Pulse Coupled Oscillators for Drone-to-Drone Synchronisation

Anony Mouse
Universidad Anonymous
Anon
anon@anon

Not Known
Department of Not Known
Univeristy of Not Known
not@known

***Abstract*—The autonomous operation of drone swarms currently poses a challenge due the limited bandwidth and reliability of drone-to-drone or drone-to-controller communication. In this paper we present a decentralised drone-to-drone communication algorithm for synchronisation that uses a mesh network topology. Current approaches use centralised control and communication organised in a star topology that prevents drone-to-drone communication. This poses a single point of control an communication failure and scalability problems. Drone-to-drone communication organised as a mesh network is a viable option that could increase swarm autonomy and resilience to communication failure. In this paper we propose the Epoch Synchronisation with Pulse Coupled Oscillators (ESPCO) distributed algorithm for decentralised synchronisation. We design, implement, and demonstrate a Pulse Coupled Oscillator based agreement protocol that is decentralised and robust to individual drone failure.**

***Index Terms*—robotics, drones, autonomous vehicles, communication, protocols**

I. Introduction

Multi-Drone Systems (MDS) [1] are systems that consist of multiple autonomous unmanned aerial vehicles(UAVs) that work together to achieve an objective such as create aerial displays, perform search and rescue tasks, sensing tasks for precision agriculture [2] and infrastructure monitoring [3]. Most current MDS utilise centralised control organised in a star topology [1].

Autonomy of the drones in a MDS is a desirable quality for these applications [4]. Drone autonomy can occur at different levels, from being able to hover without any external influence, to being able to change plans dynamically as the environment changes mid-mission. For MDS there is a crucial system level of autonomy where the drones can agree upon a heading and velocity and move together without drone-to-drone collisions. This behaviour is referred to as flocking or swarming and is necessary to be able to perform the above mentioned tasks at scale or with higher levels of autonomy.

A key problem for autonomy in drone swarms is how to communicate in a decentralised fashion, or mesh topology, and organise local actions at a global level [5]. In particular, swarm **synchronisation** has been identified as a very useful behaviour [6] for swarm robotics. If the robots in a swarm can intercommunication and agree a common understanding of time, they can perform synchronised actions.

Certain MDS applications such as gas leaks detection [7] or search and rescue operations [8] may require that the drones operate in an environment of little or no central communication where the drones must operate autonomously of a central controller. In these circumstances, the drones would need to inter-communicate and synchronise among themselves. In this paper we report on an experiment with the use of the Epoch Synchronisation with Pulse Coupled Oscillator protocol (ESPCO) using drone-to-drone mesh communication to synchronise an MDS.

II. Background

Pulse Coupled Oscillators (PCO) is a model of decentralised, distributed synchronisation. This type of synchronisation is observed in biological systems such as the coordinated firing of heart muscle cells or the synchronised flashing of Malaysian fireflies [9]. Mirollo and Stogatz [9] proved that a network of fully connected pulse-coupled oscillators eventually converges with the oscillators are firing synchronously. Later the condition that the network is fully connected with all-to-all communication was lifted by Lucarelli and Wang [10].

Chia-Chu Chen [11] showed that providing a threshold condition of less than 1/2, whereby under this threshold an oscillator does not change it's firing timer, a system of pulse-coupled oscillators will still converge to synchronous firing. Pulse coupled oscillators were first demonstrated to be useful to synchronise wireless ad-hoc networks by [12].

The FiGo [13] algorithm combines polite gossip with pulse-coupled oscillation for decentralised sensor network synchronisation and dissemination. FiGO uses an all-to-all mesh topology. Polite message suppression is where when a node receives a message a neighbour, it will suppress it's next broadcast [14]. In FiGo it is shown that in a fully connected mesh network of sensor nodes with polite message suppression, the converge to synchronicity. The authors of [15] formally verify that sensors using FiGo synchronise even with polite message suppression.

