CISTER
Research Centre in
Real-Time & Embedded
Computing Systems

Conference Paper

A Clockless Synchronisation Framework for Cooperating Mobile Robots

Luis Oliveira
Luís Almeida
Daniel Mossé

CISTER-TR-181102

A Clockless Synchronisation Framework for Cooperating Mobile Robots

Luis Oliveira, Luís Almeida, Daniel Mossé

*CISTER Research Centre

Polytechnic Institute of Porto (ISEP-IPP)

Rua Dr. António Bernardino de Almeida, 431

4200-072 Porto

Portugal

Tel.: +351.22.8340509, Fax: +351.22.8321159

E-mail:

http://www.cister.isep.ipp.pt

Abstract

Cooperating mobile robots are real-time systems that often require mutual synchronisation, either to carry out cooperative sensing and actuation, or to improve the quality of wireless communications. Concerning this last aspect, a common technique to improve the communication channel is to eliminate access collisions by allocating predefined disjoint time slots to robots, in a circular list, which is known as Time Division Multiple Access (TDMA). This technique typically requires a global clock to identify each slot. However, this method is not robust with respect to asynchronous transmissions generated by external or joining nodes. Consequently, this work proposes a global TDMA protocol that allows for real-time and guaranteed delivery of messages within deadlines, given its predictable schedule, and that: i) applies to dynamic mesh networks of cooperating mobile robots; ii) synchronises slots in a relative fashion using locally perceived delays of message exchanges that are globalised throughout the network, thus not relying on a global clock; and iii) tolerates external traffic and asynchronous joining robots using underneath standard ad-hoc wireless RF technologies that provide CSMA-type arbitration. We describe our protocol and prove that under common operating conditions all robots eventually reach synchronisation. We also propose a heuristic for the few cases that were not covered by the previous proof, which always led to consensus under extensive simulation testing. To the best of our knowledge, this is the first guaranteed clockless synchronisation approach for ad-hoc networks of mobile robots that works over commodity wireless protocols.

A Clockless Synchronisation Framework for Cooperating Mobile Robots

Luís Oliveira
Computer Science Dep.,
University of Pittsburgh, USA
Email: loliveira@pitt.edu

Luís Almeida
CISTER - Cen. de Inv. em Sistemas de Tempo-Real,
IT - Instituto de Telecomunicações
FEUP - Universidade do Porto, Portugal
Email: lda@fe.up.pt

Daniel Mossé
Computer Science Dep.,
University of Pittsburgh, USA
Email: mosse@cs.pitt.edu

Abstract—**Cooperating mobile robots are real-time systems that often require mutual synchronisation, either to carry out cooperative sensing and actuation, or to improve the quality of wireless communications. Concerning this last aspect, a common technique to improve the communication channel is to eliminate access collisions by allocating predefined disjoint time slots to robots, in a circular list, which is known as Time Division Multiple Access (TDMA). This technique typically requires a global clock to identify each slot. However, this method is not robust with respect to asynchronous transmissions generated by external or joining nodes. Consequently, this work proposes a global TDMA protocol that allows for real-time and guaranteed delivery of messages within deadlines, given its predictable schedule, and that: i) applies to dynamic mesh networks of cooperating mobile robots; ii) synchronises slots in a relative fashion using locally perceived delays of message exchanges that are globalised throughout the network, thus not relying on a global clock; and iii) tolerates external traffic and asynchronous joining robots using underneath standard ad-hoc wireless RF technologies that provide CSMA-type arbitration. We describe our protocol and prove that under common operating conditions all robots eventually reach synchronisation. We also propose a heuristic for the few cases that were not covered by the previous proof, which always led to consensus under extensive simulation testing. To the best of our knowledge, this is the first guaranteed clockless synchronisation approach for ad-hoc networks of mobile robots that works over commodity wireless protocols.**

I. Introduction

Teams of mobile robots, or mobile agents in general, are frequently used in cooperative applications such as surveillance, search and rescue, automatic warehouses and industrial manufacturing. These systems fall in a class recently called Mobile Cyber-Physical Systems (M-CPS) [4] in which sensors and actuators can be relocated dynamically within a region and are inherently real-time systems.

In these systems, mobility and physical distances lead to the common use of Radio Frequency (RF) wireless communications. Since the wireless medium is shared and current standard protocols are half duplex, simultaneous transmissions may cause collisions and packet drops. With few robots and low bandwidth requirements, common arbitration-based access control protocols, such as those based on Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA), can generally deal with collisions. However, the efficiency of these protocols is seriously reduced when the number of agents and the communication load increases, leading to more collisions, packet losses, and long delays [15]. Moreover, these systems frequently use periodic communication patterns that may lead to intervals of time, called *critical interference periods* [4], during which the probability of collisions increases significantly, independently of the communication load, due to persistent simultaneous communication attempts and retransmissions [12].

Both critical interference periods and excess of collisions in general can be mitigated using synchronisation, since it allows for each agent to transmit in disjoint time slots, a technique commonly called Time Division Multiple Access (TDMA), which is typically used in real-time communications given its predictable timing characteristics. Moreover, synchronisation can benefit other periodic processes within the team, from synchronising sensor measurements with the communication protocol (improving data freshness) to synchronising distributed behaviours such as formation control and target tracking [10].

A common technique to achieve synchronisation is setting up a single absolute time reference, that is, a global clock. The time reference can be external, e.g., using GNSS or NTP over the Internet, or internal, using extra messages to exchange clock values. However, transmitting periodically on predefined time instants does not cope well with external interference produced by nodes outside the team, or even by nodes joining the team asynchronously. In particular, critical interference periods may still occur when the interference is periodic with a period close to an integer (sub)multiple of the TDMA round [3].

Alternatively, looser forms of synchronisation, such as relative synchronisation using the delay patterns suffered by messages exchanged within the team, are simpler to implement and have been shown to be effective in practice in highly dynamic mobile robotic scenarios, e.g., robotic soccer competitions, outperforming traditional (absolute) clock synchronisation in robustness. This technique is the basis of the Reconfigurable and Adaptive TDMA (RA-TDMA) protocol [4] that was developed for infrastructured networks.

In this paper we take inspiration from RA-TDMA to put forth the following contributions:

- The Ad-hoc Reconfigurable and Adaptive TDMA (RA-TDMA+) protocol to achieve relative synchronisation in ad-hoc mesh networks of mobile robots;

- An original TDMA framework modeling technique based on phases of periodic processes;
- A formal proof of convergence (synchronisation) to a global periodic TDMA framework under typical team operating conditions;
- A heuristic that allows synchronising the robots in the cases beyond those covered by the previous proof.

To the best of our knowledge, this is the first guaranteed clockless (relative) synchronisation framework for dynamic mesh networks of mobile robots that works over commodity wireless protocols, facilitating real-time.

The remainder of the paper is organised as follows. The next section discusses related work. Section 3 presents the background of the global synchronisation approach leading to the RA-TDMA+ protocol. Section 4 introduces the TDMA framework modelling approach relying on phases. Then, in Section 5 we present the synchronisation convergence analysis, including the formal proof of convergence and a complementary heuristic for the few cases that do not fit the proof conditions. Finally, Section 6 presents the protocol validation via extensive simulation while Section 7 concludes the paper.

II. Related Work

Currently, IEEE 802.11 [1] is arguably the most common standard RF communications technology used by mobile robots due to its robustness, cost, and reach. Since it is a commodity and general purpose, overlay protocols are frequently defined on top to provide specific properties.

A commonly desired additional property is reduced or no access collisions, to improve the quality of communications. IEEE 802.11 uses CSMA/CA arbitration, a mechanism that detects collisions and invokes a back-off and retry procedure that is known to be particularly effective with a lightly loaded medium. However, when the load increases, variable and unpredicted delays may develop. Nonetheless, CSMA/CA is still quite useful as a base layer for solving collisions created by unpredictable or uncontrollable interferences.

