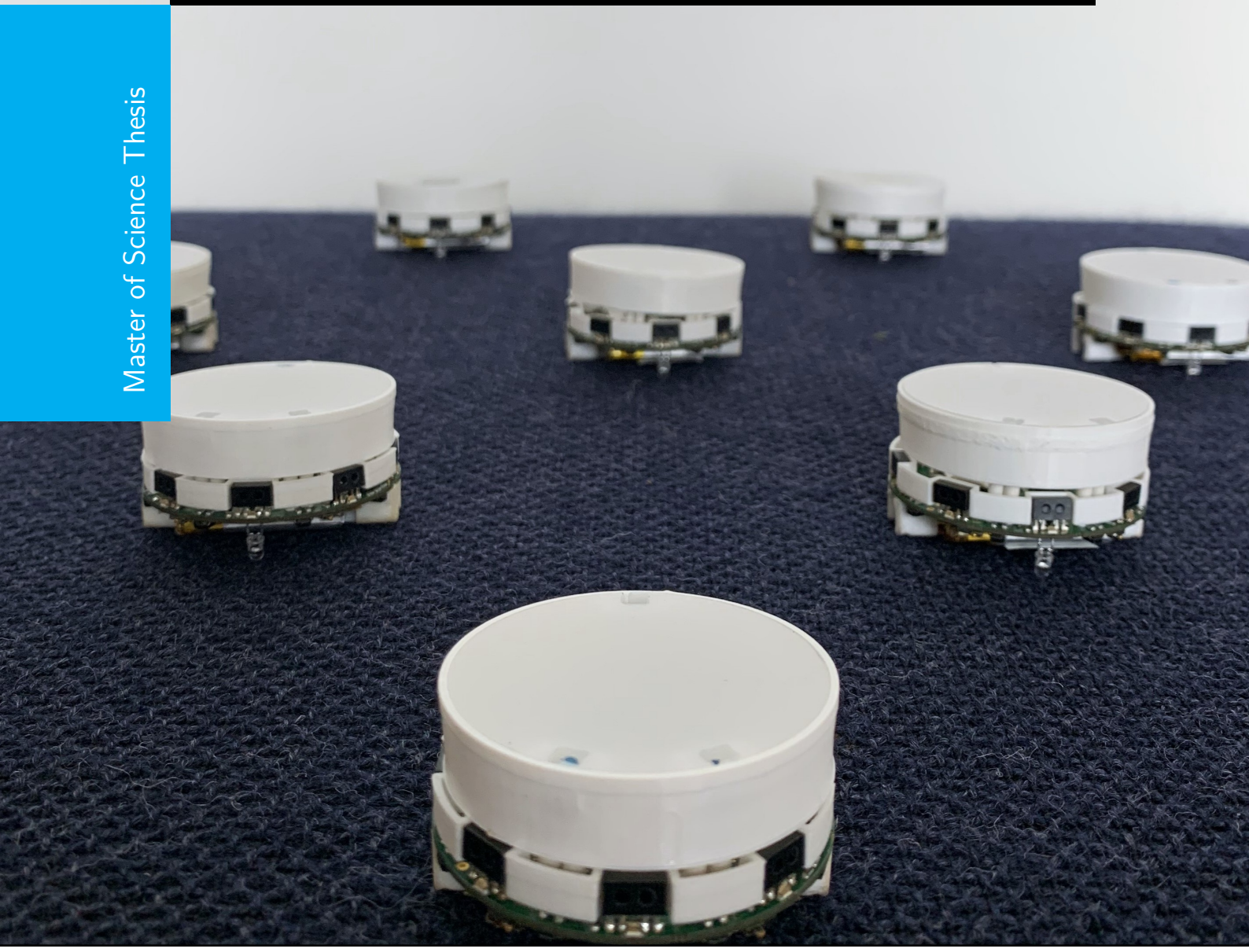
Consensus algorithms for Single-Integrator Multi-Agent Systems

Eva Zwetsloot

Master of Science Thesis



TU Delft Delft University of Technology

Delft Center for Systems and Control

Consensus algorithms for Single-Integrator Multi-Agent Systems

Master of Science Thesis

For the degree of Master of Science in Embedded Systems at Delft University of Technology

Eva Zwetsloot

May 18, 2023

Faculty of Electrical Engineering, Mathematics, and Computer Science · Delft University of Technology

Delft University of Technology
Department of
Delft Center for Systems and Control (dcsc)

The undersigned hereby certify that they have read and recommend to the Faculty of Electrical Engineering, Mathematics, and Computer Science for acceptance a thesis entitled

Consensus algorithms for Single-Integrator Multi-Agent Systems

by

Eva Zwetsloot

in partial fulfillment of the requirements for the degree of

Master of Science Embedded Systems

Dated: May 18, 2023

Supervisor(s):

dr.ir. Manuel Mazo Jr.

Reader(s):

dr.ir. Raj Thilak Rajan

dr.ir. Dimitris Boskos

Abstract

In multi-agent systems reaching consensus has been a long-standing problem. A considerable amount of research has been focused on how event-triggered consensus (ETC) can be used to limit the energy consumption of the communication system while still ensuring convergence to the neighbourhood of a point or a formation. These algorithms rely on having knowledge of the global state information of each agent in the network. Given this information, it is possible to consider using distributed algorithms from the field of Computer Science to achieve the same.

This thesis presents a study on average consensus algorithms for single-integrator multi-agent systems. The focus is on comparing the time required to reach convergence and the energy required by the communication system. To perform this study, three methods are used to compare the algorithms. The first method involves deriving mathematical bounds on the convergence time and the number of transmissions. The second involves a simple simulation to verify the mathematical bounds. Finally, the algorithms are compared in a practical application where robots are subjected to noisy measurements and need to assemble in a formation.

The results of the comparisons show that distributed algorithms are capable of fast convergence and require less energy for communication. Therefore, it is worth considering the use of distributed algorithms when agents have enough memory to temporarily store the global state of all agents in the network. There might still be situations in which the ETC algorithm converges slightly faster but this benefit is overshadowed by the increase in energy consumption.

Table of Contents

List of Figures

List of Tables

Acknowledgements

First of all, I would like to thank my supervisor prof. dr. ir. Manuel Mazo Jr. for his guidance during the writing of this thesis and for coming up with an interesting thesis topic that was fun to work on. I am also grateful for the interesting weekly group meetings that provided me with new insights into various topics.

Next, I would like to thank my family for their continued support throughout my studies. The love and encouragement offered by my parents helped me overcome challenges and kept me motivated throughout the years.

I would also like to thank my friends for bringing me joy and laughter throughout my studies. Their support has made this experience unforgettable.

Finally, I would like to thank the members of the examination committee for offering their time and interest in my thesis topic.

Delft, University of Technology
May 18, 2023

Eva Zwetsloot

"It always seems impossible until it's done"
— *Nelson Mandela*

Chapter 1

Introduction

Multi-agent systems (MAS) have received much attention in recent years [12, 33, 73]. Mainly due to the unique advantages they have over a single centralised, highly intelligent system. To keep the cost to a minimum the individual systems have limited computational, sensing, and communication capabilities. However, through cooperative control, these agents are able to achieve higher-level tasks making them suitable for a wide range of applications in diverse fields, including robotics, autonomous vehicles, swarm intelligence, sensor networks, and air traffic control.

Given the wide range of applications, cooperative control in MAS has brought about numerous research opportunities. One area of study focuses on achieving consensus, where the goal is to ensure a group of agents or processes converges to the neighbourhood of a common decision or value. The design of these algorithms is influenced by the type of system, communication network, and application domain. As such, a variety of algorithms have been developed to guarantee convergence for different situations.

Around the same time, the idea of event-triggered control regained popularity. It had been applied in the 1960s to address the issue of controlling a continuous-time system using a digital controller, but it regained interest in the last decade due to the increasing interest in distributed systems. The main idea behind event-triggered control is to decrease the frequency of control updates by executing a control task only when a certain event has occurred. The occurrence of such events is determined by an event-triggering mechanism that ensures the control task is only executed when necessary. In this way, event-triggered control can reduce the computational and network loads, resulting in a lower energy consumption of the (wireless) system.

Naturally, event-triggered control can also be used to limit the number of transmissions and reduce the energy consumption in MAS. The combination of the two ideas of cooperative control and event-triggered introduces many new challenges that do not exist in either area alone [86]. Most of these challenges are addressed with a focus on solving consensus, also referred to as event-triggered consensus (ETC) problems. Consensus is usually taken as the research focus since different problems that involve MAS happen to be closely related, such as rendezvous, flocking, synchronisation, sensor fusion, and formation control [90].

In addition to the field of Systems and Control, achieving consensus has also been an area of research in Computer Science, as it serves the basis for distributed computing [107, 71]. Multiple distributed algorithms have been designed to address this problem for different applications. In contrast to ETC, these algorithms often only use a finite number of transmission to exchange the necessary information between agents and to reach a common decision [107].

1-1 Problem Statement

Implementing event-triggered control in MAS to solve the consensus problem has been and is still being investigated extensively. In order to achieve consensus, agents are required to communicate either global or relative state information at specific moments in time. To the best of our knowledge, all related research papers assume the availability of this state information, therefore, forcing strict measurement requirements.

Due to the availability of this information using a distributed algorithm to converge in finite time to the final consensus state is also a possibility. A comparison between these methods for solving consensus has so far not been made.