A. MAV Synchronisation

Schilcher, Udo et al. studied solving issues around applying the swarmulators model, which couples synchronisation and swarming behaviour [16]. Swarmulators use stochastic coupling, a method for reducing the number of state broadcasts. Agata Barcis and Christian Bettstetter proposed Sandsbots following this work [17], that demonstrates the swarmulators model on synchronising and swarming for real world robots, both ground robots and on the Crazyflie, but used a central controller communicating with the drones in a star

topology. Fernando Perez-Diaz et al. [18] investigated small grounded mobile pulse-coupled oscillators synchronised using light pulses and directional cameras, and how changing the movement of the robots can change the speed of global convergence.

Fawaz Alsolami et al. [19] investigated time synchronisation in a UAV (Unmanned Arial Vehicle) network, where several drones autonomously generate swarm clusters, with a cluster head that initiates time synchronisation, synchronising within that cluster. These still require that each cluster has a leader, which decreases robustness in the time synchronisation if the cluster head is lost.

Anders Christensen et al. showed that synchronisation can be used to detect faults in a swarm by using pulse coupled oscillation to synchronise a swarm with LEDs, and then [20]. This is then used for swarm self-repair capabilities, shown in simulation and on small ground robots.

Geoffrey Werner-Allen et al. proposed the Reachback Firefly Algorithm (RFA) [21] to synchronise a network of nodes to within 100 microsec. This accounts for message delays and losses by time-stamping the messages at a low level to estimate the time delay before broadcast. There is however no notion of shared time provided for the network to perform coordinated actions. The Trickle algorithm proposed by Philip Levis et al. [22] introduces polite message suppression, to disseminate information to a network of nodes with a small trickle of packets. The above algorithms were introduced for wireless sensor networks, and as such do not have the latency requirements of in-flight drone swarms.

III. ESPCO Algorithm

The pseudocode for the Epoch Synchronisation with Pulse Coupled Oscillators (ESCPO) algorithm can be seen in Algorithm 1. ESPCO uses broadcast radio communication. It assume that the drones are organised in an all-to-all network topology.

Every time the synchronisation timer fires, the epoch will be incremented by one. We then start a new synchronisation timer with a time period depending on our offset from any other drone's synchronisation messages, using the `sync_offset`. If we have not received any synchronisation messages in the last period, then we send a sync message. This is the *polite gossip* part of the algorithm, and results in less messages being sent to synchronise. Then, with probability `P` we start a timer to send an epoch message to the other drones halfway through the next period. This ensures that the epoch will be correct during the period. We send it halfway through the period because if we send the epoch straight away then it could produce instability in the epoch number, since even when the drones are synchronised there will still be a small synchronisation error between them. If a drone receives an epoch number that is higher than it's own then it updates it's own epoch to that value.

The idea of stochastically sending state updates to synchronising nodes was also explored in Bettsetter work on Swarmulators, which couple synchronisation with positional state [16]. However in this paper this is applied to the problem of coupling positional state updates and synchronisation messages rather than a shared time epoch, and the synchronisation messages are sent stochastically, whereas we employ message suppression with the polite gossip paradigm.

Also in order to quickly update new drones that join the network, if a drone receives a synchronisation message with an epoch number is more than two behind the current epoch, then it will send an epoch message to get the new drone up to date. This means when a drone joins the network, once it has received a synchronisation message from the synced drones in the network, it cannot ever be more than one epoch out from the other drones.

When a synchronisation message is received, we get the time `now` which is the time since our current period's timer started. We then set this to `sync_offset`, which means when the current timer expires the next timer will only last for this offset time. This means that after the next timer has expired the two drones timers should fire at the same time. This is proven to synchronise as shown in [11].

Algorithm 1 ESPCO

```
1: epoch ← 0
2: sync_offset ← 0
3: received_count ← 1
4: P ← 20
5: procedure SYNC_TIMER_EXPIRE
6:     epoch ← epoch + 1
7:     if sync_offset > 0 then
8:         START_SYNC_TIMER(sync_offset)
9:     else
10:        START_SYNC_TIMER(PERIOD)
11:    end if
12:
13:    if received_count < 1 then
14:        SEND_SYNC_MESSAGE(epoch)
15:    end if
16:
17:    random ← choose random number from 0 to 100
18:    if random < P then
19:        START_EPOCH_TIMER(PERIOD/2)
20:    end if
21:
22:    received_count ← 0
23:    sync_offset ← 0
24: end procedure
25:
26: procedure EPOCH_TIMER_EXPIRE
27:    SEND_EPOCH_MESSAGE(epoch)
28: end procedure
29:
30: procedure RX_SYNC_MESSAGE(other_epoch)
31:    now ← time since SYNC_TIMER started
32:    if PERIOD > now & now >= PERIOD/2 then
33:        sync_offset ← now
34:    end if
35:
36:    if other_epoch < epoch − 2 then
37:        SEND_EPOCH_MESSAGE(epoch)
38:    end if
39: end procedure
40:
41: procedure RX_EPOCH_MESSAGE( other_epoch)
42:    if other_epoch > epoch then
43:        epoch ← other_epoch
44:    end if
45: end procedure
```