One way of handling these conditions, is the token-passing approach for IEEE 802.11 proposed by the WIreless Chain networK Protocol (WICKPro) [2][5] aiming at providing real-time guarantees for Wireless Mesh Network (WMN) in chain topologies. The protocol uses a typical cyclic approach with a macro-cycle that repeats itself over time, and is formed by several micro-cycles within which the protocol circulates a token to serialise transmissions. A token master enforces the periodic token release that synchronises all robots in the network. Synchronisation is thus centralised and strictly following the clock of the token master, particularly not adapting to external interference.

Other approaches use implicitly defined transmission slots mechanisms, e.g. using TDMA or transmission schedules. For example, the work presented in [6] defines transmission schedules that are calculated by all robots in parallel, and updated at an agreed future instant every time there is a topology change. This relies on global clock synchronisation, as robots need to agree on the times of each transmission and when the schedule is to be updated.

There are many algorithms that provide global clock synchronisation, for example DMTS [13], where the time master broadcasts a timestamp that receivers use to update their clocks. In [9], a master clock is used, in this case part of the Access Point (AP) in a IEEE802.11 network. When the AP sends a beacon that contains its timestamp, receivers get the packet near-simultaneously, allowing them to synchronise their local clocks with the AP clock.

FLOPSYNC-2 [14] is another clock synchronisation approach in which a master node imposes its clock, from which slaves align their local times. The master transmits a periodic packet, which is received by all slaves, each of which calculates a correction factor. To have all the slaves receive the master message, FLOPSYNC-2 combines with Glossy [7], a protocol that uses concurrent transmissions of the same packet that interfere constructively at each hop, flooding the multi-hop network in the minimum number of hops from transmitter to the farthest receiver (network radius). This is a rather promising flooding technique but it is highly dependent on the clock precision achieved. Currently, to the best of our knowledge, there are no reported implementations on commodity IEEE 802.11 technology, yet, and it is unclear how this technique would cope with asynchronous transmissions in an open environment, thus being inadequate for our context.

A relevant aspect is that global rigid periodic frameworks based on clock synchronisation perform worse when they are exposed to critical interference periods, i.e. periods in which external periodic transmissions are synchronous with robots transmissions, while looser synchronisation approaches, e.g., relative, fare better on these periods [4]. In fact, by adjusting the offset of periodic processes in response to relative delays it is possible to escape from periodic interference, as opposed to frameworks which transmissions are stuck to fixed instants in an absolute timeline. The work in [8] is an example, achieving the synchronisation of multiple agents in a distributed and adaptive way. In this work, agents emit a periodic pulse (not messages) with a known frequency. Agents that receive a pulse either delay or advance their own pulses within certain constraints, in an effort to match the received one. Eventually, agents synchronise and start emitting simultaneously. However, this technique cannot be directly used in our context, either. Firstly, the transmission of messages instead of pulses is significantly more prone to delays and omissions (losses). Secondly, advancing transmissions may imply a transient violation of the bandwidth allocated to each robot. Finally, the purpose of our synchronisation is to prevent simultaneous transmissions of different messages, which would lead to collisions and unpredictable delays.

On the other hand, RA-TDMA [4], a protocol developed within the Cooperative Autonomous Mobile roBots with Advanced Distributed Architecture (CAMBADA) robotic soccer team [10] for the RoboCup Middle-Size League (MSL), follows similar goals to ours. This protocol organises the team communications in a periodic TDMA round. However, when a robot's transmissions are delayed, e.g. due to interference, all other robots delay their transmissions, too, in an attempt to keep their slots separate and in order, without relying on a global clock. This corresponds to delaying the whole TDMA round, enforcing a loose clockless synchronisation. This mechanism proved to be effective when compared with using absolute clock [3] in harsh operational scenarios, such as the RoboCup MSL games. Contributing to reduce collisions within the team and eliminating critical interference periods.

Despite its favourable properties, RA-TDMA follows the constraint imposed by the MSL league of using a managed, i.e., infrastructured, network. Thus, it does not work in dynamic mesh networks leaving open the development of a new clockless synchronisation approach that could exhibit similar properties in such challenging domains.

III. RA-TDMA+

Our target is to support real-time collaboration among teams of connected robots that periodically share their state. We assume that robots can move freely within a given area of operation that is larger than the individual communication ranges. Thus, each robot can only exchange messages with its neighbours, but needs to communicate with the entire team leading to a dynamic mesh topology. Our first early attempt was a global TDMA framework relying on a global clock but we faced problems with critical interference periods, as explained before. We then moved to RA-TDMA, and faced several additional challenges arising from the robots lack of a direct view of the team membership and network topology. In addition, synchronisation based on perceived delays is also more complex since each robot listens to a potentially small subset of the team members, only.

In this section we introduce the RA-TDMA+ protocol that addresses these challenges in a fully distributed way and enforces the desired global synchronisation extending the basic mechanism of RA-TDMA. One of our requirements is to build it on a standard communication layer that provides CSMA/CA type of arbitration (e.g. ad-hoc IEEE 802.11), to cope with external interference, i.e., from sources outside the team, as well as new robots that may join the team asynchronously, interfering until they receive their own slot.

We took inspiration from our earlier implementations [12]: ideas for information dissemination and for tracking of both network topology and team membership. We briefly describe these mechanisms in this section.

A. The team of robots and its TDMA round

Let $\mathcal{T}_A$ be the set of N *active* robots, i.e., those that are currently engaged in the team. This set may vary dynamically as robots join or leave, but we consider these changes to occur at a low average rate compared with the speed of the topology tracking mechanism.

The robots in the team communicate one at a time in a cycle, i.e., the TDMA round. The duration of the round (T_{up}) defines the reactivity of the communications and is set according to the real-time requirements of the cooperative applications the team executes. This parameter is fixed and known a priori by all robots in the team. The round is divided in N consecutive time slots of equal duration ($t_{\mathtt{slot}} = T_{up}/N$), each assigned exclusively to a robot in the team. As the number of robots in the team increases, the slot duration decreases, that is, there is a trade-off of scale for bandwidth.

Each robot has an unique numeric physical ID. The protocol uses a simple algorithm that assigns slots to robots by increasing physical ID. For example, in the team $\mathcal{T}_A = \{ID_2, ID_3, ID_5\}$ robot ID_2 gets slot s_0, robot ID_3 gets slot s_1, and robot ID_5 gets slot s_2. Since the team is dynamic, it is convenient to refer to the robots in a way independent from their physical ID. Thus, the same algorithm assigns each active robot a logical ID that corresponds to its slot, i.e., the logical ID l_i is assigned to the robot that owns slot s_i. As long as all robots have a consistent view of the team, i.e., consistent team membership, they will all derive the same slot assignment and unique logical IDs. In particular, they will all know which slot marks the beginning of the round, i.e., slot s_0.

B. State dissemination

We use flooding with aggregation to share the state of the robots throughout the mesh network. The state of each robot encompasses its view of the network topology, the team membership, and other team-shared information. Each robot disseminates its own state in the beginning of its slot using a multicast message to the entire team. When a robot receives the state from a neighbour, it updates its own state. The state of a robot also includes the *age* of each state variable, which is set to zero when transmitted by the robot owning the variable and it is incremented as the variable travels through the network mesh. State aggregation only keeps the freshest values.

This mechanism, similar to [6], allows the entire team to converge to the same team state in a bounded number of rounds, assuming there is a path between any pair of robots (i.e., a fully connected mesh). The number of rounds depends on network diameter, message loss rate, and topology changes (e.g., arising from motion).

Once the robot transmits its state message, the remaining slot time is available to transmit messages from applications running in the robots. An application example is a multi-hop unicast delivery mechanism, where the topology information (explained below) is used to set-up routes in the network [11].

C. Tracking topology

The network topology is part of the team state and is represented by a connectivity matrix M (Fig. 1), where $M(j,k) = 1$ if robot l_j can receive messages from robot l_k[1]. Robot l_i has a version of the matrix (M^i) and it is responsible for updating row i indicating the neighbours from whom it receives messages. Note, however, that to avoid instability due to transient message losses, the state of each link is switched after a predefined number of consecutive rounds with a consistent behaviour, only, either connected or disconnected.