Therefore, this thesis will challenge; *whether the knowledge of global or relative state measurements undermines the benefit of using (periodic) event-triggered control over using a distributed algorithm to solve the consensus problem in a MAS.*

Due to the existence of various event-triggered algorithms tailored to different types of applications, the focus of this thesis will be limited to robotic swarms with single-integrator agents.

1-2 Thesis Outline

This thesis is structured as follows; in Chapter 2 related works relevant to this thesis are presented. Chapter 3 presents the required preliminary knowledge. Subsequently, Chapter 4 gives a more detailed description of the problem statement using the knowledge obtained in the previous chapters. Hereafter, Chapter 5 compares the algorithms based on mathematical derivations, Chapter 6 compares the algorithms using a simplified simulation, and Chapter 7 compares the algorithms based on a realistic simulation and an actual implementation. Finally, conclusions and suggestions for future work will be given in Chapter 8.

As a last remark, the code used for the implementation of the algorithms in Chapter 6 and 7 can be found at https://github.com/EvaZwetsloot/consensus_algorithms.

Chapter 2

Related Work

This chapter provides an overview of the evolution of consensus algorithms. Initially, the focus was on the development of algorithms that solve consensus problems for various applications. Later, the combination of event-triggered control and consensus gained interest. Section 2-1, therefore focuses on the development of consensus algorithms, while section 2-2 , discusses event-triggered consensus algorithms.

2-1 Consensus in control

The work of Borkar and Varaiya [11], Tsitsiklis [110] and Tsitskilis, Bertsekas and Athans [109] was a starting point of networked consensus problems in the field of Systems and Control. The objective of this research, as well as of consensus in general, is to design a protocol for a group of agents, connected by a communication network, that ensures the agents reach an agreement on a certain quantity of interest based only on the local information of each agents [115, 95, 69, 3]. A significant contribution to solving and proving the convergence of consensus problems was the introduction of graph theory [25, 38, 76]. Specifically, the introduction of graph Laplacians and their spectral properties.

Using graph theory and previous work from Fax and Murray [39], Olfati-Saber and Murray created a theoretical framework for posing and solving consensus problems in networked multi-agent systems (MAS) [89]. Their findings demonstrate that in the presence of communication delays, a system with single-integrator dynamics and a strongly connected balanced network topology will asymptotically converge to the average of the initial conditions. Around the same time, other key papers that played a significant role in advancing the field of network systems were published [90]. Ren and Beard [96] together with Moreau [81] demonstrated that consensus can be achieved in applications with dynamically changing network topologies. While, Fax and Murray [39] showed that consensus protocols can also be applied to solve formation control problems by including a bias.

These papers illustrate that the design of a consensus algorithm for multi-agents is influenced by three main factors; the dynamics of the agents, the network topology, and the objective.

These three factors introduce various challenges and sparked a wide variety of research, which is still extremely active [86].

2-1-1 Dynamics of agents

The system dynamics greatly influence the final consensus value of MAS. Early research on consensus problems mainly focus on single-integrator or double-integrator dynamics. Single-integrator agents converge to a constant value, which is usually the (weighted) average of the initial conditions [132, 15]. On the other hand, double-integrator agents are defined by two states of which one converges to a constant final value while the other converges to a trajectory [93, 94, 100].

However, in practical MAS, agents are more complex than single or double integrators. Consequently, consensus problems with general linear dynamics were studied (e.g. [103, 111, 98, 123, 99, 131]), where consensus algorithms ensure the states of the agents converge to the same trajectory.

This work was followed by further research into nonlinear systems, which includes systems with complex networks (e.g. [63, 60, 124, 126, 125]), nonholonomic dynamics (e.g. [26, 30]), and rigid body dynamics (e.g. [97]). An often-made assumption in these papers is that the dynamics of the agents are homogeneous and known, which indicates that the agents all have the same known system dynamics. However, real-world systems are often subject to uncertainties and variations, resulting in heterogeneous agents with different dynamics or models. As a consequence, research has also been focused on developing consensus control strategies specifically designed for such systems [132]. Studies, such as [31, 101], have investigated various approaches for achieving consensus in heterogeneous MAS.

Even though, system dynamics have a great effect on the consensus algorithm. Single-integrator or double-integrator dynamics are often still assumed for agents in research papers that also incorporate event-triggered control as they are useful to demonstrate certain ideas without having to consider the additional complications that arise with complex agents [86].

2-1-2 Network topology

The network topology has a significant influence on the design of consensus algorithms. The focus here lies more on finding necessary and/or sufficient conditions such that consensus can be achieved [15]. A network topology can be characterised by several aspects, such as whether it is deterministic or stochastic, time-varying or time-invariant, directed or undirected, and balanced or unbalanced.

Much research has been done on finding the conditions for deterministic network topologies [132], including works by Olfati-Saber and Murray [89], Shi and Hu [101], and Mishra and Ishii [78]. It is reasonable to assume a deterministic topology when considering an ideal communication network. The stochastic network setting is however important when considering uncertainties such as random packet dropout, unknown/uncontrolled states, and random communication delays. Hatano and Mesbahi [52] were one of the first to study the effect stochastic networks have on reaching consensus. A significant difference compared to a deterministic

network is that consensus is achieved when agents almost surely reach an agreement on the final state while in a deterministic network setting they surely reach an agreement. Continuing on the work of Hatano and Mesbahi, Wu [116] and Tahbaz-Salehi [106] provided conditions for more general linear systems. Naturally, consensus problems for double-integrator dynamical systems and more complex systems were also researched (e.g. [129]).

In case of temporary communication obstruction or moving agents, the network topology might not be fixed in time (time-invariant) but the topology switches (time-varying). Olfati-Saber and Murray [89] created the theoretical framework for a switching network topology. They imposed that the network had to be balanced and connected in order to ensure average consensus in both fixed and switching network topologies. This rather strict requirement was further studied and relaxed by Ren and Beard [96] and Moreau [81]. Ren and Beard prove that as long as the union of the interaction graphs includes a spanning tree frequently enough consensus is reached for directed and undirected networks.

Olfati-Saber and Murray [89] also discuss consensus conditions for both directed and undirected network topologies. They show that directed network topologies require more stringent conditions in order to reach consensus. Connected undirected networks will asymptotically converge to the average consensus, which is only the case for directed networks when they are strongly connected and balanced.

In most practical robotic MAS, the agents are capable of two-way communication, resulting in an undirected topology. Initially, the agents form a connected network topology that may become disconnected or may switch over time. However, as Ren and Beard [96] proved if the union of the network graphs is connected over time, the system will eventually converge.

2-1-3 Objectives

As stated in Chapter 1, consensus problems occur in various applications, resulting in a broad range of research with slightly different objectives. This section first highlights how consensus protocols can be used to solve different cooperative control problems. Hereafter, two factors that are relevant to this thesis will be discussed that characterise different consensus algorithms.

There are many factors that impact consensus algorithms, most of these will not be discussed but are worth mentioning as they have gained increasing interest over the years [15], examples include quantisation [43], asynchronous effects [117], and sampled-data frameworks [120].

Cooperative control problems

Formation control describes the problem where multiple agents are required to assemble in a desired formation or pattern by maintaining a specific relative distance from other agents [15]. As previously mentioned, Fax and Murray [39] argued that formation control could be reformulated into a consensus problem. Essentially, a formation control problem requires agents to agree on the centre of the formation, a task that can be achieved by means of a consensus protocol. The formation itself is maintained by adding a bias to required relative displacement between agents for the formation.

Consensus algorithms are also required to address the flocking problem [90]. The goal of flocking problems is to maintain a specific distance between agents, avoid collisions, and

maintain a similar velocity between agents. Olfati-saber [88] proposed a consensus-based framework for the design and analysis of flocking algorithms, here consensus algorithms are required to ensure velocity matching between agents.

Finally, another problem that makes use of consensus protocols is rendezvous. Rendezvous requires agents to agree on a common meeting location [20]. Take for instance, the challenge of two spacecraft that need to connect for a resupplying mission. This docking process requires the agents to match each other velocities and meet at the same location. Due to the nature of both problems, protocols designed for solving consensus can be adapted to solve the rendezvous problem.

Each of these problems requires a different variation of a consensus protocol. These protocols are further influenced by the presence of a leader, multiple leaders, or no leaders. Leader algorithms, (e.g. [16, 102]), use centralised control where one or more leaders guide the agents to one decision. In contrast, leaderless algorithms, (e.g. [51, 6]), are decentralised algorithms where each agent receives information from its neighbours and makes its own decision. Leader algorithms have the advantage of having on central point with all the information, resulting in more efficient decision-making and a simpler implementation. However, these algorithms rely heavily on the leaders, making them less flexible and less robust to failure. This thesis will primarily focus on leaderless consensus algorithms, although the results obtained can also be extended to leader algorithms.

Time delays

Most practical systems experience time delays due to various factors which include limited communication speed, the time required to obtain measurement information from sensors, the computation time required to generate control signals, and the time between updating the control input and observing an actual response. Time delays might degrade the system performance and even cause instability, therefore a considerable amount of research has focused on the conditions under which convergence is still guaranteed [4].

In literature, two types of delays have been considered, communication time delay (CTD) and input time delay (ITD). ITD is used to refer to the processing time of the data at each agent [108]. While CTD refers to the delay caused by the communication from one agent to another. In actuality, these two types of delays impact the system at the same time [108, 130].

