delta+1, 2\Delta]$, there are maximum δ different timeslots for transmitting packets to all neighbors; one for each $p_k \in \delta_i$. The choice of a timeslot is random and, thus, there is an additional factor of δ to hit this timeslot. □ □

We borrow the proof of a self-stabilizing clock synchronization algorithm that is based on the converge-to-the-max principle [12].

Theorem 2. *Let R be an execution of Algorithm 1 that starts from an arbitrary configuration $c[x]$. Within expected $E(\Phi)$ frames, a configuration $c[x^{synchro}]$ is reached after which all clocks are synchronized, where $E(\Phi)$ is $\mathsf{diam}(3/2)\tau\ell$ when $2\Delta < \tau$, and $\mathsf{diam}(3/2)\tau\ell\delta$ when $\tau > 4\delta$.*

Proof Sketch. The correctness proof of the algorithm in [12] considers the set of nodes, $M \subseteq P$, that have the maximum clock value. The proof in [12] assumes that if node p_i transmits message m periodically, once every ϕ time units, then within Φ time units, $p_j \in \delta_i$ receives m. We argue that for the case, our clock model, which has zero clock skew, we can consider the expected time, $E(\Phi)$, for p_j to receive m, rather than the constant Φ. When the clock skews are zero, the assumption in [12] is required for arguing that the maximum clock values propagates among neighbors in a timely manner, rather than arguing that the clock offset among neighbors does not grow larger than a certain bound, as it is required when the clock skews are not zero. This facilitates the proof in [12], which argues by induction on M, that within a period of $\mathsf{diam}\cdot(\Phi+\phi)$, we have that M becomes P. Therefore, the proof of this theorem can substitute $(\Phi+\phi)$ with $E(\Phi)$. The proof of the clock synchronization algorithm in [12] consider warp around of the clock values, see Lemma 7 in [12]. In the first case, all clock values, C_i, are smaller than the maximal clock value, $c-1$, by at least the algorithm convergence time, i.e., $\forall_{p_i} : C_i \in [0, c-1-E(\Phi)\tau\xi]$. The proof of this case follows that arguments above. The second case, $\forall_{p_i} : C_i \in [0, E(\Phi)\tau\xi - 1] \vee C_i[c-E(\Phi)\tau\xi, c-1]$, is that all clocks are near wrapping around. Clocks can change from the lower interval to the higher one, but after expected $E(\Phi)\tau\xi$ rounds, all clocks have wrapped around, and reached the lower interval $[0, E(\Phi)\tau\xi - 1]$, or have left the lower interval by counting up normally, but not by calling $AdvanceClock()$. Thus, the proof of the second case is followed by the arguments of the first case. The third case supposes that there are nodes in configuration $c[x]$ whose clock values are in the range $[0+E(\Phi)\tau\xi], c-1-E(\Phi)\tau\xi]$, and at least one with a clock in $[c-E(\Phi)\tau\xi, c-1]$.

When there is no node in $[c-2E(\Phi)\tau\xi, c-1-E(\Phi)\tau\xi]$, it takes $E(\Phi)\tau\xi$ steps from $c[x]$ until configuration $c[x']$ is reached. Between $c[x]$ and $c[x']$, all large clocks, $C \in [c-E(\Phi)\tau\xi, c-1]$, and those who adjusted to this value, have wrapped around. Now no clock is in $[c-E(\Phi)\tau\xi, c-1]$, and the first case applies. Suppose that immediately after $c[x']$, there is a node, p_j, with clock $C_j \in [c-2E(\Phi)\tau\xi, c-1-E(\Phi)\tau\xi]$, such that p_j does not adjust its clock to a large value. Then after $E(\Phi)\tau\xi$ steps from $c[x']$, a configuration $c[x'']$ is reached in which each node in P is expected to: (1) wrapped around a large clock, (2) have a large clock, or (3) have adjusted to p_j's clock. However, in $c[x'']$ it holds that p_j's clock gets large, and the rest of the proof follows by the arguments of the second case. ☐

Once the clocks are synchronized the TDMA timeslots are aligned, as well. Thus, the number of timeslots that conflicting nodes can block is $4/3$, as Lemma 7 shows, rather than $3/2$, as Lemma 6 showed (for the case of arbitrary clock offsets).

Lemma 7. *Let R be an execution of Algorithm 1 that starts from a configuration, $c[x^{synchro}]$, in which all clock are synchronized. Every expected $E(\beta')$ frames in R, node $p_i \in P$ receives at least one message from all direct* passive *neighbors, $p_j \in \delta_i$, where $E(\beta')$ is $(4/3)\ell$ frames if $2\Delta < \tau$, and $(4/3)\ell\delta$ frames if $\tau > 4\delta$. Moreover, every expected $E(\gamma')$ frames, p_i receives at least one message from all* active *neighbor, $p_k \in \delta_i$, where $E(\gamma')$ is $(4/3)\tau\ell$ frames if $2\Delta < \tau$, or $(4/3)\tau\ell\delta$ frames if $\tau > 4\delta$.*