IV. Evaluation

A. Implementation

A Micro Air Vehicle (MAV) is a small, lightweight flying robot [5]. These can have various designs, from fixed wing robots [23] and flapping wing [24] designs to quad-copters. The Crazyflie 2.1 (Figure 1) is a palm sized MAV that weighs only 27g. The Crazyflie is equipped with a STM32F405 MCU (Microcontroller unit) to handle the main applications, and an nRF51822 MCU to handle radio communications and power management. It has a flight time of 7 minutes, and a maximum recommended payload weight of 15g. The development platform for the Crazyflie is open source, giving flexibility to modify the firmware depending on your application.

The STM32F405 (STM) microprocessor runs the Crazyflie Firmware [25]. This firmware runs the controller which handles state estimation of the Crazyflie, including the sensor information and motors speed control, as well as sending communication packets to the NRF radio MCU. It also runs applications that can be created by the user, which can be used to run the Crazyflie in a decentralised manner without a centralised controller.

The nRF51822 MCU is a System on a Chip (SoC) manufactured by Nordic Semiconductor that provides 2.4 GHz radio protocols, including Bluetooth Low Energy (BLE) and Nordic's proprietary ESB protocol [26]. The Crazyflie uses BLE for control using the IOS and Android applications, and the 2.4GHz radio to send peer to peer communications between drones using the Enhanced ShockBurst (ESB) protocol provided by Nordic.

The implementation of the ESPCO algorithm consisted of first implementing the algorithm on the nRF51822 MCU, ensuring that the drones synchronise and the epochs are shared. Then implementing a method to send this shared epoch to the Crazyflies STM processor to be used for applications, such as coordinated flight.

V. Flight experimentation

To evaluate how well ESPCO and its epoch updates worked for synchronised flight, we ran five flight tests with three drones using the Vicon camera system for positioning [27]. In each test, the drones were left to synchronise for one minute, and then proceeded to fly for one minute, before landing. The `FLIGHT_TIME_S` for how long the Crazyflie will take to fly to the proposed relative `(x, y, z)` was 2 seconds.

The setup before flight can be seen in Figure 1, and the drones mid flight can be see in Figure 2. The drones were placed under 0.5 meters away from each-other. In each of the tests the drones flew at synchronised times without having any collisions caused by drones not reacting to proposed flight times at the same epoch. In one of the flights, one of the drones ended up failing mid-flight, due to it running low on battery, however the other two still flew in synchronisation, demonstrating the robustness to failure of the synchronisation.