Olfati-Saber and Murray [89], studied the situation where agents, with first-order linear dynamics, experienced a constant CTD. To address this problem, they proposed that each agent intentionally introduces an input delay equal to the CTD to its own state measurements in order for it to synchronise with the incoming data of its neighbours.

Additional research has analysed the effect of time-varying delays (e.g. [108, 104, 82, 66, 130, 9]) and the impact of time delays on more complex systems; double integrator dynamics (e.g. [63, 121]), and higher-order dynamics (e.g. [130, 4]).

Convergence

Consensus algorithms guarantee asymptotic convergence, but the convergence rate is an important aspect that affects the system performance. The algebraic connectivity of the network

directly influences the convergence rate, with a higher algebraic connectivity resulting in faster convergence [89].

Research has been done to increase the algebraic connectivity in specific network topologies either, by changing the weight of the communication topology (e.g. [59, 119]), by changing the topologies (e.g. [87, 53]) or by intentionally delaying the states (e.g. [40, 80]). In some problems, it is required that consensus is reached in a finite time. This sparked research into finite-time consensus (FTC) (e.g. [113, 127, 65, 49, 118, 21]).

In this thesis, we consider a robotic application where it suffices to convergence in finite time to a neighbourhood around the consensus point. Therefore, FTC algorithms are not required.

2-2 Event-triggered consensus

Event-triggered control systems, often also referred to as event-based control systems, were already researched in the 1960s and 1970s [32]. However, time-triggered control still dominated, due to the difficulty of developing system theory for event-based systems and the strong assumption that systems could guarantee deterministic sampling times [133]. In time more works started to compare time-triggered to event-triggered and self-triggered sampling techniques (e.g. [92, 7, 105, 133, 5, 72, 114, 1]).

A strong motivation for the rebound of event-triggered ideas can most likely be contributed to the increasing interest in cyber-physical systems (CPS) and Internet of Things (IoT). As in these systems, the focus lies on keeping the information exchange to a minimum by only communicating to neighbours when it is valuable as this improves the scalability, robustness to faults, and limits the use of resources [24]. Limiting the use of resources is crucial to limit energy consumption and reduce wireless communication congestion [86].

The main challenge in the design for event-triggered consensus is designing a triggering function that guarantees convergence, liveness and the absence of deadlocks [105]. Ensuring liveness and the absence of deadlocks is mostly done by ruling out Zeno behaviour. Although, later a stricter requirement followed namely ensuring the existence of a positive minimum inter-event time (MIET) [10] (discussed in more detail in Chapter 3).

Early works for MAS, focus on centralised event-triggered consensus (ETC) with simple dynamics (e.g. [105, 27, 29, 28, 68]). In these works, a triggering condition is designed that depends on the state of all agents.

Consequently, this was extended to decentralised systems (e.g. [27, 29, 28, 100, 68]), where the triggering function only depends on the states of close neighbours. However, it should be mentioned that the triggering functions often depend on variables, such as the convergence rate, that require global information, making these implementations not fully distributed [17].

Furthermore, these earlier works often fail to rule out Zeno behaviour. For instance, Dimarogonas [28] proposes a triggering function that guarantees convergence only if there exists at least one agent with a strictly positive inter-event time. However, this needs to hold for all agents in order to fully exclude Zeno behaviour for all trajectories [10]. Consequently, this implementation has the additional requirement that convergence is only guaranteed for non-Zeno trajectories.

These works all illustrate that the design of a triggering function can be influenced by 5 factors; system dynamics, network topology, trigger response, event detection and trigger dependence [86].

2-2-1 Trigger response

Trigger response refers to the action an agent will take once a triggering function is violated. In research, the distinction is made between three types of responses: event-triggered control, event-triggered communication and event-triggered coordination. The first involves updating the control action only when an event is triggered. The second type involves broadcasting the state information once an event is triggered. Lastly, event-triggered coordination is a combination of the first two.

Consensus algorithms typically rely on event-triggered coordination to limit the communication load in a network. Without it, agents are required to continuously monitor the states of neighbouring agents. This is impractical in robotic MAS, as it would require an unlimited network bandwidth and energy source [58]. Hence, this thesis focuses on event-triggered coordination for event-triggered consensus algorithms.

Event-triggered communication can further be distinguished by considering the type of information exchange. Either the agent broadcasts its own information to its neighbours, pulls state information from its neighbours (e.g. [37, 84]) or performs an information exchange with its neighbours (e.g. [13]).

2-2-2 Event detection

Event detection refers to the way in which the triggering function is evaluated either continuously, periodically or periodically. Continuous monitoring, (e.g. [101, 100, 89, 27]), is possible if tasks can be executed concurrently on the agents. However, due to the typically low cost of the agents, it is unlikely that parallel processing is an option.

Apart from this practical aspect, it is also harder to exclude Zeno behaviour for continuous systems. Therefore, a benefit follows from having discrete times at which the triggering function is evaluated. Motivated by this discussion, research was conducted into periodic (e.g. [83, 85, 42, 112, 74]) and aperiodic detection (e.g. [34, 67, 54, 18, 61, 17]). Periodic detection schemes check the triggering function every period. In between these triggering instances the agents are operating under open-loop control. Therefore, there exist strict requirements for the sampling period and the magnitude of disturbances, such as the communication delay, in order to ensure stability.

Most aperiodic detection schemes use self-triggered control, in these algorithms the next triggering instance is precomputed using the past and current state information. Aperiodic detection schemes also operate under open-loop control in between triggering instances. Therefore, they experience the same problem as periodic detection schemes and require a maximum bound on the inter-event time in order to ensure stability.

2-2-3 Trigger dependence

Triggering functions can be distinguished based on the parameters it depends on. Generally, there are two overarching categories; static and dynamic. In static triggering functions, the event times only depend on currently available information. Dynamic triggering functions, on the other hand, rely on dynamic variables that represent the state of the system and its evolution over time. These dynamic triggering functions were introduced to reduce the number of triggering events further. Some of the first work on dynamic triggers was presented by Girard [47].

A further classification of static triggers has been made by Borgers and Heemels [10], they classify three classes of static triggers; relative (e.g. [36, 85, 43, 105]), absolute (e.g. [122, 100]) and mixed. Relative static triggers use a state-dependent function, absolute triggers include a constant threshold, and mixed triggers are composed of a combination of the two.

2-3 Distributed consensus

Distributed algorithms have been and continue to be a significant area of research in Computer Science as they allow for increased computing power by having multiple systems working together. The development of distributed algorithms was first motivated by the need to exchange data between different computers. For instance, to allow different universities to exchange information [71].

As computer technology advanced, more complex algorithms were developed to enable various applications, including parallel processing, distributed databases, cloud computing, and blockchain technology.

The development of distributed algorithms involves addressing common issues such as broadcasting information, achieving global synchronisation between processes, and triggering the execution of an event in each process [107]. In order to complete these tasks, messages need to be passed between all processes in the system. To achieve this, multiple message-passing schemes were developed, often referred to as Wave algorithms.

Consensus algorithms are essentially message-passing schemes that ensure an agreement is reached on a specific value. Therefore, the consensus problem can also be addressed by using a Wave algorithm.

Chapter 3

Preliminaries

3-1 Notations

The notation $\mathbb{R}$ is used to denote the set of real numbers, $\mathbb{Z}$ represent the set of integer numbers, $\mathbb{N}$ denotes the set of natural numbers, and $Z = \mathbb{R}^n \times L_2([-\tau, 0]; \mathbb{R}^n)$ is used to refer to the Hilbert space with L_2 the space of square Lebesgue integrable functions from $[-\tau, 0]$ to $\mathbb{R}^n$.

The euclidean norm for vectors or the induced two-norm for matrices is denoted with $\|\cdot\|$. Furthermore, the infinity norm of vector is represented by $\|\cdot\|_\infty = \max_i |x_i|$.

Given a matrix M, M^T denotes the transpose of M, and $M > 0$ (or $M \geq 0$) means that M is positive definite (or semi-positive definite). To represent the minimum and maximum eigenvalues of a matrix, the notations $\lambda_{min}(M)$ and $\lambda_{max}(M)$ will be used respectively.

A function $\gamma : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ belongs to the $\mathcal{K}$ class when it is strictly increasing and $\gamma(0) = 0$. A function $\beta : \mathbb{R}_{\geq 0} \times \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ belongs to the $\mathcal{KL}$ class if for each fixed $s \geq 0$, the function $\beta(r, s)$ is a $\mathcal{K}$-function and for each fixed $r \geq 0$, the function $\beta(r, s)$ is decreasing with respect to s and $\beta(r, s) \to 0$ as $s \to 0$.

3-2 Graph theory

Consider a graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ consisting of a set of nodes $\mathcal{V} = \{1, ..., N\}$ and edges $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$. When there is an edge (i, j) between the nodes i and j, then i and j are called adjacent. The graph $\mathcal{G}$ is called *undirected* when $(i, j) \in \mathcal{E} \Leftrightarrow (j, i) \in \mathcal{E}$, otherwise the graph is called *directed*. When there exists an edge between two nodes i and j, $(i, j) \in \mathcal{E}$, the nodes are referred to as *neighbours*. The set of neighbours of a given node i is denoted by $\mathcal{N}_i$.