Proof Sketch. Let $p_i \in P$ and partition the set $\Delta_i = A_i \uplus I_i \uplus P_i =: F$, where A_i is the set of active neighbors of p_i having a unique (from p_i's point of view) data packet time slot and I_i is the subset of active neighbors that interfere with each other, i.e., $\forall_{p_\ell \in I_i} \exists_{p_k \in I_i}$: p_ℓ's timeslots intersects p_k's. Note that p_ℓ and p_k using the same timeslot in this case and thus I_i intersects maximum $|I_i|/2$ timeslots. P_i is the set of passive neighbors of p_i. Then the data packet time slots of the nodes A_i are contained in FI_i. Maximum $2\Delta - |A_i| = 2(|I_i| + |P_i|) + |A_i|$ and minimum $2\Delta - |A_i| - |I_i|/2 = 1/2|I_i| + 2|P_i| + |A_i|$. Thus, maximum $1/2|I_i|$ out of $2|I_i|$ timeslots are unused regarding FI_i. Moreover, a random choice of a timeslot is successful with probability $3/4$. Therefore, a factor of $4/3$ to the expected communication delay. The same arguments hold for $\tau > 4\delta$. ☐

Once the clocks are synchronized, and the TDMA timeslots are aligned, Algorithm 1 allocates the bandwidth using distance-2 coloring. This happens within $\mathcal{O}(\tau^3)$, and $\mathcal{O}(\tau^3\delta)$ steps when $\tau > 2\Delta$, and respectively, $\tau > \max\{4\delta, \Delta+1)$, see Theorem 3.

Theorem 3. *Let R be an execution that starts from an arbitrary configuration $c[x^{synchro}]$ in which all clocks are synchronized. Within expected $E(\zeta)$ frames from $c[x^{synchro}]$, the system reaches a configuration, $c[x^{alloc}]$, in which each node $p_i \in P_j$ has a timeslot that is unique in Δ_i, where $E(\zeta)$ is $(4/3)\tau\ell\max\{\alpha, (4/3)\tau\ell\}$ when $2\Delta < \tau$, or $(4/3)\tau\ell\max\{\alpha, (4/3)\tau\ell\}\delta$ when $\tau > 4\delta$.*

Proof Sketch. We have showed that active nodes get feedback within an expected time. Therefore, also conflicting active nodes. 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 $\max\{\alpha,(4/3)\tau\ell\}$ if $2\Delta \leq \tau$, or $\max\{\alpha,(4/3)\tau\ell\}\delta$ if $\tau > 4\delta$, 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 $(4/3)\tau\ell\max\{\alpha,(4/3)\tau\ell\}$, where the first factor $(4/3)\tau\ell$ is introduced by the delay for a negative feedback. □

The proof of Theorem 1 is concluded by showing that configuration, $c[x^{alloc}]$ (Theorem 3), is a safe configuration with respect to LE_{TDMA}, see Lemma 8.

Lemma 8. *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 Sketch. Obviously, the properties of legal executions LE_{TDMA} are fulfilled for $c[x]$. Since the clocks are equal no node transmits a packet that is leading to clock adjustment of another node. Furthermore, the timeslot and frame numbers are synchronized. There are no collisions, since every node is transmitting at a unique timeslot. □

5 Conclusions

This work considers fault-tolerant systems that have basic radio and clock settings without access to external reference 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. 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 $\tau/\delta \leq 4$. Unfortunately, our result implies that, for our systems settings, there is no distributed mechanism for asserting the convergence criteria within a constant time. For distributed systems that do *not* require constant communication delay, we propose to explore such criteria assertion mechanisms as future work.

Acknowledgments This work would not have been possible without the contribution of Marina Papatriantafilou in many helpful discussions, ideas, problem definition and analysis.

References

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

[2] N. Alon and B. Mohar. The chromatic number of graph powers. *Combinatorics, Probability & Computing*, 11(1):1–10, 2002.

[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. *Distributed Computing*, 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] T. Herman. Models of self-stabilization and sensor networks. In S. R. Das and S. K. Das, editors, *IWDC*, volume 2918 of *Lecture Notes in Computer Science*, pages 205–214. Springer, 2003.

[11] T. Herman and S. Tixeuil. A distributed TDMA slot assignment algorithm for wireless sensor networks. In *ALGOSENSORS*, volume 3121 of *Lecture Notes in Computer Science*, pages 45–58. Springer, 2004.

[12] T. Herman and C. Zhang. Best paper: Stabilizing clock synchronization for wireless sensor networks. In A. K. Datta and M. Gradinariu, editors, *SSS*, volume 4280 of *Lecture Notes in Computer Science*, pages 335–349. Springer, 2006.

[13] 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.

[14] A. Jhumka and S. S. Kulkarni. On the design of mobility-tolerant TDMA-based media access control (MAC) protocol for mobile sensor networks. In T. Janowski and H. Mohanty, editors, *ICDCIT*, volume 4882 of *Lecture Notes in Computer Science*, pages 42–53. Springer, 2007.

[15] J. Johnson and B. Dewberry. Ultra-wideband aiding of gps for quick deployment of anchors in a gps-denied ad-hoc sensor tracking and communication system. In *Proceedings of the 24th International Technical Meeting of The Satellite Division of the Institute of Navigation (ION GNSS 2011)*, pages 3959–3966, Oregon Convention Center, Portland, Oregon, September 2011.

[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. Analyzing protocols for media access control in large-scale mobile ad hoc networks. In *Workshop on Self-Organising Wireless Sensor and Comm. Net. (Somsed)*, 2009.

[19] P. Leone, M. Papatriantafilou, E. M. Schiller, and G. Zhu. Chameleon-MAC: adaptive and self-$\star$ 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.

[20] P. Leone and E. M. Schiller. Self-stabilizing TDMA algorithms for dynamic wireless ad-hoc networks. *To appear in International Journal of Distributed Sensor Networks and also avalable in CoRR*, abs/1210.3061, 2012.

[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] 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.

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

[27] 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 *Lecture Notes in Computer Science*, pages 74–89. Springer, 2006.

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

[29] 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.