[1]The connectivity matrix uses signal strength for better link management but here we represent links as binary for simplicity.

The remaining rows are taken from the matrices received from the neighbours; as mentioned above, a row will only be updated if the received values are fresher than the current values, i.e., lower age.

$$
\begin{array}{cc|c|c|c|c|c|c|}
 & & l_0 & & \cdots & & & l_i \\
\hline
 & & 0 & 1 & 2 & 3 & 4 & 5 \\
\hline
l_0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\
 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
\vdots & 2 & 1 & 0 & 0 & 0 & 0 & 1 \\
 & 3 & 1 & 0 & 0 & 0 & 1 & 1 \\
 & 4 & 1 & 0 & 0 & 1 & 0 & 0 \\
l_i & 5 & 0 & 0 & 1 & 1 & 0 & 0 \\
\hline
\end{array}
$$

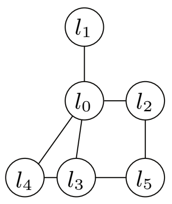


Fig. 1: Connectivity matrix representing a team of 5 robots and the corresponding network topology

Since each robot is constantly updating its own matrix, any change in the network topology is propagated in the following round, eventually reaching all robots a few rounds later.

However, in the dissemination process, matrices may be temporarily inconsistent among robots, that is, some of the paths may no longer exist or vice-versa. This inconsistency is transient and intrinsic to the system, and collaborative applications executing in the team must be aware of it. If transient disruption of certain links is not tolerated, adequate topology control must be used at the team level, e.g., doing formation control.

D. Tracking membership

Membership refers to the current team composition, $\mathcal{T}_A$. Tracking membership is fundamental for the communications management for two reasons: (a) to define N, the current team size, which is used to divide the communications round in N slots, and (b) to determine each robot's slot allocation. Thus, the connectivity matrix includes a row with robots' physical IDs (top row in Fig. 1).

Whenever a robot in the team receives a message from a joining team member (asynchronously), it inserts a new row and column in its matrix for the new robot in a position consistent with the slot allocation policy and increases N, effectively creating a new slot. This will be done by all team members that hear the joining robot. In the following round, the updated matrix is propagated to its neighbours, signalling the arrival of the new member and its inclusion in the TDMA schedule. The joining robot will also receive messages from its new neighbours and update its matrix according to the current team. Removing robots from the team can be triggered by any team member, although typically there will be several robots initiating the action. This is done when a robot (or more) detects one or more null rows or columns in the matrix for a predefined consecutive number of rounds. Such rows and columns are deleted from the matrix, N updated accordingly and the matrix propagated through the network. If the team network is partitioned in cliques, each will form its own team with its own matrix. Most importantly, as soon as one robot links several cliques, they are merged back into a single team.

Collisions are handled by the CSMA/CA arbitration of the native wireless protocol that we keep active. This is useful to tolerate external traffic but also during joining phases, while joining robots transmit asynchronously and until they get a slot and synchronise with the team.

E. Team synchronisation mechanism

Recall that our purpose is to synchronise the team without using absolute physical time, but relying on periodic network state dissemination to enforce the slot allocation policy. Thus, whenever a robot l_k receives a state packet from a neighbour, it can assert if the packet was received when expected or if it was delayed. If a delay is detected, then the robot delays its own next transmission by the same amount in order to maintain the slots in order and separated. In fact, l_k assesses the delays of all state packets received in a round and uses the maximum for its own compensation, achieving synchronisation even with mesh topologies (Section V). Note that the actual source of delays is irrelevant to the protocol, whether coming from interference in the medium, in the protocol stack or from drifts of robots internal clocks or even from misconfiguration with different values of T_{up}. Moreover, the overhead is negligible, owing to both the small state packets to synchronise communications and to the simplicity of the algorithm.

Fig. 2 shows an example of the synchronisation process in a line topology accommodating a delay of δ time units affecting robot l_0 transmission at instant a). The dashed vertical lines represent the slot boundaries in the original round structure, i.e., before the delay, while the solid vertical lines represent the slots in the shifted round structure, after accommodating the delay. The arrows show the propagation of the delays as imposed by the synchronisation mechanism.

The initial delay at time a) was detected by robot l_2, the recipient of the message, which delays its own transmission by the same amount, i.e., δ time units, when it transmits its own message, at time b). This delay is then detected by robot l_1, which also delays its next transmission, at time c). Similarly, this delay is now detected by robot l_3 that also delays its next transmission at time d), concluding the synchronisation of the team. If there are different delays affecting the messages received by a robot in one round, it considers the maximum to delay its transmission. However, if the delays are too large, the robot communications can be stalled, jeopardising its real-time properties, or they can also fall within a slot of another robot. This is prevented truncating large delays that affect the received messages to an upper bound (Δ). The value Δ has significant consequences for the dynamic behaviour of the synchronisation mechanism, which will be discussed later on.

Fig. 3 shows in the Y-axis the offsets of the robots transmissions in each TDMA round, with respect to the transmissions of robot 0, as shown in [12]. The X-axis shows the sequence of rounds in a timeline.Five robots are communicating using $T_{up} = 200ms$ and several consecutive network reconfigurations occur, with robots joining and leaving. Robots 0, 2, and

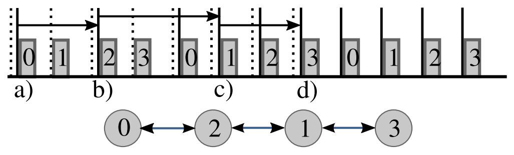


Fig. 2: RA-TDMA+ synchronisation: (bottom) robots topology; (top) delay propagation. Dotted lines represent the slot structure before a delay occuring at time a) and the solid lines represent the slot structure after that delay is accommodated.

4 form a network; robot 3 joins at round 24 (solid vertical line) causing a resynchronisation from 3 to 4 slots. At round 45, robot 1 also joins (second solid line). Then, at round 60, robot 4 leaves the team (dotted line), which causes a resynchronisation on round 72 (dashed line). Note that robot 4's absence was confirmed after 10 consecutive rounds without transmitting and then a reconfiguration was triggered to have it removed from the team. The same happens when robot 2 leaves the network on round 73 (dotted line), and is removed from the team on round 86 (dashed line).

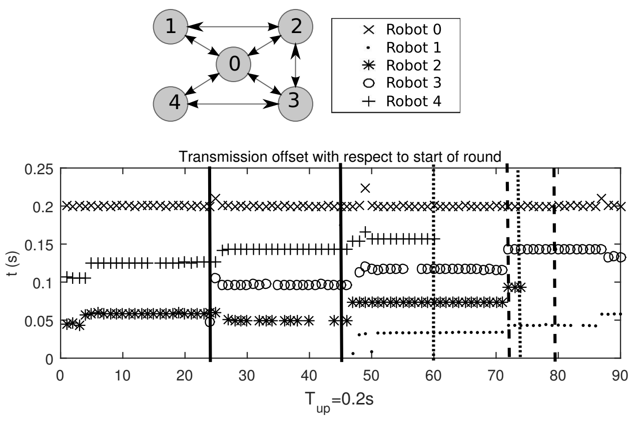


Fig. 3: Synchronisation protocol operating with robots joining (at $t = 24, 45$), leaving (at $t = 60, 73$) and removed from the team (at $t = 72, 86$). Y-axis is offset of transmissions in each round. X-axis is sequence of rounds in a timeline.