The adjacency matrix $\mathrm{A} \in \mathbb{R}^{N \times N}$ indicates whether pairs of nodes are neighbours. If and only if node i and j are neighbours will the matrix entry, a_{ij}, be nonzero. In all other cases, the matrix entry, a_{ij}, is zero. A path from i to j is a sequence of distinct nodes starting from

i and ending with j such that each pair of consecutive nodes is adjacent. If there is a path from i to j, then i and j are connected. If all pairs of nodes in the graph $\mathcal{G}$ are connected then the graph $\mathcal{G}$ is *connected*. A *strongly connected* graph, is a graph where there exists a directed path between any two nodes. In other words, there is a path from node i to j and a path from j to i. Connected undirected graphs are always strongly connected.

A *tree* in an undirected graph, is a subgraph in which every pair of nodes is connected by exactly one path. A *spanning tree* of an undirected graph is a subgraph that is a tree containing all the edges of the graph.

The distance $d(i,j)$ between two nodes is the number of edges of the shortest path from i to j. The diameter D of graph $\mathcal{G}$ is the length of the shortest path between the most distanced nodes, in other words, it is the maximum distance, $d(i,j)$ over all nodes. The degree matrix $\overline{D}$, is a diagonal matrix with elements d_i equal to the cardinality of node i's neighbour set, $|\mathcal{N}_i|$.

The Laplacian matrix L of a graph $\mathcal{G}$ is defined as $L = \overline{D}$ - A. In undirected graphs, the Laplacian matrix L is both symmetric and positive semi-definite, expressed as $L = L^T \geq 0$. The sum of each row in L is equal to zero, and as a result, the eigenvector corresponding to the eigenvalue zero is the vector of ones, denoted by $\mathbf{1}$. For strongly connected graphs, the Laplacian has exactly one eigenvalue equal to zero with the rest strictly positive, $0 = \lambda_{min}(L) < \lambda_2(L) \leq ... \leq \lambda_{max}(L)$. The magnitude of the second smallest eigenvalue, $\lambda_2(L)$ is often referred to as *algebraic connectivity* and reflects how well-connected a graph is [41]. The algebraic connectivity of a graph is not always known. In general, Theorem 3.1, can provide an upper and lower bound on the algebraic connectivity.

Theorem 3.1 (Algebraic connectivity [41])**.** *Let $\mathcal{G}$ be a connected graph with a minimum degree $\delta = \min_i d_i$. Then:*

$$2\delta - N + 2 \leq \lambda_2(\mathcal{G}) \leq \frac{N}{N-1}\delta \tag{3-1}$$

where N refers to the number of nodes in graph $\mathcal{G}$.

The largest eigenvalue, also referred to as the *spectral radius*, is an often-used parameter in event-triggered consensus algorithms. For simple graphs, graphs without loops or multiple edges, it can be upper and lower bound by Theorem 3.2 and 3.3 respectively.

Theorem 3.2 (Lower bound on spectral radius [128, 48])**.** *Let $\mathcal{G}$ be a simple connected graph with at least one edge and the maximum degree, $\Delta = \max_i d_i$. Then:*

$$\lambda_{max}(\mathcal{G}) \geq \Delta + 1 \tag{3-2}$$

with equality if and only if there exists a node that is adjacent to all other nodes in $\mathcal{G}$

Theorem 3.3 (Upper bound on the spectral radius [75])**.** *Let $\mathcal{G}$ be a simple graph and the maximum degree, $\Delta = \max_i d_i$. Then:*

$$\lambda_{max}(\mathcal{G}) \leq \max\{d_u + d_v | (u,v) \in \mathcal{E}\} \leq 2\Delta \tag{3-3}$$

3-3 Basic idea of event-triggered consensus

In this section, we present the mathematical background for solving the consensus algorithms with event-triggering mechanisms. These algorithms can be used in applications with N homogeneous single-integrator agents connected via an undirected and connected communication network.

3-3-1 Consensus algorithm

The dynamics of a single-integrator is described by

$$\dot{x}_i(t) = u_i(t), \quad i \in \{1, ..., N\} \tag{3-4}$$

Here $u_i(t)$ denotes the control input and is also referred to as the consensus protocol or algorithm. Furthermore, in this thesis we assume the agents operate in a two-dimensional plane therefore $x_i(t) \in \mathbb{R}^2$ and $u_i(t) \in \mathbb{R}^2$. The most common continuous-time consensus algorithm is given by:

$$u_i(t) = -\sum_{j=1}^{n} a_{ij}[x_i(t) - x_j(t)], \quad i \in \{1, ..., N\} \tag{3-5}$$

This closed-loop system can be rewritten by using the Laplacian to

$$\dot{x}(t) = -Lx(t) \tag{3-6}$$

Where $x(t) = [x_1(t), x_2(t), ..., x_N(t)]$. Consensus is achieved if, for all initial states $x_i(0)$ and all $i, j \in \{1, ..., N\}$, it holds $|x_i(t) - x_j(t)| \to 0$ as $t \to \infty$. It is well-known that under these circumstances and given a connected, undirected graph the system asymptotically achieves average consensus [89].

Theorem 3.4 (Average consensus [89])**.** *Given a connected, undirected graph $\mathcal{G}$ and single-integrator dynamics 3-4, if all agents implement the control law 3-5, then the system asymptotically achieves multi-agent average consensus;*

$$\lim_{t\to\infty} x_i(t) = \frac{1}{N}\sum_{j=1}^{N} x_j(0) \quad \forall i \in \{0, ..., N\} \tag{3-7}$$

3-3-2 Distributed event-triggered consensus

Event-triggered consensus ensures that continuous monitoring of the state of the neighbours is no longer necessary. Each agent only has access to the last broadcasted state: $\hat{x}_i(t) = x_i(t_k^i)\ \forall i \in \mathcal{N}$, where t_k^i represents the time the state was last communicated. This transforms the closed loop system in equation (3-6) to:

$$\dot{x}(t) = -L\hat{x}(t) = -Lx(t) - Le(t) \tag{3-8}$$

Where $e(t) = [e_1(t), e_2(t), ..., e_N(t)]$ represents the measurement error vector, with $e_i(t) = \hat{x}_i(t) - x_i(t)$ for $i \in \{1, ..., N\}$.

To ensure stability, a triggering function, $f_i(\cdot)$, is designed to determine when the agent should communicate its state and update its broadcasted state, $\hat{x}_i(t)$. A triggering event happens as soon as the condition:

$$f_i(t, x_i(t), \hat{x}_i(t)) > 0 \tag{3-9}$$

is fulfilled. Therefore, the event times of agent i are defined by $t^i_{k+1} = \inf\{t : t > t^i_k, f_i(t, x_i(t), \hat{x}_i(t)) > 0\}$, with t^i_0 the first triggering instance. The triggering function must not only guarantee stability but should also ensure that Zeno behaviour is excluded for all possible initial conditions.

Definition 3.1 (Zeno behaviour [86]). *Given the closed-loop dynamics, $\dot{x} = F(x, k(\hat{x}))$, with control dynamics, $u(t) = k(\hat{x}(t))$, driven by $t_{k+1} = \inf\{t : t > t_k, f_i(t, x_i(t), \hat{x}_i(t)) > 0\}$ a solution with initial conditions $x(0) = x_0$ exhibits Zeno behaviour if there exists a $T > 0$ such that $t_k \leq T$ for all $k \in \mathbb{Z}^+$.*

In other words, if the triggering function demands that an infinite number of events happen in a finite time, the solution exhibits Zeno behaviour. Different initial conditions result in different solutions, each of these solutions must not exhibit Zeno behaviour. A strict condition for ruling out Zeno behaviour is ensuring a positive minimum inter-event time (MIET) exists. This is also more appropriate for practical systems, as these can only operate at a maximum frequency.

Definition 3.2 (Positive minimum inter-event time [10]). *The minimum inter-event time, τ^{min}, is the time between two consecutive events and must be positively bounded:*

$$t_{k+1} - t_k \geq \tau^{min} > 0 \tag{3-10}$$

As discussed in Chapter 2, many triggering functions have been proposed. A common design practice, originating from [105], for finding a proper triggering function is to ensure the system is input-to-state practically stable (ISpS) with respect to an external disturbance, $e(t)$ [10].

Definition 3.3 (input-to-state practically stable [56]). *The system 3-8 is input-to-state practically stable if there exists a $\mathcal{KL}$-function β, a $\mathcal{K}$-function γ and a constant $d \in \mathbb{R}_{\geq 0}$ such that for each essentially bounded input $e \in \mathbb{R}^N$ and for each $x_0 \in \mathbb{R}^N$ it holds that*

$$||x(t, x_0, e)|| \leq \beta(||x_0||, t) + \gamma(||e||_\infty) + d \tag{3-11}$$

for each $t \in \mathbb{R}_{\geq 0}$. If the ISpS property holds with $d = 0$, then the system is input-to-state stable (ISS).

Ensuring ISpS leads to a triggering function that enforces [10]:

$$\rho(||e(t)||) \leq \sigma||x(t)|| + \beta \tag{3-12}$$

with ρ a $\mathcal{K}$-function, $\beta > 0$, and $0 \leq \sigma < 1$.

The triggering function proposed by Seyboth, et al. [100] enforces this condition with $\sigma = 0$, excludes Zeno behaviour, provides a positive MIET, and can also be applied to time-delayed single-integrator systems. Therefore, this thesis will use Seyboth's proposed triggering function for the event-triggered consensus (ETC) algorithm as defined by Theorem 3.5.

