

Self-stabilizing TDMA Algorithms for Wireless Ad-hoc Networks without External Reference ^{*}

Thomas Petig [†] Elad M. Schiller[†]
 petig@chalmers.se 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 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.

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

^{*}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 brief announcement [24].

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

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 a bound on the 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* [8] 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 used when the application requires bounded communication delays, e.g., IEEE 802.15.4 and deterministic self-stabilizing TDMA [2, 14]. Without such external references, the TDMA algorithms have to align the timeslots while allocating them. Existing algo-

gorithms [4] 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 use 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 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 Herman and Zhang [11] assume constant bounds on the communication delay and present self-stabilizing clock synchronization algorithms for wireless ad hoc networks. Herman and Tixeuil [10] 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. [11], but on the other hand, existing MAC algorithms that provide bounded communication delay, often assume access to synchronized clocks, e.g. [10]. 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 [11] 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 [11].

The proposal in [9] 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 [6], topology discovery in anonymous networks [19], random distance- k vertex coloring [20], deterministic distance-2 vertex coloring [3], 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 [15] present a MAC algorithm that uses convergence from a random starting state (inspired by self-stabilization). In [16, 22], 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 [17]. 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 [7, 13] 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 [4, 5, 7, 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 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 3). 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 [21] ($\chi_2 = 5\delta/3 + \mathcal{O}(1)$).

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 4). The convergence period is within $\mathcal{O}(\text{diam} \cdot \tau^2\delta + \tau^4\delta^2)$ 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.

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 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, $\Delta \geq \max_{p_i \in P} (|\Delta_i|)$. We assume that $\text{diam} \geq \max_{p_i, p_j \in P} d(p_i, p_j)$ 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, $c \gg \text{diam}\tau^2$, 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 $m \leftarrow \text{MAC_fetch}_i()$ from the upper layer, and before the receiver, p_j , delivers it to the upper layer in $\text{MAC_deliver}_j(m)$, they exchange m via the operations $\text{transmit}_i(m)$, and respectively, $m \leftarrow \text{receive}_j()$. 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 $\text{transmit}_i(m)$ inserts a copy of m to every $q_{i,j}$, such that $p_j \in \delta_i$. Once m arrives, p_j executes $\text{receive}()$ 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 $c := (st_1, \dots, st_{|P|})$ 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 \mathcal{T} is a set 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 3, we show that the task solution can depend on the (arbitrary) starting configuration, rather than just the communication graph.

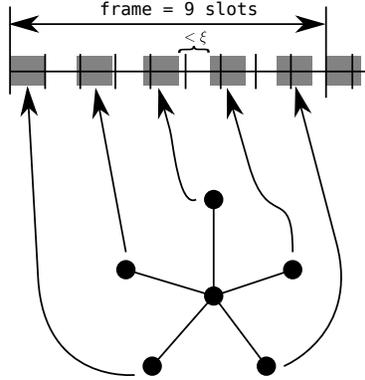


Figure 1: The outer five nodes are covering nine timeslots. The top horizontal line and its perpendicular marks depict the radio time division according to the central node, p_δ . The gray boxes depict the radio time covered by the leaf nodes, $p_i \in L$.

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 χ_2 is the chromatic number for distance-2 vertex coloring.

Lemma 1. *Let $\delta \in \mathbb{N}$ and $\tau < 2\delta$. 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 $c[x]$, such that an execution R starting from $c[x]$ of any algorithm solves the task $\mathcal{T}_{\text{TDMA}}$ does not converge.*

Proof. We prove this Lemma by showing an example in which we assign timeslots to a subset of nodes in a way, such that they block each other and, thus, disconnect the communication graph.

Let $\tau = 2\delta - 1$. Let $c[x]$ be such that: (1) C_i in $c[x]$ has the properties $(C_i + (2\xi - 1)i) \bmod \xi = 0$ and $(C_i + (2\xi - 1)i) \div \xi \bmod \tau = s_i$ for all $p_i \in L \setminus \{p_\delta\}$, (2) $s_\delta = \perp$ and (3) there is no message in transit. Figure 1 shows such a graph for $\delta = 5$. This means the next timeslot of node $p_i \in L \setminus \{p_\delta\}$ starts $(2\xi - 1)i$ clock steps after $c[x]$. The gap between the time p_i 's timeslot ends and p_{i+1} 's timeslot starts is $(2\xi - 1)(i+1) - ((2\xi - 1)i + \xi) = \xi - 1 < \xi$ clock steps long and thus smaller than a time slot. The gap between the next transmission of $p_{\delta-1}$ and the next next transmission of p_0 is $(2\delta - 1)\xi + (2\xi - 1)0 - ((2\xi - 1)(\delta - 1) + \xi) = \delta - 1 < \xi$. This pattern repeats, because only p_δ receives these messages transmitted by the leaves and p_δ does not have a time slot assigned and according to Property 1

any attempt of p_δ in transmitting can fail. Thus, no algorithm can establish communication here. \square