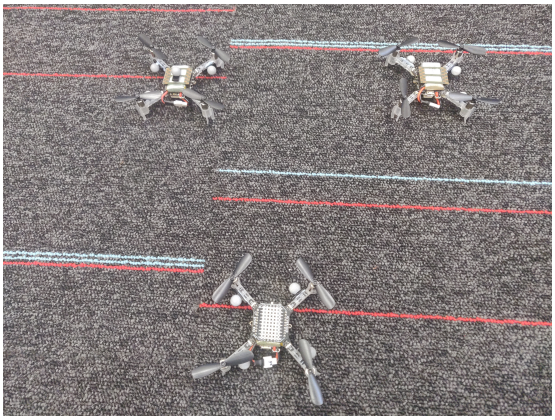

Fig. 1. Drone setup before flight

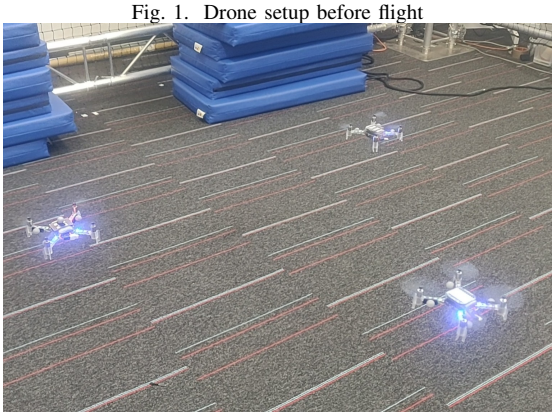

Fig. 2. Drones flying mid experiment in synchronisation.

VI. Conclusion

Overall we can see from the results that the ESPCO algorithm has the following benefits:

- **Decentralised Drone-to-Drone communication:** The ESPCO algorithm does not rely on a single controller or communication path. It uses a mesh all-to-all topology that is resilient to the failure of any individual drone. As long as the network is fully connected all drones will eventually synchronise and receive updates.
- **Generally fast synchronisation:** The ESPCO algorithm synchronises in under a minute. In general the algorithm seems to synchronise much faster when all the drones turn on at the same time. This speed can be adjusted if required by changing the `PERIOD` of the algorithm, with a trade off that faster periods result in more messages sent in the network however would result in faster convergence.
- **Robust Synchronisation:** We have seen above that the synchronisation of ESPCO is very robust, and once drones are synchronised, they do not lose synchronisation for at least 10 minutes as tested. They also do not change much in their synchronisation, with a standard deviation of 0.02ms.
- **Low synchronisation error:** When the drones are synchronised, we can see that they synchronise with an

error of less than 0.1ms. This is certainly sufficient for coordinated flight maneuvers, as well as changing behaviour in unison.

- **Coordinated Epoch Updates:** The synchronisation of epochs in the network is very fast, and in general seems bounded by the synchronisation of the oscillations of the drones.
- **Tolerance for drones entering the network:** The algorithm is tolerant to new drones entering the network, and can converge to a stable oscillation with the new drone included. This does however take more time than when all the drones enter the network separately.
- **Unaffected by drones leaving the network:** We have demonstrated that the networks synchronisation is unaffected when drones leave the network, and in fact may actually improve slightly as a drone that left the network could have had a bigger synchronisation error.
- **Sufficient for planning synchronised behavior:** We have demonstrated in flight tests that the epochs can be used to schedule synchronised flight behaviours, with drones proposing actions that are then performed at the same time.

We have also identified the following drawbacks of the ESPCO algorithm:

- **Indeterminate synchronisation time:** The algorithm does not have a clearly defined time at which it will definitely be synchronised, it is only proven to synchronise in some finite time.
- **Synchronisation time when joining the network:** The synchronisation time seems worse when a drone is joining an already established synchronised network. This could limit the applications for drones leaving and rejoining the network to perform tasks.

REFERENCES

[1] Rinner B, Bettstetter C, Hellwager H, Weiss S. Multidrone Systems: More than the Sum of the Parts. IEEE Computer. 2021 Mar;54(3):185-94.

[2] Radoglou-Grammatikis P, Sarigiannidis P, Lagkas T, Moscholios I. A compilation of UAV applications for precision agriculture. Computer Networks. 2020;172:107148.

[3] Floreano D, Wood RJ. Science, technology and the future of small autonomous drones. Nature. 2015;521(7553):460-6.

[4] Rizk Y, Awad M, Tunstel EW. Cooperative heterogeneous multi-robot systems: A survey. ACM Computing Surveys (CSUR). 2019;52(2):1-31.

[5] Coppola M, McGuire KN, De Wagter C, de Croon GCHE. A Survey on Swarming With Micro Air Vehicles: Fundamental Challenges and Constraints. Frontiers in Robotics and AI. 2020;7. Available from: https://www.frontiersin.org/article/10.3389/frobt.2020.00018.

[6] Schranz M, Umlauft M, Sende M, Elmenreich W. Swarm Robotic Behaviors and Current Applications. Frontiers in Robotics and AI. 2020;7. Available from: https://www.frontiersin.org/article/10.3389/frobt.2020.00036.