Theorem 3.5 (Event-triggered consensus [100]). *Consider the multi-agent system (3-4), with control law, $\dot{x}(t) = -L\hat{x}(t-\tau), \quad i \in \{1, ..., N\}$, undirected connected graph $\mathcal{G}$, communication delay $\tau \in [0, \pi/(2\lambda_{max}(\mathcal{G})]$ and triggering function given by*

$$f_i(t, x_i(t), \hat{x}_i(t)) = ||\hat{x}_i(t) - x_i(t)|| - (c_0 + c_1 e^{-\alpha t}) \tag{3-13}$$

with constants $c_0, c_1 \geq 0$, $c_0 + c_1 > 0$ and $0 < \alpha < \omega$, where ω is the convergence rate of system (3-8) with $e(t) = 0$. Then, for all initial conditions $x(0) = x_0 \in \mathbb{R}^N$ and $x(s) = 0$ for $s \in [-\tau, 0)$, the closed-loop system does not exhibit Zeno behaviour. Moreover, the system converges to a ball centred around the average consensus.

3-3-3 Distributed Periodic Event-triggered consensus

Periodic Event-triggered consensus algorithms check the triggering function at a predefined nonzero sampling interval, h. Due to this sampling period, a positive MIET is always ensured and Zeno behaviour is excluded.

Most proposed periodic triggering functions for single-integrator systems guarantee ISpS and enforce the condition in equation (3-12). The periodic triggering function proposed by Wang [112], in Theorem 3.6, is one of the few that also ensures convergence to the average consensus in the presence of time delays and will therefore be used throughout this thesis.

Theorem 3.6 (Periodic event-triggered consensus [112]). *In the multi-agent system (3-4), with control law, $\dot{x}(t) = -L\hat{x}(t-\tau), \quad i \in \{1, ..., N\}$, in a connected undirected network $\mathcal{G}$, assume that the event-checking period h is shared by all agents, and the communication delay $\tau \in [0, h)$. If*

$$0 < \sigma < \frac{1}{\lambda_{max}}, \quad 0 < h < \frac{2(1 - \lambda_{max}\sigma)}{\lambda_{max}}, \quad 0 < \tau < \frac{1}{\lambda_{max}} \left(1 - \lambda_{max} - \frac{\lambda_{max} h}{2}\right) \tag{3-14}$$

and the event-triggering function, with $t \in (0, h, 2h, ...]$

$$f_i(t, x_i(t), \hat{x}_i(t)) = ||x_i(t) - \hat{x}_i(t)|| - \sigma| \sum_{j \in \mathcal{N}_i} \hat{x}_i(t) - \hat{x}_j(t)|_{min} \tag{3-15}$$

then all agents asymptotically converge to average consensus.

3-4 Basic idea of distributed algorithms

Distributed algorithms are used in various applications. These algorithms often require the passing of messages between agents. These message-passing schemes are referred to as wave algorithms. Wave algorithms provide topology-dependent schemes to ensure messages are propagated to all processes:

Definition 3.4 (Wave Algorithm [107]). *A wave algorithm is a distributed algorithm that satisfies the following three requirements:*

1. ***Termination.*** *Each computation (C) is finite:*

$$\forall C : |C| < \infty$$

with |C| indicating the number of computation events.

2. ***Decision.*** *Each computation contains at least one internal event (e):*

$$\forall C : \exists e \in C : e \text{ is an internal event}$$

3. ***Dependence*** *In each computation each internal event is causally preceded by an event in each process:*

$$\forall C : \forall e \in C : (e \text{ is an internal event} \Rightarrow \forall q \in \mathbb{P}\ \exists f \in C_q : f \leq e)$$

with $\mathbb{P}$ indicating the set of processes, $\leq$ indicating that event f either happens before or at the same time as event e, and C_q denoting the subset of computations that occur just in process q.

There is a special type of wave algorithm that is classified as a traversal algorithm:

Definition 3.5 (Traversal Algorithm [107])**.** *A traversal algorithm is a special class of wave algorithm and has the following three properties.*

1. *In each computation there is one initiator, which starts the algorithm by sending out exactly one message.*
2. *A process upon receipt of a message, either sends out one message or decides.*
3. *The algorithm terminates in the initiator and when this happens each process has sent a message at least once.*

Per definition, traversal algorithms are centralised algorithms. They can however be transformed into a decentralised algorithm by initiating it in multiple nodes, essentially running a multi-start traversal algorithm.

As long as all processes have a specific identity, the traversal and wave algorithms can both be used to converge to the average consensus while requiring only a finite number of communications [107].

There exist three algorithms that can be applied to an arbitrary network. The properties of these algorithms are given in Table 3-1. Here the bit complexity refers to length of a message sent over each channel, memory refers to the number of states that need to be saved in the memory of each agent, and time refers to the time complexity of the algorithm.

Algorithm	Traversal	Bit Complexity	# Messages	Memory	Time
Phase	no	N	DN	N	$2D^{**}$
Finn	no	$2N$	$\leq 4N\|E\|$	$2N$	$O(D)$
Tarry	yes	1	$2\|E\|^{*}$	1	$2\|E\|$

Table 3-1: Distributed algorithms, based on [107]

*This only applies if the algorithm has a single initiator. For a decentralised implementation the number of messages grows linearly with the number of initiators.

**This only applies when the algorithm is initiated by a single agent. If multiple agents initiate the algorithm, the time required for convergence may decrease as all agents receive the required number of D messages earlier.

Based on the results in Table 3-1 the decision was made to implement both the phase and Tarry's algorithm. The phase algorithm requires less time to converge but more memory, and for Tarry's algorithm, this is the other way around. The pseudocode for both algorithms is given respectively in Algorithm 1 and 2.

It should be noted that the phase algorithm works optimally when all agents possess information about the network diameter. Nonetheless, it can still run if agents possess knowledge of either an overestimation of the diameter or the number of agents in the network. When an overestimation of the network diameter is used, the agents will send some redundant messages, resulting in a longer execution.

An overestimation of the diameter can be made by using the eigenvalues of the Laplacian, as per Theorem 3.7.

Theorem 3.7 (Network diameter [19])**.** . *For an undirected graph $\mathcal{G}$ with N nodes, we have:*

$$D(\mathcal{G}) \leq 1 + \left\lfloor \frac{\cosh^{-1}(N-1)}{\cosh^{-1}\left(\frac{\lambda_{max}+\lambda_2}{\lambda_{max}-\lambda_2}\right)} \right\rfloor \tag{3-16}$$

Finally, the number of agents can also be used to verify if an agent has received information from every other agent in the network. Therefore, the knowledge of the network diameter is not necessary.

Algorithm 1 Phase Algorithm [107]

global:
- D ▷ Network diameter
- $\mathcal{N}$ ▷ Set of all agents

local:
- $\mathcal{N}_p$ ▷ Set of neighbours of agent p
- $Rec_p[q] \quad : 0 \quad \forall q \in \mathcal{N}_p$ ▷ Number of messages received per neighbour
- $Sent_p \quad : 0$ ▷ Total number of messages sent
- x_0^p ▷ Initial position of agent p
- $Mem_p[i] \quad : 0 \quad \forall i \in \mathcal{N}$ ▷ Memory array, with $Mem_p[p] = x_0^p$

begin
if p is initiator **then**
  for all $q \in \mathcal{N}_p$ **do**
    send Mem_p to q
    $Rec_q[p] = Rec_q[p] + 1$
  end for
  $Sent_p += 1$
end if

while $\min_q \mathrm{Rec}_p[q] < D \quad \forall p \in \mathcal{N}$ **do**
  for all $n \in \mathcal{N}$ **do**
    if n receives Mem_q from $q \in \mathcal{N}_n$ **then**
      $Rec_n[q] = Rec_n[q] + 1$
      Update: $Mem_n = Mem_q$
    end if

    if $\min_q Rec_n[q] \geq Sent_n$ **and** $Sent_n < D$ **then**
      for all $q \in N_n$ **do**
        send Mem_n to q
      end for
      $Sent_n += 1$
    end if
  end for
end while
end

Algorithm 2 Tarry's Algorithm [107]

local:

$Rec_p[q]$: $0 \quad \forall q \in \mathcal{N}_p$ ▷ Number of messages received per neighbour
$Send_p[q]$: $False$ ▷ Array with number of messages sent
$Father_p$: $udef$ ▷ Contains the ID of father node
x_0^p ▷ Initial position of agent p
Sum_p : 0 ▷ Sum of initial positions
$\mathcal{N}_p$ ▷ Set of neighbours of agent p

For the initiator k, execute once:
begin
$Father_k = k$
$Sum_k = Sum_k + x_0^k$
Choose one: $q \in \mathcal{N}_k$
Send: Sum_k to q
$Send_k[q] = True$
end

For all processes p, upon receipt of Sum_q from $q \in \mathcal{N}_p$
begin
if $Father_p = udef$ **then**
    $Father_p = q$
    $Sum_p = Sum_q + x_0^p$
    Choose one: $m \in \mathcal{N}_p$
    Send: Sum_p to m
    $Send_p[m] = True$
else
    if $Send_p[m] \quad \forall m \in \mathcal{N}_p$ **then**
        $Sum_p = Sum_q$
        Terminate
    else
        $Sum_p = Sum_q$
        if $\exists m \in \mathcal{N}_p : (m \neq Father_p$ **and not** $Send_p[m])$ **then**
            Send Sum_p to m
            $Send_k[m] = True$
        else
            Send Sum_p to $Father_p$
            $Send_k[Father_p] = True$
            Terminate
        end if
    end if
end if
end

Chapter 4

Problem Statement

The research on consensus in the field of control and multi-agent systems (MAS) has greatly evolved in the last decade. Many variations of event-triggered algorithms have been developed to limit the network load while solving consensus under different circumstances. To the best of our knowledge most papers, at least the ones discussed so far, assume that a global or relative state measurement is available in order to evaluate the triggering function.