The proof of Lemma 1 considers that case of $\tau < 2\delta$ 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 $\tau < \max\{2\delta, \chi(G^2)\}$ 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 $S := \{[a\xi, (a+1)\xi) : a \in [0, \tau-1]\}$ be a partition of $[0, \xi\tau)$. The intervals $C := \{[b_i, b_i + \xi) : b_i \in \mathbb{R}\}_i$ 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 $b_i + \xi$. \square

4 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 1: Self-stabilizing TDMA Allocation, code for node p_i

```
statusi ∈ {active, passive}; /* current node status */
si ∈ [0, τ - 1]; /* current data packet timeslot */
waiti, waitAddi ∈ [0, maxWait]; /* current back off countdown */
FIi := {idk, typek, occurrencek, rxTimek}k ⊂ FI; /* frame information */
timeOut; /* constant, age limit of elements in FIi */
BackOff() := let (tmp, r) ← (waitAddi, random([1, 3Δ])); return (r + tmp,
3Δ - r); /* reset backoff counter */
frame() := (GetClock() ÷ ξτ) mod τ; /* the current frame number */
Slot(t) := (t ÷ ξ mod τ), s() := Slot(GetClock()); /* slot number of time t */
Local(set) := {⟨•, local, •⟩ ∈ set}; /* dist-1 neighbors in set */
Used(set) := ⋃_{⟨•, tk⟩ ∈ set} [Slot(tk), Slot(tk + ξ - 1)];
Unused(set) := [0, τ - 1] \ Used(set); /* set of (un)used slots */
ConflictWithNeighbors(set) := (∄_{⟨idi, •⟩ ∈ set} ∨ si ∈ [Slot(ti), Slot(ti + ξ)] ∨
∃_{⟨k, •, rxTime⟩ ∈ set, k ≠ idi : si ∈ [Slot(rxTime - tj + ti), Slot(rxTime - tj + ti + ξ)]);
/* check for conflicts */
AddToFI(set, o) := FIi ← FIi ∪ {⟨x, y, remote, z'⟩ : ⟨x, y, •, z⟩ ∈ set, z' :=
(z + max{0, o}) mod c, z' ≤ timeOut Ci}; /* set + FIi */
IsUnused(s) := s ∈ Unused(FIi) ∨ (Unused(FIi) = ∅ ∧ s ∈ Unused(Local(FIi)));
/* is s an unused slot? */
1 upon timeslot() do
2   if s() = si ∧ statusi = active then /* send data packet */
3     | transmit((statusi, Local(FIi), MAC_fetch()))
4   else if ¬(statusi = active ∧ frame() ≠ si) then /* check if our frame */
5     | if IsUnused(s()) ∧ waiti ≤ 0 then /* send control packet */
6       | transmit((statusi, Local(FIi), 0));
7       | ⟨waiti, waitAddi⟩ ← BackOff(); /* next control packet countdown */
8       | if statusi ≠ active then ⟨si, statusi⟩ ← ⟨s(), active⟩;
9       | else if waiti > 0 ∧ IsUnused((s() - 1) mod τ) then /* count down */
10        | waiti ← max{0, waiti - 1}
11      FIi ← {⟨•, rxTime⟩ ∈ FIi : rxTime ≤ timeOut GetClock()}; /* remove old
12      entries from FIi */
13
14 9 upon ⟨j, tj, ti, ⟨statusj, FIj, m'⟩⟩ ← receive() do
15  | if ConflictWithNeighbors(FIj) ∧ statusi = active then /* conflicts? */
16  |   | ⟨⟨waiti, waitAddi⟩, status⟩ ← ⟨BackOff(), passive⟩; /* get passive */
17  |   if statusj = active then /* active node acknowledge */
18  |   | if m' ≠ ⊥ then FIi ← {⟨idi, •⟩ ∈ FIi : idi ≠ j} ∪ {⟨j, message, local, ti⟩};
19  |   | else if tj = ti ∧ Slot(tj) ∉ Used(FIi) then /* passive node acknowledge */
20  |   |   | FIi ← {⟨idi, •⟩ ∈ FIi : idi ≠ j} ∪ {⟨j, welcome, local, ti⟩};
21  |   |   if ti < tj then /* converge-to-the-max */
22  |   |   | AdvanceClock(tj - ti); /* adjust clock */
23  |   |   | FIi ← {⟨•, (rxTime + tj - ti) mod c⟩ : ⟨•, rxTime⟩ ∈ FIi}; /* shift
24  |   |   | timestamps in FIi */
25  |   |   | ⟨⟨waiti, waitAddi⟩, statusi⟩ ← ⟨BackOff(), passive⟩; /* get passive */
26  |   |   AddToFI(FIj, ti - tj); /* Aggregate information on used timeslots */
27  |   |   if m' ≠ ⊥ then MAC_deliver(m');
```