IV. Timings and phases in RA-TDMA+

To formalise our synchronisation problem, we first define the TDMA round as shown in Fig. 4 (right). Round k starts at t_{r_k} as given by Eq. (1), where t_ϕ represents the offset of round zero, i.e. $t_\phi = t_{r_0}$. Slot i in round k starts at $t_{s_{k,i}}$ as given by Eq. (2), which is expressed as an offset with respect to the start of the respective round. Note that $\forall_k.\ t_{s_{k,0}} = t_{r_k}$ and $t_{s_{k,N}} = t_{r_{k+1}}$.

$$t_{r_k} = t_\phi + k \times T_{up}, \texttt{ where } k \in \mathbb{N}_0 \tag{1}$$

$$t_{s_{k,i}} = t_{r_k} + i \times \frac{T_{up}}{N}, \texttt{ where } i = 0, \ldots, N-1 \tag{2}$$

1) Transforming time to phases: Similarly to many domains that handle periodic processes (e.g., AC circuit analysis), we adopt a phase representation of time, using a simple linear conversion (see Eq. (3)) of the absolute time for each robot $[0, \infty($ (Fig. 4 right) to a corresponding angle θ in the phase space $[0, 2\pi($ that represents one single round (Fig. 4 left). All time instants separated by T_{up} are mapped onto the same point in the phase space (same point in the circumference). Offsets and delays are represented as angles in this space. Thus, the extra complexity of considering absolute rounds that is intrinsic to the timeline approach, and which is irrelevant for synchronisation purposes, is avoided using phases. As the time progresses for a robot, the corresponding phase moves around the circle counter-clockwise. When the phase reaches points a, b, and c respectively, slot 0, 1, and i start (as indicated by the angle θ).

$$\theta(t) = \frac{2\pi}{T_{up}} t \tag{3}$$

Let us define ϕ the initial phase of a given periodic round with initial time offset t_ϕ. The conversion by Eq. 3) of an arbitrary round k, as in Eq. 4, shows how index k becomes irrelevant and the start of all rounds maps to ϕ, i.e., $\forall_k.\ \phi = \theta(t_{r_k}) = \theta(t_\phi)$.

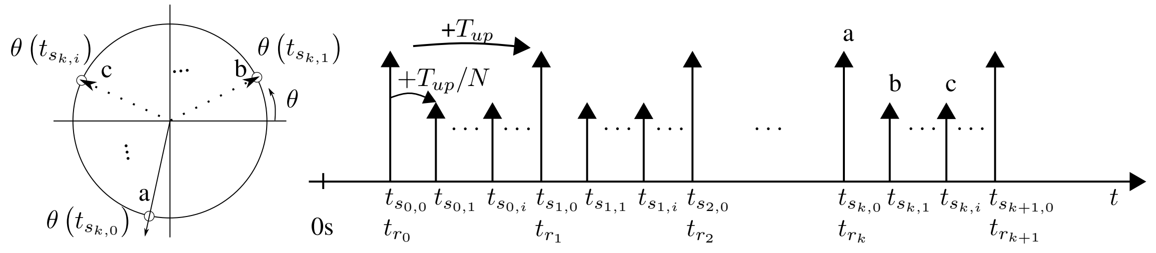
$$\begin{aligned}\theta(t_{r_k}) =& \frac{2\pi}{T_{up}} \times (t_\phi + k \times T_{up}) \\ =& \theta(t_\phi) + 2\pi k = \theta(t_\phi) = \phi\end{aligned} \tag{4}$$

Similarly, we define σ_i as the phase-equivalent of the start time of slot i using Eq. 3 for $t = t_{s_{k,i}}$ (Eq. 2), which again eliminates index k, leading to Eq. 5.

$$\begin{aligned}\theta(t_{s_{k,i}}) =& \frac{2\pi}{T_{up}} \times t_{s_{k,i}} \\ =& \frac{2\pi}{T_{up}} \times \left(t_{r_k} + i \times \frac{T_{up}}{N}\right) = \phi + i \times \frac{2\pi}{N} = \sigma_i\end{aligned} \tag{5}$$

2) Quantifying the synchronisation status: We note that each robot l_i has its own local view of the start of the communication round, with phase ϕ_i, relative to which it defines its own transmission slot s_i. The synchronisation mechanism explained above attempts at making all round phases equal, that is, $\forall_{i,j}\phi_i = \phi_j$, so that all slots are referred to a similar time reference. Let us also define Φ as the set of phases of all robots within the team, $\Phi = \{\phi_0, \phi_1, \ldots, \phi_{N-1}\}$. Similarly, let Φ_i be the phases of all robots that are neighbours of robot l_i including its own phase, i.e., $\Phi_i = \{\phi_i \cup \{\phi_k : M^i(i,k) \neq 0 \wedge M^i(k,i) \neq 0\}\}$.

Similarly to [8], we define the Arc(Φ), denoted A_Φ, as the shortest phase interval that contains all the phases of set Φ (Fig. 5). Using A_Φ we can measure the state of synchronisation of the network since the smaller the value of the Arc, the smaller are the differences between the phases of the robots. For example, Fig. 5 shows a set of robots on the left that are closer to synchronisation than those shown on the right. However, measuring this value is not simple. Conversely, its



$\theta(t_{s_{k,i}})$
$\theta(t_{s_{k,1}})$
$\theta(t_{s_{k,0}})$
a
b
c
θ
$+T_{up}$
$+T_{up}/N$
$t_{s_{0,0}}$ $t_{s_{0,1}}$ $t_{s_{0,i}}$ $t_{s_{1,0}}$ $t_{s_{1,1}}$ $t_{s_{1,i}}$ $t_{s_{2,0}}$ $t_{s_{k,0}}$ $t_{s_{k,1}}$ $t_{s_{k,i}}$ $t_{s_{k+1,0}}$
0s
t_{r_0} t_{r_1} t_{r_2} t_{r_k} $t_{r_{k+1}}$
t

Fig. 4: Right: TDMA round structure where t_{r_k} is the start of round k and $t_{s_{k,i}}$ is the start of slot i of round k, the round duration is $\mathtt{T_{up}}$ and the slot duration is $\mathtt{T_{up}}/N$ with N active robots. Left: The corresponding phase representation.

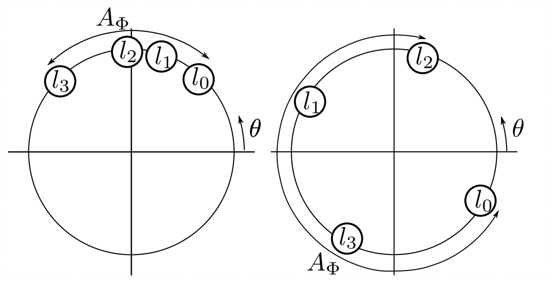


Fig. 5: The Arc of a set of phases – The left set is closer to synchronisation (smaller Arc) than the right set (large Arc).

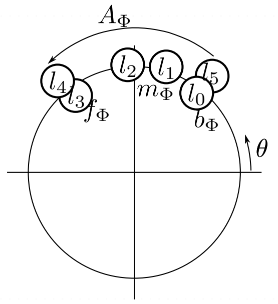


Fig. 6: Robots can either be $\mathbf{f}_\Phi$, $\mathbf{b}_\Phi$, or $\mathbf{m}_\Phi$, if they are in the front, back, or middle of the Arc, respectively.

complement, defined as the pair of consecutive phases furthest apart from each other, i.e., ($A^*_\Phi = 2\pi - A_\Phi$) is much simpler to measure, as follows.