In most practical robotic applications, agents either have the ability to sense the relative position to their neighbours, have access to their global position or can obtain the relative distance to their neighbour. Having the ability to continuously sense the relative position, for instance via a proximity sensor, would eliminate the need to communicate the relative position and the implementation of (periodic) event-triggered to reduce the network load is no longer interesting.

In the situation where agents have continuous access to their global position, for instance via a GPS, using an (periodic) event-triggered algorithm does seem interesting. However, in addition to this approach, the field of Computer Science offers another potential alternative, wave algorithms. Wave algorithms are message-passing schemes that can be used to distribute the global position of each agent to all others in the system.

Naturally, each algorithm has its own advantages and disadvantage. Event-triggered consensus ensures asymptotic convergence to the final consensus, but the design is affected by uncertainties, as well as system dynamics and the network topology. Conversely, wave algorithms achieve convergence within a finite time and require a fixed number of communication events, but might have a high time complexity.

The question has however never been raised whether wave algorithms might be more efficient under specific circumstances. The purpose of this thesis is to start this discussion and challenge:

In which situation the knowledge of global measurements undermines the benefit of using (periodic) event-triggered algorithms over using a wave algorithm to solve the average consensus problem in single-integrator MAS.

Specifically, this thesis will limit to comparing the event-triggered consensus algorithm by Seyboth, et al. [100], the periodic event-triggered consensus algorithm by Wang [112], the phase algorithm, and Tarry's algorithm [107], with the objective to achieve average consensus in an undirected network of single-integrator robots.

This comparison will consist of the following objectives:

- *Compute mathematical lower and upper bounds for the performance metrics*
  Mathematical bounds will be computed for the convergence time, number of triggering events, and the required memory storage. These bound give an indication of the feasibility of the algorithms.
- *Implement a basic simulation subjected to time-invariant communication delays*
  The purpose of this simulation is to verify the mathematical bounds and quantitatively determine the effect of the network topology, the number of agents, and communication delay on the number of triggering events and the convergence rate of each algorithm.
- *Implement a formation control problem for the Elisa-3 robotic system*
  A application-based simulation will be used to study the effect of having a non-ideal situation. This simulation will include obstacle avoidance and formation control. Furthermore, the results of the simulation will be confirmed by implementing the algorithms on real Elisa-3 robots.

Chapter 5

Mathematical Bounds

In this chapter, we analyse the performance of three algorithms by establishing lower and upper bounds for three performance metrics. We begin by presenting the problem description, followed by deriving the bounds for each algorithm. Finally, by comparing these bounds we gain initial insights into which aspects impact performance.

5-1 Problem formulation

We consider a multi-agent systems (MAS) consisting of N single-integrator agents and a communication graph $\mathcal{G}$. We impose the following assumptions:

Assumption 5.1.

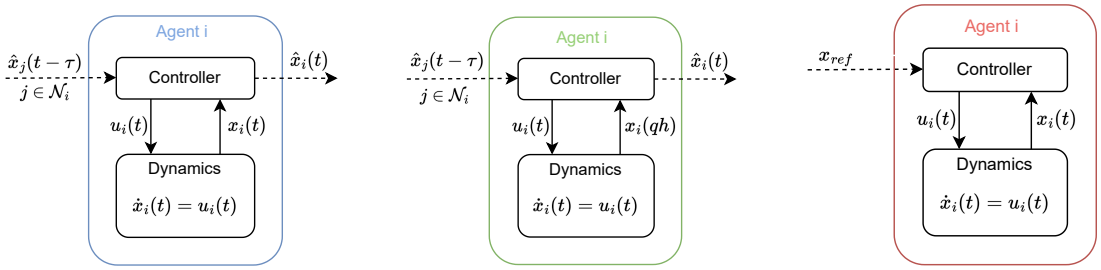
1. *Graph $\mathcal{G}$ is known, undirected, connected, and time-invariant.*
2. *The communication over each edge of Graph $\mathcal{G}$ may be subjected to a bounded constant time delay, $0 \leq \tau < \frac{\pi}{2\lambda_{max}(\mathcal{G})}$.*
3. *The agents operate in a two-dimensional plane.*
4. *The control input of the algorithms is limited to u^{max}.*
5. *Convergence is achieved once all agents are ϵ close to the final consensus point.*
6. *The processing time of a task/event requires zero time units.*

Assumption 5.1.4 is introduced to ensure a fair comparison by restricting the maximum control input of the wave algorithms to the maximum control input used in the event-triggered consensus (ETC) and periodic event-triggered consensus (PETC) algorithms. In most cases, the maximum control input of the ETC and PETC algorithms will be the same and equal to the control input at initiation; $\max_{i\in\mathcal{N}} u_i(0)$.

Problem. *Determine upper and lower bounds on the convergence time t, number of triggering events $|C|$, and the required memory needed by the ETC, PETC, and wave algorithms.*

5-2 System Overview

The focus of this thesis is on single-integrator agents. Each agent consists of a controller and dynamics as shown in Figure 5-1.



(a) ETC **(b)** PETC **(c)** Wave

Figure 5-1: Model of a single agent using in Figure 5-1a an Event-Triggered algorithm, 5-1b a Periodic Event-Triggered algorithm, and 5-1c a Wave algorithm

where $x_i(t) \in \mathbb{R}^2$ is the state, $\hat{x}_i(t) \in \mathbb{R}^2$ is the last broadcasted state, $u_i(t) \in \mathbb{R}^2$ is the control input, $h \in \mathbb{R}_{>0}$ is the sampling period, and $q \in \mathbb{N}$ takes values in increasing order.

The next sections, provide the closed-loop system model used by each algorithm and discusses system variables that are of importance to the derivation of the bounds.

5-2-1 Event-triggered Algorithm

System Model

As discussed in Chapter 3, the controller of each agent monitors its own state continuously and broadcasts its current state over the network when the triggering function in Theorem 3.5 is fulfilled.

The controller used in Theorem 3.5 generates the closed-loop dynamics as described in equation (3-8) and duplicated here for convenience:

$$\begin{aligned} \dot{x}(t) &= -L\hat{x}(t-\tau) = -Lx(t-\tau) - Le(t-\tau) \\ x(0) &= x_0 \end{aligned} \tag{5-1}$$

By applying a change of variables, $\delta_i(t) \in \mathbb{R}^2$, satisfying $\delta_i(t) = x_i(t) - \overline{x}_0$ with $\overline{x}_0$ the average of the initial positions x_0, the closed-loop dynamics are rewritten as:

$$\begin{aligned} \dot{\delta}(t) &= -L\delta(t-\tau) - Le(t-\tau) \\ \delta(t) &= -\overline{x}_0 \quad \forall t \in [-\tau, 0) \\ \delta(0) &= x_0 - \overline{x}_0 \end{aligned} \tag{5-2}$$

Here $\delta(t) = [\delta_1(t), \delta_2(t), ..., \delta_N(t)]^T \in \mathbb{R}^{N\times 2}$ is the transformed state vector, and $e(t) = [e_1(t), e_2(t), ..., e_N(t)]^T \in \mathbb{R}^{N\times 2}$ is the error vector, satisfying $e_i(t) = \hat{x}_i(t) - x_i(t)\ \forall i \in \mathcal{N}$.

Convergence rate

The design of the triggering function for the ETC algorithm depends on the convergence rate of system (5-2) with $e(t-\tau)=0$. Duan et al. [35] have presented Theorem 5.1, which indicates that the convergence rate is influenced by the time delay.

Theorem 5.1 (Rate of convergence of a unforced linear time-delay system [35])**.** *Consider the linear time-delayed system $\dot{x}(t)=Ax(t-\tau)$ with $\phi(t), t\in[-\tau,0]$. Furthermore, $\lambda(A)=\{\alpha_1,...,\alpha_n\}\subset\mathbb{R}_{<0}$, A is Hurwitz with real eigenvalues, and the delay is fixed and bounded by $\overline{\tau}=\frac{\pi}{2|\alpha_n|}$. Then,*

The rate of convergence, is given by

$$\omega=-\max\left[\frac{1}{\tau}\mathfrak{Re}\left(W_0(\alpha_i\tau)\right)\right]_{i=1}^{n} \tag{5-3}$$

with $W_0(z)$ the principal branch of the Lambert function.

Moradian and Kia [79] extend this Theorem and show that the maximum achievable convergence rate is achieved when the system is subjected to a time delay, $\tau=\tau^*$, as presented in Theorem 5.2.