[7] Duisterhof BP, Li S, Burgués J, Reddi VJ, de Croon GCHE. Sniffy Bug: A Fully Autonomous Swarm of Gas-Seeking Nano Quadcopters in Cluttered Environments. 2021 7. Available from: http://arxiv.org/abs/2107.05490.

[8] Mcguire K, De Wagter C, Tuyls K, Kappen H, Croon G. Minimal navigation solution for a swarm of tiny flying robots to explore an unknown environment. Science Robotics. 2019 10;4:eaaw9710.

[9] Mirollo RE, Strogatz SH. Synchronization of Pulse-Coupled Biological Oscillators. SIAM Journal on Applied Mathematics. 1990;50(6):1645-62. Available from: https://doi.org/10.1137/0150098.

[10] Lucarelli D, Wang IJ. Decentralized Synchronization Protocols with Nearest Neighbor Communication. In: Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems. SenSys '04. New York, NY, USA: Association for Computing Machinery; 2004. p. 62–68. Available from: https://doi.org/10.1145/1031495.1031503.

[11] Chen CC. Threshold effects on synchronization of pulse-coupled oscillators. Phys Rev E. 1994 Apr;49:2668-72. Available from: https://link.aps.org/doi/10.1103/PhysRevE.49.2668.

[12] Hong YW, Scaglione A. Time synchronization and reach-back communications with pulse-coupled oscillators for UWB wireless ad hoc networks; 2003. p. 190 194.

[13] Breza M, McCann J. Polite Broadcast Gossip for IOT Configuration Management; 2017. p. 1-6.

[14] Levis P, Clausen T, Hui J, Gnawali O, Ko J. The trickle algorithm; 2011.

[15] Webster M, Breza M, Dixon C, Fisher M, McCann J. Formal Verification of Synchronisation, Gossip and Environmental Effects for Wireless Sensor Networks. Electronic Communications of the EASST Volume. 2019;076.

[16] Schilcher U, Schmidt JF, Vogell A, Bettstetter C. Swarmalators with Stochastic Coupling and Memory. In: 2021 IEEE International Conference on Autonomic Computing and Self-Organizing Systems (ACSOS); 2021. p. 90-9.

[17] Barcis A, Bettstetter C. Sandsbots: Robots That Sync and Swarm. IEEE Access. 2020;8:218752-64.

[18] Perez Diaz F, Zillmer R, Groß R. Firefly-Inspired Synchronization in Swarms of Mobile Agents. vol. 1; 2015. .

[19] Alsolami F, Alqurashi F, Hasan M, Saeed R, Abdel-Khalek S, Ishak A. Development of Self-Synchronized Drones' Network Using Cluster-Based Swarm Intelligence Approach. IEEE Access. 2021 03;PP:1-1.

[20] Christensen AL, OGrady R, Dorigo M. From Fireflies to Fault-Tolerant Swarms of Robots. IEEE Transactions on Evolutionary Computation. 2009;13(4):754-66.

[21] Werner-Allen G, Tewari G, Patel A, Welsh M, Nagpal R. ABSTRACT Firefly-Inspired Sensor Network Synchronicity with Realistic Radio Effects; 2005. p. 142-53.

[22] Levis P, Patel N, Culler D, Shenker S. Trickle: A self-regulating algorithm for code maintenance and propagation in wireless sensor networks; 2004. .

[23] Green WE, Oh PY. Autonomous hovering of a fixed-wing micro air vehicle. IEEE; 2006. p. 2164-9. Available from: http://ieeexplore.ieee.org/document/1642024/.

[24] de Croon GCHE, Perçin M, Remes BDW, Ruijsink R, Wagter CD. The DelFly. Springer Netherlands; 2016.

[25] Crazyflie Firmware. Bitcraze;. Accessed 20/06/2022. Available from: https://github.com/bitcraze/crazyflie-firmware.

[26] nRF51822 - System on Chip - Bluetooth Low Energy and 2.4 GHz SoC. Nordic Semiconductor;. Accessed 20/06/2022. Available from: https://www.nordicsemi.com/products/nrf51822.

[27] TRACKER. Vicon;. Accessed 20/06/2022. Available from: https://www.vicon.com/software/tracker/.