To measure A^*_Φ, we first sort the set Φ with growing phase, thus obtaining $\Phi' = \{\phi'_0, \phi'_1, \ldots, \phi'_{N-2}, \phi'_{N-1}\}$, where $\phi'_0 < \phi'_1 < \cdots < \phi'_{N-2} < \phi'_{N-1}$. Using this ordered set we can calculate the clockwise distances[2] between every pair of consecutive sorted phases $D_{\Phi'} = \{d'_{0,1}, d'_{1,2}, \ldots, d'_{N-2,N-1}, d'_{N-1,0}\}$. Consequently, distance $d'_{i,j}$ normalised in the phase space $[0, 2\pi)$, between the phases of robot l_i and l_j can be computed with Eq. 6.

$$d'_{i,j} = (\phi'_j - \phi'_i) - 2\pi \left\lfloor \frac{\phi'_j - \phi'_i}{2\pi} \right\rfloor \tag{6}$$

Finally, we can obtain A^*_Φ as in Eq. 7 and, using this result, solve Eq. 8 to calculate A_Φ.

$$A^*_\Phi = max\,(D_{\Phi'}) \tag{7}$$

$$A_\Phi = 2\pi - A^*_\Phi = 2\pi - max\,(D_{\Phi'}) \tag{8}$$

Within A_Φ, robots can be classified according to their phase ϕ, as shown in Fig. 6:

- $\mathbf{f}_\Phi$ – Robots that have the most advanced phase within the arc A_Φ (*front*)
- $\mathbf{b}_\Phi$ – Robots that have the earliest phase within the arc A_Φ (*back*)
- $\mathbf{m}_\Phi$ – Robots that are neither in $\mathbf{b}_\Phi$ nor a $\mathbf{f}_\Phi$ (*middle*)

Robots can only be classified in one of the categories above; if all have the same phase, we say all robots are $\mathbf{f}_\Phi$ robots.

A. The synchronisation method in the phase space

When robot l_j reaches its slot and sends a packet ($\mathtt{P}_j$) to its neighbourhood, its neighbours can estimate the phase ϕ_j of l_j. For that purpose, when a neighbour l_i receives the packet from l_j at time $t^j_{\mathtt{rx}}$, it calculates the slot start of the latter robot ($t_{s_{k,j}}$) as in Eq. 9, where $t^j_{\mathtt{len}}$ is the packet transmission time, i.e., the time required to transmit b bits at the network bit rate. Using this value in Eq. 5, we can calculate the phase of slot s_j (Eq. 10) and then the phase (ϕ_j) of robot l_j (Eq. 11).

$$t_{s_{k,j}} = t^j_{\mathtt{rx}} - t^j_{\mathtt{len}} \tag{9}$$

$$\sigma_j = \theta\left(t_{s_{k,j}}\right) = \phi_j + j \times \frac{2\pi}{N} \tag{10}$$

$$\phi_j = \sigma_j - j \times \frac{2\pi}{N} \tag{11}$$

When robot l_i reaches its own slot, it will have collected the phases of all its neighbours ($\mathcal{N}_i$) in the previous round, and can calculate how much delay (δ_i) it needs to add to its next transmission to synchronise with the team (Eq. 12).

$$\delta_i = \max_{k \in \mathcal{N}_i \cup l_i} (\phi_k - \phi_i) \times \frac{T_{up}}{2\pi} \tag{12}$$

Note that δ_i is never negative since at least $\phi_k - \phi_i = 0$ for $k = i$. On the other hand, if a limit is not imposed on

[2]We define the clockwise distance between two phases as the angle separating them in a clockwise direction.

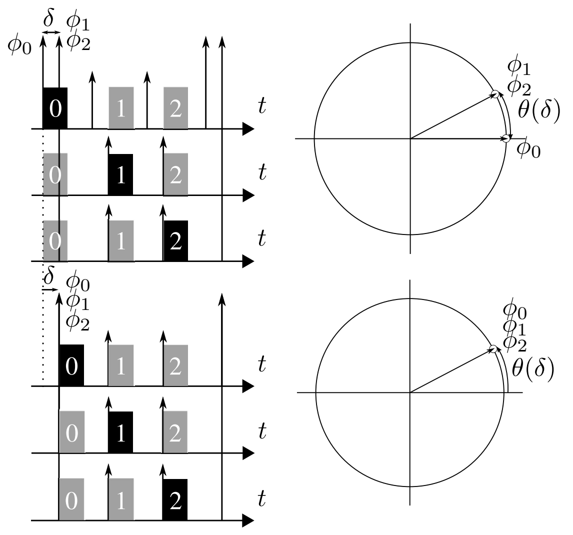


Fig. 7: Synchronisation in two consecutive rounds. Top, l_0 detects a delay of δ in l_1 and l_2; bottom, l_0 delays its transmission δ and recovers synchronisation.

this delay, robots can be a long time without transmitting, jeopardising their real-time communications; or even start transmitting in another robot's slot. To prevent such situations, we define a delay saturation value Δ that we apply when adjusting the phase of each robot (Eq. 13, where ϕ_i^{nxt} is the new phase for robot l_i and ϕ_i^{now} its current phase). The specific value of Δ determines how much delay a robot can apply to its TDMA round, thus it also determines the speed of convergence. Increasing Δ allows larger phase adjustments, thus longer delays but faster convergence. Conversely, shorter Δ constrains the delays but slows convergence. This is a trade-off between speed of convergence and stability of the instantaneous round period.

$$\phi_i^{\text{nxt}} = \phi_i^{\text{now}} + \min(\Delta, \delta_i) \times \frac{2\pi}{T_{up}} \tag{13}$$

Fig. 7 shows the synchronisation process. The top part represents a round in which l_0 detects a delay of δ relatively to l_1 and l_2. The lower part shows the next round where robot l_0 adjusted its phase to synchronise with l_1 and l_2. The left side shows the robots timelines while the corresponding phase representation from the perspective of robot l_0 is on the right.

B. Implementation of the phase-based approach

The synchronisation approach represented in the phase space was implemented in Matlab for extensive simulation purposes. A video of a simulation showing the synchronisation process is available at https://youtu.be/l0soqkJyKkE. In the video, robots are joining the team one by one, until the team has five active members. Their transmissions are being affected by random delays that trigger the synchronisation mechanism and represent load in the network beyond the team communications, as well as the asynchronous transmissions of the joining nodes. Then, robots start leaving the team, until only three are left.

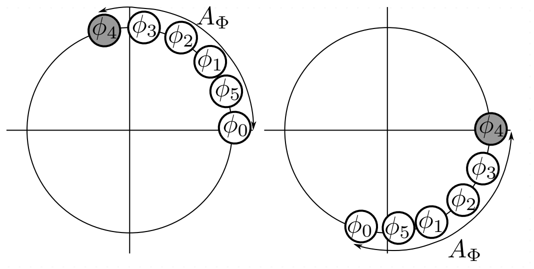


Fig. 8: Different views by different robots. The Arc of the team is smaller than π: all robots, either l_0 (left) or l_4 (right), agree on the most advanced phase (l_4 - dark grey circle).

V. Convergence of the Synchronisation Algorithm

In this section, we determine the initial conditions that enforce convergence of our synchronisation algorithm (Section V-A), that is, we carry out an analysis of when our algorithm will reach synchronisation. However, under certain peculiar starting conditions, particularly when robots join, convergence is not guaranteed. For those situations, we propose an heuristic that avoids the pitfalls that prevent synchronisation (Section V-B) and, although we cannot prove it converges, has converged in every experiment during extensive simulations. Specifically, we ran simulations using 10 robots in 1500 different random topologies, and for each topology 1000 random starting conditions.

A. Proving the convergence

Assume that we have a network of N robots, each with its own view of the phase set, ϕ_i. $\forall_{i=0..N-1}$, forming an Arc. Also, assume that at a certain instant in time the phases are not aligned (not all equal), caused, for example, by asymmetric interference suffered by the robots or one robot that joined or left the team. When the phenomena causing phase misalignments cease, the synchronisation algorithm adjusts the phases towards the maximal one, as explained before, and we can state Theorem 1.

Theorem 1. *If the Arc of the network is within π, $A_\Phi < \pi$, then the proposed synchronisation protocol eventually converges to a consistent phase $\phi = \phi_i$, $\forall_{i=0..N-1}$.*

Proof. In order to prove that the phases eventually reach $\phi = \phi_i$, $\forall_{i=0..N-1}$, we will show equivalently that all robots l_i in the team $(i = 0..N-1)$ will become $\mathbf{f}_\Phi$ robots. In particular, we will show that using our protocol the number of $\mathbf{f}_\Phi$ robots increases for all i and never decreases.