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 $waitAdd_i \leftarrow 3\Delta - r$, 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 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 $\tau > \max\{4\delta, \Delta + 1\}$. For the former case, Lemma 3 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 4 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*: Active 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 timeslots 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).

Correctness The proof of Theorem 1 starts by showing the existence of unused timeslots by considering the cases in which $\tau > 2\Delta$ and $\tau > \max\{4\delta, \Delta + 1\}$ (Lemmas 3, and respectively, 4). This facilitates the proof of network connectivity (Lemma 5), clock synchronization (Theorem 2) and bandwidth allocation (Theorem 3).

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

The proof considers the following definitions. Given a configuration c , we denote by $\mathcal{A}(c) = \{p_i \in P : status_i = \text{active} \wedge wait_i = 0\}$ the set of active nodes, and by $\mathcal{P}(c) = P \setminus \mathcal{A}(c)$ the set of passive ones. A *frame* for a node $p_i \in P$ is the time between two successive events of $timeSlot(s)$ with $s = 0$. Note that these frames depend on C_i and, thus, might not be aligned between nodes. Communication among neighbors is possible only when there are timeslots that are free from transmissions by nodes in the local neighborhood. Lemma 3 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. This 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 3. *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' 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 $c_i^{FrameStart}[x_{\ell+1}]$. \square

Lemma 3 is basically the application of Lemma 2, were we identify the timeslots with intervals. Lemma 4 extends Lemma 3 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 $p_j \in \delta_i$, as in the proof of Lemma 3.

Lemma 4. *Suppose $\tau > \max\{4\delta, \Delta + 1\}$, $p_i \in P$ 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 3. We show that $\tau > \max\{4\delta, \Delta + 1\}$ 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 3 for S 's definition. By the proof of Lemma 2, C interests maximum $2|C|$ elements

of the set S . Since $|S| = \tau > \max\{4\delta, \Delta + 1\}$, $|C| \leq 2\delta$, and the assumption that C 's elements cannot interest all elements in S , we have $\tau > \max\{4\delta, \Delta + 1\}$ implies the existence of at least one unused timeslot between $c_i^{FrameStart}[x_\ell]$ and $c_i^{FrameStart}[x_{\ell+1}]$. \square

Lemma 5 shows that the control packet exchange provides network connectivity. Recall that Lemmas 3 and 4 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 3 and 4 refer to the cases when $\tau > 2\Delta$ and $\tau > \max\{4\delta, \Delta + 1\}$ for which Lemma 5 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 [12], 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 $\ell := (1 - e^{-1})^{-1}$.

Definition 1. Let $c \in R$ be a configuration and $p_i, p_j \in P : p_j \in \delta_i$ two neighbors. We say that the tuple $e := \langle j, \bullet, \text{local}, \text{time} \rangle \in FI_i$ is *locally steady* in c when $((C_j + \tau\xi - (\text{time} \bmod \tau\xi)) \div \xi) \div \tau = s_j$. For the case of $p_i, p_j, p_k \in P : p_k \in \delta_i \wedge p_j \in \delta_k$ we say that $e := \langle j, \bullet, \text{remote}, \text{time} \rangle \in FI_i$ is *remotely steady* in c when $((C_j + \tau\xi - (\text{time} \bmod \tau\xi)) \div \xi) \div \tau = s_j$. A tuple $e \in FI_i$ is *stale* if e is neither locally steady, nor steady.