Theorem 5.2 (Maximum rate of convergence of a unforced linear time-delay system [79])**.** *Consider the linear time-delayed system $\dot{x}(t)=Ax(t-\tau)$ with $\phi(t), t\in[-\tau,0]$. Furthermore, $\lambda(A)=\{\alpha_1,...,\alpha_n\}\subset\mathbb{R}_{<0}$, A is Hurwitz with real eigenvalues, and the delay is fixed and bounded by $\overline{\tau}=\frac{\pi}{2|\alpha_n|}$. Then,*

The maximum rate of convergence

$$\omega^*=e^{\frac{\arccos\frac{\alpha_1}{\alpha_n}}{\sqrt{\left(\frac{\alpha_n}{\alpha_1}\right)^2-1}}}|\alpha_1| \tag{5-4}$$

is attained at

$$\tau^*=\frac{\arccos\frac{\alpha_1}{\alpha_n}}{\sqrt{\left(\frac{\alpha_n}{\alpha_1}\right)^2-1}|\alpha_1|}e^{-\frac{\arccos\frac{\alpha_1}{\alpha_n}}{\sqrt{\left(\frac{\alpha_n}{\alpha_1}\right)^2-1}}} \tag{5-5}$$

It is important to note that, Theorem 5.1 and 5.2 only hold for time-delayed systems that satisfy the Hurwitz criteria. System (5-2) with $e(t-\tau)=0$ does not satisfy this requirement as the Laplacian of an undirected connected network is not Hurwitz. However, by applying a change of variables, $y=T^T\delta$ with $T=[\frac{1}{\sqrt{N}}\mathbf{1}_N,R]$ with R such that $T^TT=TT^T=I_N$, the unforced dynamics can be represented by:

$$\begin{aligned}\dot{y}_1(t)&=0\\ \dot{y}_{2:N}(t)&=-(R^TLR)y_{2:N}(t-\tau)\end{aligned}$$

The matrix $-(R^TLR)$ is Hurwitz with $\lambda(-(R^TLR))=\{-\lambda_2,...,-\lambda_N\}$ and Theorem 5.1 and 5.2 can be applied.

Overshoot

We refer to the overshoot, M, as the maximum deviation of the system from the maximum initial disagreement, $||\delta(0)||_{max} = \max_{i \in \mathcal{N}} \|\delta_i(0)\|$.

Proposition 5.1. *Under the event-triggering condition in Theorem 3.5, the time-delayed system (5-2) may exhibit an overshoot ranging from*

$$1 \leq M \leq 1 + \frac{c_0 + c_1 + \tau |u^{max}|}{||\delta(0)||_{max}} \tag{5-6}$$

Here $||\delta(0)||_{max}$ refers to the maximum initial disagreement.

Proof. In the best-case scenario, the maximum distance between the agents and the consensus point will never exceed $||\delta(0)||_{max}$. Therefore, the overshoot is lower bounded by 1.

The worst-case overshoot occurs when an agent is a distance of $||\delta(0)||_{max}$ away from the average consensus point and starts moving in the opposite direction. Due to the triggering function $||e_i(t)|| < c_0 + c_1$, an agent can move at most a distance of $c_0 + c_1$ before triggering. Once a triggering event occurs, the time delay and maximum control input ensure an agent can move at most a distance of $\tau |u^{max}|$ before correcting its trajectory. Therefore, an agent can move at most a distance of $c_0 + c_1 + \tau |u^{max}|$ away from $||\delta(0)||_{max}$. This leads to the upper bound in equation (5-6). □

5-2-2 Periodic Event-triggered Algorithm

System Model

The periodic event-triggered algorithm in Theorem 3.6 employs the same controller as the event-triggered algorithm. Consequently, the same change of variables can be applied to the system, and equation (5-2) can be used to describe the closed-loop system.

Convergence rate

Since the closed-loop system of the periodic event-triggered algorithm is equivalent to the closed-loop system of the event-triggered algorithm, the convergence rate of the time-delayed system (5-2), with $e(t - \tau) = 0$, described in Theorem 5.1 can be extended to the periodic event-triggered algorithm.

Overshoot

Proposition 5.2. *Under the periodic event-triggering condition in Theorem 3.6, the time-delayed system (5-2) may exhibit an overshoot ranging from*

$$1 \leq M \leq 1 + \frac{(h + \sigma + \tau) |u^{max}|}{||\delta(0)||_{max}} \tag{5-7}$$

Here $||\delta(0)||_{max}$ refers to the maximum initial disagreement.

Proof. The overshoot can be computed in a similar as in the proof of proposition 5.1. Here the worst-case overshoot is influenced by the triggering function $||e_i|| \leq \sigma|u^{max}|$, the time delay τ, the sampling period h, and the maximum control input $|u^{max}|$. These elements together ensures that an agent can move at most a distance of $(h+\sigma+\tau)|u^{max}|$ away from $||\delta(0)||_{max}$. Therefore we have equation (5-7). □

5-2-3 Wave Algorithm

System model

The wave algorithm is used to compute the final consensus position $x_{ref} \in \mathbb{R}^2$. Once this value is known to all agents, each agent applies a simple proportional controller to converge to x_{ref}.

Consequently, the closed-loop dynamics for every agent i, can be expressed as follows:

$$\begin{aligned} \dot{x_i}(t) &= u_i(t) \\ u_i(t) &= -\max\left(-u^{max}, \min\left(\begin{bmatrix} k1 & 0 \\ 0 & k2 \end{bmatrix}(x_i(t) - x_{ref}), u^{max}\right)\right) \\ &= -\max\left(-u^{max}, \min\left(K(x_i(t) - x_{ref}), u^{max}\right)\right) \end{aligned} \tag{5-8}$$

Here, $k1 \in \mathbb{R}_{\geq 0}$ and $k2 \in \mathbb{R}_{\geq 0}$ are the gain factors of the controller. Finally, by applying the same change of variables, $\delta(t) = x(t) - \overline{x}_0$, where $\overline{x}_0 = x_{ref}$, we obtain:

$$\begin{aligned} \dot{\delta}_i(t) &= u_i(t) \\ u_i(t) &= -\max\left(-u^{max}, \min\left(K\delta_i(t), u^{max}\right)\right) \end{aligned} \tag{5-9}$$

Execution time

The time complexity in Table 3-1 provides the execution time of each algorithm, under the assumption that the transmission time is at most one-time unit and the processing time requires zero time units.

Given Assumption 5.1.2, the communication delay is bounded and constant. Therefore the execution time of the algorithms can be defined by:

Algorithm	**# Messages**	**Memory**	**Execution time**
Phase	DN	N	$2D\tau^*$
Tarry	$2\|E\|^{**}$	1	$2\|E\|\tau^{**}$

Table 5-1: Wave algorithm properties

*This only applies when the algorithm is initiated by a single agent. When multiple agents initiate the algorithm, the execution time could potentially reduce due to the earlier exchange of information among the agents.

**Note that the execution time and the number of messages of Tarry's algorithm are determined by assuming the algorithm is initiated by a single initiator. Consequently, after

completion of the algorithm, only the initiator has obtained the consensus position x_{ref}. To ensure that all agents have knowledge of x_{ref}, two options are available: either all agents must act as initiators, resulting in more messages, or another round of communication is necessary to circulate x_{ref} to all other agents.

5-3 Event-triggered algorithm

5-3-1 Convergence time

Lower bound

Proposition 5.3. *Given Assumption 5.1, the minimum time required for the time-delay multi-agent system (5-2) to converge, subject to the event-triggering condition in Theorem 3.5, is*

$$t_{ETC} > \max_{i \in \mathcal{N}} \frac{||\delta_i(0)|| - \epsilon}{||\sum_{j \in N_i} \delta_j(0) - \delta_i(0)||} \tag{5-10}$$

Proof. The event-triggered algorithm converges the fastest when all agents move in a straight line to the final consensus point without any triggering events.

Moving in a straight line, each agent travels at most $||\delta_i(0)|| - \epsilon$. If the agents do not trigger, the control input is defined by $u_i(0)$ over the entire trajectory.

This results in a lower bound of:

$$t_{ETC} > \max_{i \in \mathcal{N}} \frac{||\delta_i(0)|| - \epsilon}{||u_i(0)||} = \max_{i \in \mathcal{N}} \frac{||\delta_i(0)|| - \epsilon}{||\sum_{j \in N_i} \delta_j(0) - \delta_i(0)||} \tag{5-11}$$

□

Proposition 5.5 indicates that increasing the connectivity to neighbours will result in faster convergence, as it will increase an agent's average control input, $u_i(t)$.

Furthermore, the proposed bound relies on the assumption that agents will not trigger and will maintain their initial control input, $u_i(0)$, throughout the trajectory. However, this assumption is unlikely since the triggering condition enforces a trigger event when an agent has moved a distance of at most $c_0 + c_1$. After triggering, the agents will have moved closer, and the control input $u_i(t)$ will decrease. Consequently, this bound may be overly conservative.

Upper bound

Proposition 5.4. *Given Assumption 5.1, the time required for the time-delayed multi-agent system (5-2) to converge using the triggering condition in Theorem 3.5 can be bounded by*

$$t_{ETC} \leq \frac{1}{\alpha} \ln \left[e^{\omega \tau} (1 + \tau)^{1/2} |\delta(0)|_{max} + \frac{||L|| c_1 \sqrt{N}}{\omega - \alpha} \right] - \frac{1}{\alpha} \ln \left[\frac{\epsilon}{M} - \frac{||L|| c_0 \sqrt{N}}{\omega} \right] \tag{5-12}$$

Here $\omega \in \mathbb{R}_{>0}$ denotes the convergence rate in Theorem 5.1, $M \in \mathbb{R}_{>0}$ is associated with the overshoot in proposition 5.1, and $|\delta(0)|_{max}$ is the maximum absolute initial displacement $|\delta(0)|_{max} = \max_{r,t\in\mathcal{N}} |\delta_{rt}(0)|$.