Since all phases start within π, $A_\Phi < \pi$, all robots are able to agree on who has the most advanced phase[3]. For example, as we show in Fig. 8, the most advanced phase belongs to l_4 (dark grey circle); both from the point of view of robot l_0 (Fig. 8 left) and robot l_4 (Fig. 8 right)

[3]Note that robots are asynchronous and therefore do not know the slots or phases of other robots in the beginning.

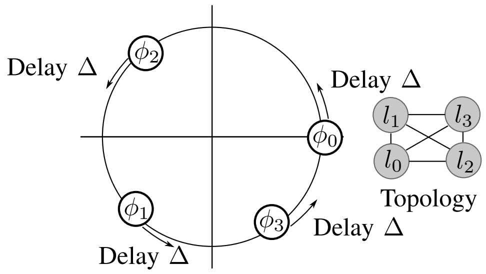


Fig. 9: Failure to synchronise (1): All robots measure a delay above Δ, thus all robots delay their own transmission by Δ. The relative distances of the phases is kept constant, and synchronisation is never reached.

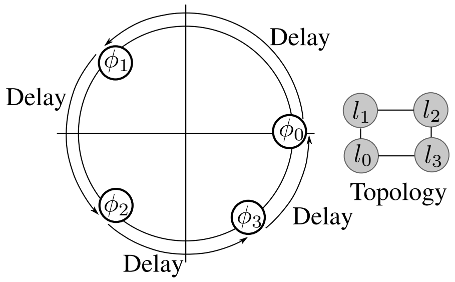


Fig. 10: Failure to synchronise (2): topology loop causes robots to adopt the phase of the previous one. Here, robots l_0, l_1, l_2, and l_3 will sequentially change their phases to those of robots l_1, l_2, l_3, and l_0 respectively, thus not synchronising.

Now consider an arbitrary robot l_i in the team and let $|F|$, $|M|$, $|B|$ be the number of $\mathbf{f}_\Phi$, $\mathbf{m}_\Phi$, and $\mathbf{b}_\Phi$ robots, respectively. If l_i is a $\mathbf{f}_\Phi$, then it will not detect a phase more advanced than its own, thus it will not delay its own transmission. If l_i is a neighbour of an $\mathbf{f}_\Phi$ robot, it will observe a phase more advanced than its own. Therefore, it will delay its own communications, increasing its phase according to Eq. 13 and reducing the difference to the phases of the $\mathbf{f}_\Phi$ robots until that difference is zero. At that point, it becomes an $\mathbf{f}_\Phi$ robot and $|F|$ increases. This reasoning can be repeated with the new $\mathbf{f}_\Phi$ robots and their neighbours, until $|M| = 0$ and $|B| = 0$. As all $\mathbf{b}_\Phi$ robots delay their communications, the value of A_Φ decreases, eventually reaching zero when all robots become $\mathbf{f}_\Phi$ robots and $|F| = N$. This trend occurs in all robots, eventually leading to the synchronisation of the whole team. □

B. Addressing the $\mathbf{A}_\Phi < \pi$ assumption

Despite the condition $A_\Phi < \pi$ being very frequent in practice, essentially corresponding to periods of stable team composition, there are still situations when $A_\Phi \geq \pi$, typically related to the arrival of a new robot. In these cases, certain rare scenarios may prevent synchronisation from converging, for example when all robots are turned on simultaneously with random phases or when unsynchronised groups of robots come together. Such an example is shown in Figure 9, in which all robots detect delays that are superior to the maximum allowed delay correction per round (Δ). The robots form a fully connected network (see right part of the figure) and, as a consequence, they all delay their communications by the same amount of time. The resulting behaviour is that their relative phases do not change, thus the robots remain unsynchronised transmitting with a period equal to $T_{up} + \Delta$.

We also found that a loop topology (Figure 10) where all robots have a neighbour that has a larger phase than their own, can prevent synchronisation convergence. For example, in the situation shown, each robot l_i will sequentially try to synchronise with robot l_{i+1} while robot l_{N-1} will try to synchronise with robot l_0. Consequently, all robots will delay their transmission in the round which, similarly to the previous problem, inhibits their ability to synchronise their rounds.

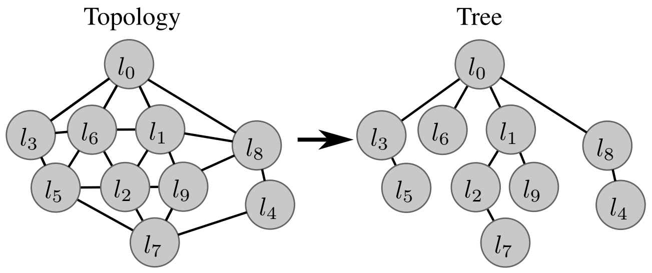


Fig. 11: Disseminating synchronisation information: Building a tree from the connectivity matrix allows limiting the set of neighbours with which robots synchronise, preventing loops.

To reach synchronisation even in such cases, we modified our protocol to avoid loops that hinder the ability to reach a consensus. Our solution is to build a spanning tree (thus eliminating loops) using the topology information contained in the connectivity matrix as shown in Figure 11. In the figure, messages that arrive through links that exist in the current topology (left) but not on the tree (right) are ignored for synchronisation purposes.

Since the matrix is available in all robots, each one of the team members can build its own tree locally following a common set of rules, thus obtaining the same tree, as follows. Initially, the root of the tree is the robot with the lowest ID, namely robot l_0. Then we add all neighbours of robot l_0 to the tree by increasing ID (increasing index), and repeat the process for the neighbours of the neighbours until all the robots are added. Note, however, that the tree does not actually limit the propagation of information. It is only used to select a subset of neighbours with which the robot synchronises, instead of synchronising with all of its neighbours.

The drawback of having a tree is a reduction in speed of disseminating synchronisation information throughout the network, thus slowing the synchronisation process. If robots were able to measure the value of A_Φ directly, then it would be simply a matter of using the tree when $A_\Phi \geq \pi$, only. However, since robots can only observe their neighbourhood, they have a limited vision of the network and robot l_i can only

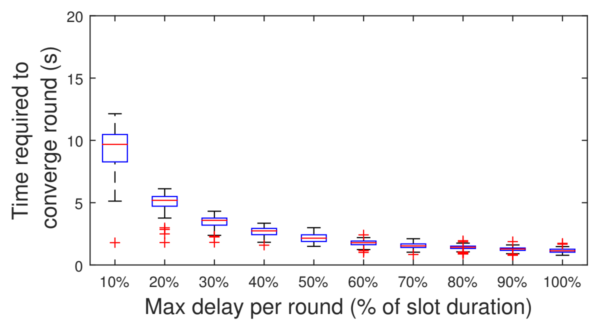


Fig. 12: Time to synchronise a static topology as a function of Δ, using $T_{up} = 200ms$ and random initial phases ($A_{\Phi} \leq \pi$).

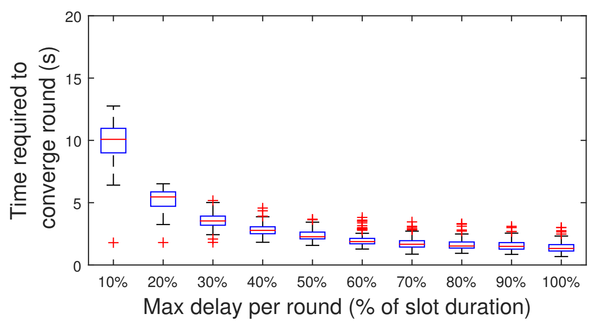


Fig. 13: Time to synchronise a dynamic topology against Δ, using $T_{up} = 200ms$ and random initial phases ($A_{\Phi} \leq \pi$).

measure the Arc formed by itself and its neighbours (A_{Φ_i}). Thus, to calculate an upper bound to the Arc of the team, each robot l_i shares A_{Φ_i} as part of its state. Then, robot l_i computes $\Sigma_i = \sum_{n\in[0,N-1]} A_{\Phi_n} \geqslant A_{\Phi}$ and uses the tree if $\Sigma_i \geqslant \pi$, which is a conservative but safe option.