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 5. *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 $c[x + \mathcal{O}(\text{timeOut})]$ such that a node $p_i \in \mathcal{P}$ receives a message from all nodes in δ_i within finite time.*

Proof. 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. Lemmas 3 and 4 show the existence of a free time slot 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 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 distance in time from each entry to the current time of a node C_i increases monotonically. Thus, all elements 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 time slot.

Note that a node p_i adds only elements $e := \langle id, \text{type}, \text{occurrence}, \text{rxTime} \rangle \in \mathcal{FI}$ to FI_i within the lines 11, 14 and 19. If we add a direct neighbor in line 14 and 11 $\text{rxTime} = C_i$ holds. In latter case, line 19, we copy entries $e_i := \langle k, \bullet, \text{remote}, \text{rxTime}_i \rangle$ from an entries $e_j := \langle k, \bullet, \text{remote}, \text{rxTime}_j \rangle$ direct

neighbor $p_j \in \delta_i$. But here we update the $rxTime_j$ to $rxTime_i$ according to p_i 's clock C_i and thus $C_j - rxTime_j \bmod c = C_i - rxTime \bmod c$.

Let R be a run starting from a configuration c , such that p_i has a stale entry $e := \langle j, \bullet, t \rangle \in FI_i$. There are three cases that can occur. (1) p_i does not receive a packet during the next $timeOut - t + \tau$ clock steps either from p_j directly, or from p_k that contains an entry with the id j in the frame information and with a smaller clock difference. In this case there exists a configuration c' in R such that $e \notin FI_i(c')$ and $0 < (C_i(c') - C_i(c) - timeOut + t) \bmod c < \tau$. (2) p_i receives a packet from p_j and thus the stale entry e is replaced by a stable entry. (3) p_i receives a packet from p_k in configuration c' that contains an entry $e' := \langle j, bullet, t' \rangle$, such that $(C_i(c') - t) \bmod c > (C_j(c') - t')$. In any case e' replaces e . If case e' is stable, the stale entry e is replaced by e' . Otherwise e is replaced by the stale entry e' . But note that while replacing e with e' the age, i.e., the difference to the nodes clock is maintained. This means that the time until it is removed by time out is for p_i and p_j the same.

It follows that in a configuration c'' that is more than $timeOut$ clock steps after c , there are only stale entries due to nodes in the distance-2 neighborhood that got passive recently, i.e. at most $timeOut + \xi$ clock ticks ago. Thus, the free time slot for a node p_i , that exists by Lemma 3 and 4, is also free according to FI_i . This means p_i can communicate with all neighbors in δ_i within finite time. \square

After showing the network connectivity in Lemma 5, we continue with bounding the expected communication delay in Lemma 6 i.e., how long does it take to successfully send a control packet to a neighbor.

Lemma 6. *Let R be an execution of Algorithm 1 that starts from an configuration $c[x]$ such that a node $p_i \in \mathcal{P}$ continuously receives messages from all nodes in δ_i within finite time. Then every expected β frames in R' , node $p_i \in \mathcal{P}$ receives at least one message from all direct passive neighbors, $p_j \in \delta_i$, where β is $3\Delta\ell$ frames if $2\Delta < \tau$, or $3\Delta\ell\delta$ frames if $\tau > \max\{4\delta, \Delta + 1\}$. Moreover, 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 $2\Delta < \tau$, or $3\Delta\tau\ell\delta$ frames if $\tau > \max\{4\delta, \Delta + 1\}$.*

Proof. Lemmas 3 and 4 are showing the existence of unused time slots. We showed above that the frame information set reaches a state in which it is consistent with the actual assignment of time slots. It follows that a node has the chance to choose a free time slot and exchange packets with all neighbors. We show the expect communication delay.

Let $c[x]$ be a configuration. Assume node p_k is passive in $c[x]$. 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., $p_\ell, p_m \in \delta_k \cap \mathcal{A}(c[x])$ whose data packet timeslots are intersecting, p_k could use a slot which is used by some neighbors. Suppose that some nodes in $\delta_k \cap \mathcal{A}(c[x])$ 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. \square

For Theorem 2 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 [11].