As a practical issue, for the sake of stability of the synchronisation process, a hysteresis was added to this mode change. That is, only after a pre-determined number of TDMA rounds where the robot observes $\Sigma_i \geqslant \pi$ will it start using the tree. Similarly, only after the same number of rounds where $\Sigma_i < \pi$, will the robot stop using the tree. Moreover, to avoid inconsistencies as the ones caused by symmetries in the first example, when the user inputs the value of Δ (the maximum delay per round), each robot calculates its internal value as $(0.8 + 0.2\text{rnd})\,\Delta$, where *rnd* is a random number between 0 and 1 taken from a uniform distribution.

VI. Experimental Validation

RA-TDMA has been successfully used in robot soccer for several years [10] and a practical algorithm that served as inspiration for RA-TDMA+ was reported in [12]. In this section we analyse RA-TDMA+ through extensive simulations that cover a wide operational space and capture intrinsic protocol features. We assess the impact of (i) Δ, the maximum phase shift per round, and (ii) delays (regardless of original cause); our main metric is the time to synchronise robots under different conditions.

A. *Influence of threshold* Δ

We isolate the impact of Δ on the time required to synchronise. Remember this is an important design parameter that sets a trade-off between speed of convergence and stability of the round (i.e., the period does not increase much). To obtain data that represents the intrinsic characteristics of the proposed protocol, we do not consider any other source of interference (e.g., no clock drifts, no packet losses, and no message delivery delays) in this section.

We set up a network of 10 robots in a given topology communicating with a period T_{up} = 200 ms, and ran simulations for values of Δ ranging from 10% to 100% of the slot duration. For each of those values of Δ, we ran 100 different sets of starting phases, and measured the time required for the robots to synchronise. We run this experiment with $A_{\Phi} < \pi$ for two cases: (i) fixed topologies and (ii) dynamic random topologies, representing robots motion. In the latter case, the robots were changing positions every 10s, and take 2s to reach their new positions. We present the results in box graphs below with x-axis with different values of Δ and y-axis representing the time required to synchronise. As usual, the box top and bottom represent the 25% and 75% quartiles, the middle line represents the median, and crosses represent top and bottom 25% quartile values.

The main observation is that, in all cases, all robots were able to reach synchronisation, thus agreeing on a global TDMA framework.

We can see from Figures 12 and 13 that the difference between static and dynamic topologies is almost imperceptible, given that for phases that do not differ too much, movement does not make a difference. In addition, we can see that the time to converge (in number of periods, or rounds) decreases exponentially to start with, and is stable when $\Delta > 0.2$ (i.e., increasing Δ speeds up synchronisation). This is expected since, according to our analysis, the faster the robots can adjust their phases, the faster a consensus is reached. Moreover, since the phase differences are relatively small, the impact on the synchronisation time is insignificant for $\Delta \geq 50\%$.

We also repeated the same set of experiments without the $A_{\Phi} < \pi$ constraint, to study a situation where the robots are initially very unsynchronised (e.g., when one or more robots join the team simultaneously). Figures 14 and 15 show that the behaviour is less predictable, especially for values of $\Delta > 50\%$. We can see that static and dynamic situations have the same characteristics and values, when the maximum delay per round is 20% or above. In the case of 10% The dynamic case shows a larger time (average goes from 20s to 26s) and variance. The increased variability comparing to the static case, particularly when $\Delta > 50\%$, is due to, as mentioned above, the slower propagation of synchronisation in the imposed tree topologies.

Discussion Both dynamic and static experiments show that, when $A_{\Phi} < \pi$ the absolute time to synchronise is below 5s for $\Delta \geq 30\%$. These low values show that the protocol would

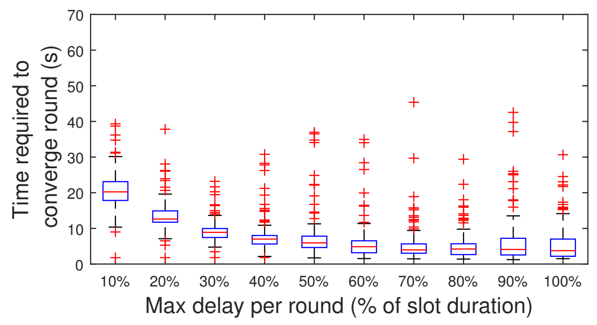


Fig. 14: Time to synchronise a static topology as a function of Δ, using $T_{up} = 200ms$ and arbitrary initial phases.

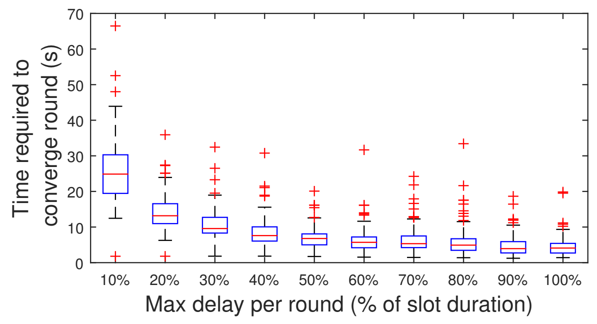


Fig. 15: Time to synchronise a dynamic topology as a function of Δ, using $T_{up} = 200ms$ and arbitrary initial phases.

enforce synchronisation in the team even for fast mobility, with topology changes as frequent as every 5s on average. Lastly, using a shorter round period (e.g., T_{up} = 100ms), would reduce proportionally the time to synchronisation, increasing resilience to dynamic topologies caused by mobility.

B. Impact of delays on the synchronisation

We analyse the impact of "observed delays" on the maximum phase difference, which is a measure of the team synchronisation. External interfering traffic, robots' clock drifts, operating system interference, etc. are all indistinctively sensed as part of the observed delays. Using the same values for T_{up} and number of robots, we fix $\Delta = 40\%$ (i.e., 8ms), and inflict a uniformly distributed random delay (between 0 and D) on all packet transmissions, where $D = 1, 2, ..., 10$ms (from [4] and more than Δ in some cases). For each value of D, we ran the experiment for 1000 seconds and recorded the value of A_Φ, that is, the maximum phase difference.

The results for both the static and the dynamic cases are presented in Figure 16, where the X-axis represents the largest difference of phases expressed in time ($A_\Phi/2\pi * T_{up}$) and the Y-axis represents the cumulative number of occurrences of that value. The results show that our synchronisation algorithm effectively keeps the system with $A_\Phi < \pi$ all the time, even for larger levels of interference. Note that with T_{up}=200ms a phase difference of π corresponds to 100ms. That is, the $A_\Phi \geq \pi$ condition never occurs in the experiments, although it may occur when new robots join the team (as we will see below).

Moreover, we see that larger interfering delay increases $\max A_\Phi$, i.e., for which CDF=1, in a near linear fashion, thus causing a graceful degradation of the synchronisation state. Note that, the lower A_Φ the more stable the TDMA round is (shorter phase adaptations). We also note the vertical lines (corresponding to the 99 percentile of $D = 1, 5,$ and 10ms) shift to the right from static to dynamic as expected, adding about 50% to 80% to the A_Φ, but still preserving it to less than 30ms.

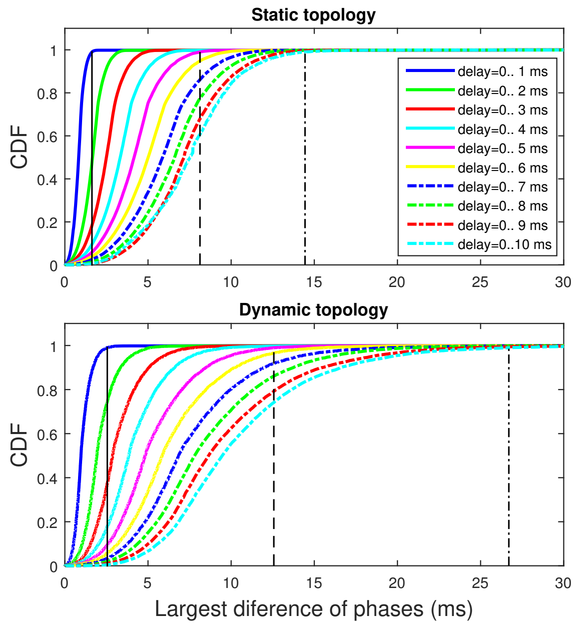


Fig. 16: CDF of the largest difference of phases ($A_\Phi/2\pi * T_{up}$) with different interference conditions, for different values of maximum message delays.

C. Impact of team changes on the synchronisation

To complete the experiments, we address a more general scenario in which we include robots mobility, random interference delays and changes in the team membership, with robots leaving and joining the team. We use the same metric, thus assessing the combined impact on the maximum phase difference. When membership changes, there is an interval of time in which the robots transmissions will lose synchronisation because their offsets (with respect to the first slot) are no longer valid. To measure this effect, we use the same pattern of movement as before and we remove one robot from the team every 50s, until only 3 robots remain. Then, we starting adding one robot back to the team every 50s until the team has 10 robots again. Using the same values for T_{up}, we fix $\Delta = 40\%$ (i.e., 8ms—26.6ms depending on the number of active robots), and inflict a uniformly distributed random delay (between 0 and 10ms) on all packet transmissions. We ran the experiment for 10,000 seconds (i.e., 200 changes in team membership) and recorded the value of A_Φ, that is, the maximum phase difference.

The results are presented in Figure 17, where the X-axis represents the largest difference of phases expressed in time ($A_\Phi/2\pi \times T_{up}$) and the Y-axis represents the cumulative number of occurrences up to that value. During this experiment, we observed several time instances where $A_\Phi \geq \pi$, caused by the membership changes. Nevertheless, this is a small fraction of all observed phase differences and our algorithm was always able to recover synchronisation and keep the system with $A_\Phi \ll \pi$ (recall that $\pi = 100ms$), as evidenced by the vertical lines corresponding to the 90, 95, and 99 percentiles.

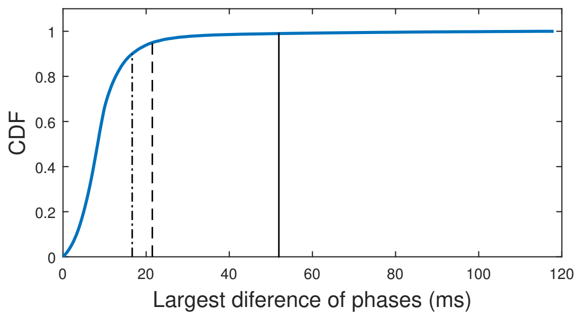


Fig. 17: CDF of the largest difference of phases ($A_\Phi/2\pi \times T_{up}$) with a varying number of mobile team members from 10 down to 3, then up to 10 again, repeatedly, every 50s.

VII. Conclusions

Our objective is the synchronisation of communications in a team of cooperating mobile robots forming an ad-hoc dynamic mesh network. We proposed a global TDMA protocol on top of a common collision detection network. Our proposed synchronisation approach is clockless, fully distributed, and symmetrical, in the sense that all nodes play a similar role (no master-slave coordination). To the best of our knowledge, this is the first such approach that can be provably used for synchronisation of periodic processes scattered over a mesh network without a global clock.

Using a consensus approach and a phase representation of the TDMA round, we proved that under common operating assumptions the network achieves synchronisation, allowing the nodes to use real-time-friendly TDMA framework. For the rare scenarios that go beyond the considered assumptions for the proof, we proposed a heuristic that allowed achieving synchronisation for all conditions tested. This has been verified by extensive simulations.

For future work, we will analyse which events (e.g., robots joining/leaving the team) can make a synchronised team of robots become unsynchronised with $A_\Phi > \pi$, and we will focus on the proof of convergence of the proposed heuristic.

Acknowledgements

This work was partially supported by the Portuguese Government through FCT grant UID/EEA/50008/2013 - Instituto de Telecomunicações.

References

[1] IEEE standard for information technology–telecommunications and information exchange between systems local and metropolitan area networks–specific requirements part 11: Wireless LAN medium access control (MAC) and physical layer (PHY) specifications. *IEEE Std 802.11-2012 (Revision of IEEE Std 802.11-2007)*, pages 1–2793, March 2012.

[2] J. Aisa and J.L. Villarroel. WICKPro: A hard real-time protocol for wireless mesh networks with chain topologies. In *Wireless Conference (EW), 2010 European*, pages 163–170, April 2010.

[3] Luis Almeida, Frederico Santos, and Luis Oliveira. Comparing adaptive tdma against a clock synchronization approach. In Andrea Bondavalli, Andrea Ceccarelli, and Frank Ortmeier, editors, *Computer Safety, Reliability, and Security*, volume 8696 of *Lecture Notes in Computer Science*, pages 71–79. Springer International Publishing, 2014.

[4] Luis Almeida, Frederico Santos, and Luis Oliveira. *Management of Cyber Physical Objects in the Future Internet of Things: Methods, Architectures and Applications*, chapter Structuring Communications for Mobile Cyber-Physical Systems. Springer, 2016.

[5] Jesús Aísa and José Luis Villarroel. The {WICKPro} protocol with the packet delivery ratio metric. *Computer Communications*, 34(17):2047 – 2056, 2011.

[6] Tullio Facchinetti, Giorgio Buttazzo, and Luis Almeida. Dynamic resource reservation and connectivity tracking to support real-time communication among mobile units. *EURASIP J. Wireless Communications and Networking*, 2005(5):712–730, 2005.

[7] Federico Ferrari, Marco Zimmerling, Lothar Thiele, and Olga Saukh. Efficient network flooding and time synchronization with glossy. In *Information Processing in Sensor Networks (IPSN), 2011 10th International Conference on*, pages 73–84. IEEE, 2011.

[8] Johannes Klinglmayr, Christoph Kirst, Christian Bettstetter, and Marc Timme. Guaranteeing global synchronization in networks with stochastic interactions. *New Journal of Physics*, 14(7):073031, 2012.

[9] M. Mock, R. Frings, E. Nett, and S. Trikaliotis. Continuous clock synchronization in wireless real-time applications. In *Proceedings 19th IEEE Symposium on Reliable Distributed Systems SRDS-2000*, pages 125–132, 2000.

[10] A.J.R. Neves, J.L. Azevedo, B. Cunha, N. Lau, J. Silva, F. Santos, G. Corrente, D.A. Martins, N. Figueiredo, A. Pereira, et al. Cambada soccer team: from robot architecture to multiagent coordination. *Robot Soccer*, pages 19–45, 2010.

[11] L. Oliveira, L. Almeida, and P. Lima. Multi-hop routing within tdma slots for teams of cooperating robots. In *2015 IEEE World Conference on Factory Communication Systems (WFCS)*, pages 1–8, May 2015.

[12] Luis Oliveira, Luis Almeida, and Frederico Santos. A loose synchronisation protocol for managing rf ranging in mobile ad-hoc networks. In Thomas Röfer, N.Michael Mayer, Jesus Savage, and Uluç Saranlı, editors, *RoboCup 2011: Robot Soccer World Cup XV*, volume 7416 of *Lecture Notes in Computer Science*, pages 574–585. Springer Berlin Heidelberg, 2012.

[13] Su Ping. Delay measurement time synchronization for wireless sensor networks. *Intel Research Berkeley Lab*, 6:1–10, 2003.

[14] F. Terraneo, L. Rinaldi, M. Maggio, A. V. Papadopoulos, and A. Leva. Flopsync-2: Efficient monotonic clock synchronisation. In *2014 IEEE Real-Time Systems Symposium*, pages 11–20, Dec 2014.

[15] Eustathia Ziouva and Theodore Antonakopoulos. Csma/ca performance under high traffic conditions: throughput and delay analysis. *Computer Communications*, 25(3):313 – 321, 2002. URL: http://www.sciencedirect.com/science/article/pii/S0140366401003693.