Theorem 2. *Let R be an execution of Algorithm 1 that starts from an arbitrary configuration $c[x]$. Within expected $\Phi := \gamma \cdot \text{diam}$ frames, a configuration $c[x^{\text{synchrono}}]$ is reached after which all clocks are synchronized, where γ is the expected communication delay as in Lemma 6.*

Proof. For a moment consider the case of an unbounded clock. Let $M \subset \mathcal{P}$ the set of nodes with maximum clock in an arbitrary configuration c . Then we expect every execution to reach a configuration c' within the communication delay of γ frames in which every neighbor p_i of M has received a message from a node in M . Thus, p_i converged the clock to the maximum value is part of the set of nodes with maximum clock in c' . Note that a node in M is not leaving M , since we neither assume clock skew, nor can it receive a larger clock value. It follows by induction that the set of nodes with maximum clock value is increasing monotonically and within expected $\gamma \cdot \text{diam}$ frames M covers P and thus a configuration $c[x^{\text{synchrono}}]$ is reached.

We proceed with investigating bounded clocks. As in [11] we consider three cases. 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., $\forall_{p_i} : C_i \in [0, c - 1 - \Phi]$. The proof of this case follows that arguments above for the unbounded case.

The second case, $\forall_{p_i} : C_i \in [0, \Phi - 1] \vee C_i \in [c - \Phi, c - 1]$, is that all clocks are near wrapping around. Clocks can change from the lower interval to the higher one, but after expected Φ clock steps, all clocks have wrapped around, and reached the lower interval $[0, \Phi - 1]$, 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 in configuration $c[x]$ there is at least on node whose clock value is in the range $[\Phi, c - 1 - \Phi]$, and at least one with a clock in $[c - 1 - \Phi, c - 1]$. Note that those with a clock in $[c - 1 - \Phi, c - 1]$ wrap around in maximal Φ clock steps. Let $c[x']$ be the configuration after $c[x]$, such that all nodes with a clock value in $[c - 1 - \Phi, c - 1]$ have wrapped around.

There are three cases. First, all nodes are able to receive a message from one of the nodes with a clock value in $[c - 1 - \Phi, c - 1]$, before they wrap around in configuration. Then we have the second case and we are done. If in configuration $c[x']$ not all nodes receive such a large clock value in $[c - 1 - \Phi, c - 1]$, then either there is no node in $c[x']$ that has a clock value in $[c - 1 - \Phi, c - 1]$ and we have the first case, or there is at least one node with a clock value in $[c - 1 - \Phi, c - 1]$.

Let p_i one of the nodes that have the largest clock value in $c[x']$ and let $c[x'']$ be the first configuration of R after $c[x]$ such that p_i 's clock is in $[c - 1 - \Phi, c - 1]$. Then, expected $\Phi \cdot \text{diam}$ clock steps after $c[x'']$, a configuration $c[x''']$ is reached in which all nodes in \mathcal{P} have either received a clock that is equal or larger to p_i 's clock value and adopted it. Thus, in $c[x''']$ all nodes have either a clock value in $[c - 1 - \Phi, c - 1]$, or in $[0, \Phi - 1]$, because they have wrapped around. So, this is reduces to the second case. \square

Once the clocks are synchronized, and the TDMA timeslots are aligned, Algorithm 1 allocates the bandwidth using distance-2 coloring. This happens within $\mathcal{O}(\gamma^2)$ frames, see Theorem 3.

Theorem 3. *Let R be an execution that starts from an arbitrary configuration $c[x^{\text{synchrono}}]$ in which all clocks are synchronized. Within expected γ^2 frames from $c[x^{\text{synchrono}}]$, the system reaches a configuration, $c[x^{\text{alloc}}]$, 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 . \square

The proof of Theorem 1 is concluded by showing that configuration, $c[x^{\text{alloc}}]$ (Theorem 3), is a safe configuration with respect to LE_{TDMA} , see Lemma 7.

Lemma 7. *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) $\forall_{p_i \in P} : \text{status}_i = \text{active}$, (3) $\forall_{p_i \in P} \forall_{p_j \in \Delta_i} : s_i \neq s_j$, (4) $\forall_{p_i \in P} : \forall_{p_j \in \Delta_i \cup \{p_i\}} \exists!_{\langle id, \text{message}, \bullet \rangle \in FI} : id = id_j$.*

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) is a restriction to general configurations in LE_{TDMA} that ensures that a node knows about its neighborhood.

We conclude by proving 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

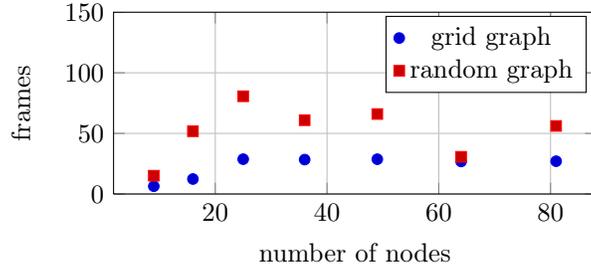


Figure 2: The converges time in frames for different graphs. In the grid graph, nodes are placed on a lattice and connected to their four neighbors. The convergence times are the average over 16 runs that start each with random clock offsets. The random node graph is a unified disk graph with random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

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 proves that $c[x]$ is a safe configuration for LE_{TDMA} . \square

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

Proof. We check that $c[x^{\text{alloc}}]$ fulfills the conditions of Lemma 7. Condition (1) follows from Theorem 2 and conditions (2), (3) and (4) are following from Theorem 3. \square

Proof of Theorem 1. Lemma 3, 4 and 5 show that communication is possible after a constant number of clock steps. Lemma 6 bounds the expected communication delay to γ . Theorem 2 shows that after $\mathcal{O}(\gamma \cdot \text{diam})$ frames a configuration $c[x^{\text{synchrono}}]$ is reached where the clocks are synchronized. Theorem 3 shows that in $\mathcal{O}(\gamma^2)$ frames after $c[x^{\text{synchrono}}]$, a configuration $c[x^{\text{alloc}}]$ is reached in which all nodes have an allocated timeslot. Corollary 1 shows that Algorithm 1 solves $\mathcal{T}_{\text{TDMA}}$ by reaching $c[x^{\text{alloc}}]$. \square

5 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 2) that we use for the correctness proof (Section 20) assumes that any (local) computation can be done in zero time. In contrast to this, the simulations use the TinyOS embedded operating systems [18] and the Cooja simulation framework [23] 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 2) 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 time slot size of $\xi = 20$ clock steps, where almost all of this period is spent on transmission, packet loading and offloading. The frame size is $\tau = 16 \geq 4\delta$ time slots for the grid graph and $\tau = 64 \geq 4\delta$ for the random graphs. For these settings, all experiments showed convergence, see Figure 2.

6 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 $\tau/\delta \geq 4$. 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 Papatrantaifilou, Olaf Landsiedel and Mohamed H. Mustafa 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] 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.
- [3] J. R. S. Blair and F. Manne. An efficient self-stabilizing distance-2 coloring algorithm. *Theor. Comput. Sci.*, 444:28–39, 2012.
- [4] C. Busch, M. Magdon-Ismael, F. Sivrikaya, and B. Yener. Contention-free MAC protocols for asynchronous wireless sensor networks. *Distrib. Comp.*, 21(1):23–42, 2008.
- [5] 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.
- [6] P. Danturi, M. Nesterenko, and S. Tixeuil. Self-stabilizing philosophers with generic conflicts. *ACM Tran. Autonomous & Adaptive Systems (TAAS)*, 4(1), 2009.
- [7] M. Demirbas and M. Hussain. A MAC layer protocol for priority-based reliable multicast in wireless ad hoc networks. In *BROADNETS*. IEEE, 2006.
- [8] S. Dolev. *Self-Stabilization*. MIT Press, 2000.
- [9] T. Herman. Models of self-stabilization and sensor networks. In S. R. Das and S. K. Das, editors, *IWDC*, volume 2918 of *LNCS*, pages 205–214. Springer, 2003.
- [10] 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.
- [11] 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 *LNCS*, pages 335–349. Springer, 2006.
- [12] 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.
- [13] 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 *LNCS*, pages 42–53. Springer, 2007.
- [14] S. S. Kulkarni and M. Arumugam. Transformations for write-all-with-collision model. *Computer Communications*, 29(2):183–199, 2006.

- [15] 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.
- [16] 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.
- [17] P. Leone and E. M. Schiller. Self-stabilizing TDMA algorithms for dynamic wireless ad-hoc networks. *Int. J. Distributed Sensor Networks*, 639761, 2013.
- [18] 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.
- [19] 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.
- [20] N. Mitton, E. Fleury, I. G. Lalous, 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.
- [21] 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.
- [22] 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.
- [23] 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.
- [24] T. Petig, E. M. Schiller, and P. Tsigas. Self-stabilizing tdma algorithms for wireless ad-hoc networks without external reference. In T. Higashino, Y. Katayama, T. Masuzawa, M. Potop-Butucaru, and M. Yamashita, editors, *SSS*, volume 8255 of *LNCS*, pages 354–356. Springer, 2013.
- [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, *IW-SOS/EuroNGI*, volume 4124 of *LNCS*, 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. (ALGOSENSORS)*, 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.