Proof. Pepe and Liang show in the proof of proposition 2.5 that the solution of the one-dimensional system:

$$\dot{\tilde{\delta}}_i(t) = -L\tilde{\delta}(t-\tau) - L\tilde{e}(t-\tau) \tag{5-13}$$

with $\tilde{\delta}(t) \in \mathbb{R}^N$ and $\tilde{e}(t) \in \mathbb{R}^N$, can be bounded by:

$$||\tilde{\delta}(t)|| \leq ||S(t-\tau)||(1+\tau)^{1/2}||\tilde{\delta}(0)||_\infty + \int_\tau^t ||S(t-\tau)||||L\tilde{e}(\tau)||d\tau \tag{5-14}$$

Here $S(t) : Z \to Z$ is the underlying C_0-semigroup associated with the unforced time-delayed system, equation (5-13) with $\tilde{e}(t) = 0$ [8].

Given that, Olfati-Saber [89], proved the unforced time-delayed system (5-13) is asymptotically stable for $\tau \in [0, \pi/(2\lambda_{max}(\mathcal{G})]$, the following inequality on the semigroup holds (see Theorem 2.3 [46]):

$$||S(t)|| \leq Me^{-\omega t}, \quad t \geq 0 \tag{5-15}$$

Here $\omega \in \mathbb{R}_{>0}$ denotes the convergence rate in Theorem 5.1, and $M \in \mathbb{R}_{>0}$ is associated with the overshoot in proposition 5.1.

The triggering function in Theorem 3.5 limits the error, $||e_i(t)|| \leq c_0 + c_1 e^{-\alpha t} \quad \forall i \in \mathcal{N}$ and thus $||\tilde{e}_i(t)|| \leq c_0 + c_1 e^{-\alpha t} \quad \forall i \in \mathcal{N}$. Combining this with inequality (5-14) results in the following bound for $||\tilde{\delta}(t)||$:

$$\begin{aligned} ||\tilde{\delta}(t)|| &\leq Me^{-\omega(t-\tau)}(1+\tau)^{1/2}||\tilde{\delta}(0)||_\infty + \int_\tau^t Me^{-\omega(t-\tau)}||L||\sqrt{N}(c_0 + c_1 e^{-\alpha\tau})d\tau \\ &= Me^{-\omega(t-\tau)}(1+\tau)^{1/2}||\tilde{\delta}(0)||_\infty + \left[\frac{M||L||c_0\sqrt{N}}{\omega} + \frac{M||L||c_1\sqrt{N}}{\omega-\alpha}e^{-\alpha t}\right] \\ &- \left[\frac{M||L||c_0\sqrt{N}}{\omega} + \frac{M||L||c_1\sqrt{N}}{\omega-\alpha}e^{-\alpha\tau}\right]e^{-\omega(t-\tau)} \end{aligned} \tag{5-16}$$

We further simplify this inequality by using that $\alpha < \omega$ to ensure asymptotic stability. This implies that $e^{-\omega t} \leq e^{-\alpha t}$. Therefore, the bound can be rewritten as:

$$\begin{aligned} ||\tilde{\delta}(t)|| &\leq Me^{-\omega(t-\tau)}(1+\tau)^{1/2}||\tilde{\delta}(0)||_\infty + \left[\frac{M||L||c_0\sqrt{N}}{\omega} + \frac{M||L||c_1\sqrt{N}}{\omega-\alpha}e^{-\alpha t}\right] \\ &\leq Me^{-\alpha t}\left[e^{\omega\tau}(1+\tau)^{1/2}||\tilde{\delta}(0)||_\infty + \frac{||L||c_1\sqrt{N}}{\omega-\alpha}\right] + \frac{M||L||c_0\sqrt{N}}{\omega} \end{aligned} \tag{5-17}$$

Finally, system (5-2) considers agents with a two-dimensional state, $\delta(t) \in \mathbb{R}^{Nx2}$. Therefore we have:

$$||\delta(t)||_{max} \leq Me^{-\alpha t}\left[e^{\omega\tau}(1+\tau)^{1/2}|\delta(0)|_{max} + \frac{||L||c_1\sqrt{N}}{\omega-\alpha}\right] + \frac{M||L||c_0\sqrt{N}}{\omega} \tag{5-18}$$

with $|\delta(0)|_{max} = \max_{r,t\in\mathcal{N}}|\delta_{rt}(0)|$. To ensure all agents convergence to a ball with radius ϵ, it must hold that $||\delta(t)||_{max} \leq \epsilon \ \forall i \in \mathcal{N}$. Solving this inequality for t results in the upper bound given in equation (5-12).

□

Proposition 5.4 reveals, just like the lower bound in proposition 5.3, that network connectivity plays a critical role in the convergence time; increasing connectivity leads to greater ω, allowing for a larger α and thus faster convergence.

Furthermore, one may observe that increasing the time delay would increase the convergence time. However, Theorems 5.1 and 5.2 prove that the convergence rate is optimal at a specific time delay, $\tau = \tau^* > 0$. Therefore, increasing the time delay may improve the performance of the ETC algorithm.

In addition, it can be deduced from inequality (5-16) that selecting the triggering parameter c_0 is critical for ensuring convergence. This is due to the fact that as $t \to \infty$, the system converges to a ball with radius:

$$r_{ETC} = \lim_{t\to\infty}||\delta(t)||_{max} \leq \frac{M||L||c_0\sqrt{N}}{\omega} \tag{5-19}$$

Finally, it is worth noting that the simplifications made in equation (5-17) might lead to a conservative bound. A less conservative estimate may be obtained by using proposition 5.4 as an initial guess for an optimisation algorithm that solves inequality (5-16) for t.

5-3-2 Triggering events

Lower bound

Proposition 5.5. *The minimum number of triggering events required by the time-delayed system (5-2) with N agents, subjected to the triggering condition in Theorem 3.5 is*

$$|C|_{ETC} \geq \sum_{i=0}^{N}\left(1 + \frac{||\delta_i(0)|| - \epsilon}{(c_0+c_1)}\right) \tag{5-20}$$

Proof. In the best-case scenario, all agents move in a straight line to the final consensus position. This ensures they travel at most a distance of $||\delta_i(0)|| - \epsilon$.

The triggering function ensures; $||e_i(t)|| \leq c_0 + c_1$, which indicates an agent can move at most a distance of $c_0 + c_1$ before a triggering event takes place.

Since an agent will always trigger at initiation, we know that a single agent will trigger at least:

$$|C_i|_{ETC} \geq 1 + \frac{||\delta_i(0)|| - \epsilon}{(c_0 + c_1)} \tag{5-21}$$

Given the number of triggering events of a single agent, the lower bound of the entire swarm of agents is given by equation (5-20).

□

This lower bound supports the intuition that the number of triggering events will increase as the number of agents and travel distance grow.

However, proposition 5.5 may be optimistic because the triggering function is time dependent, which means that the distance an agent can travel without triggering will decrease over time.

Upper bound

Proposition 5.6. *The number of triggering events required by the time-delayed system (5-2) with N agents, under the triggering condition stated in Theorem 3.5, is limited to a maximum of:*

$$|C|_{ETC} \leq \sum_{i=1}^{N} \frac{t_{ETC}||L||(M(e^{\omega\tau}\sqrt{1+\tau}|\delta(0)|_{max} + \frac{||L||c_0\sqrt{N}}{\omega} + \frac{||L||c_1\sqrt{N}}{\omega-\alpha}) + \sqrt{N}c_0)}{c_0} \tag{5-22}$$

Here t_{ETC} represents the convergence time of the ETC algorithm, as defined in proposition 5.4, M refers to the overshoot which can be bounded by proposition 5.1, ω is associated with the convergence rate in Theorem 5.1, and $|\delta(0)|_{max}$ is the maximum absolute initial displacement $|\delta(0)|_{max} = \max_{r,t\in\mathcal{N}} |\delta_{rt}(0)|$.

Proof. The maximum number of triggering events occurs when the triggering function is fulfilled each time it is assessed. The minimum inter-event time (MIET) provides the minimum time between triggering events and proposition 5.4 provides an upper bound on the convergence time.

Given this convergence time and the MIET, we can formulate the upper bound per agent:

$$|C_i|_{etc} \leq \frac{t_{ETC}}{t_{MIET}} \tag{5-23}$$

Seyboth et al., [100], provides a lower bound on the MIET under the given triggering condition:

$$t_{MIET} \geq \frac{c_0}{||L||(||\delta(t-\tau)|| + \sqrt{N}c_0)} \tag{5-24}$$

Given the bound on $||\delta(t)||$ in equation (5-16) and the number of agents, the number of triggering events is upper bounded by equation (5-22).

□

Naturally, this proposition verifies that as the connectivity increases, fewer triggering events are required as agents have access to more information. However, the need for extra triggering events with the additional neighbours may outweigh this benefit.

Moreover, when the system is subjected to a longer time delay the system may need more triggering events to converge. This can be explained by the fact that agents have a delayed response to the actions of their neighbours.

5-3-3 Scalability

Scalability considers the required memory storage in order to execute the algorithm. Each agent needs to keep track of the last broadcasted state of its neighbours and its own state. Therefore the total memory required is:

$$\max_{i \in \mathcal{N}} N_i + 1 \tag{5-25}$$

5-4 Periodic Event-triggered algorithm

5-4-1 Convergence time

Lower bound

Proposition 5.7. *Given Assumption 5.1, the minimum time required by the time-delay multi-agent system (5-2) under the periodic event-triggering condition in Theorem 3.6 to converge is*

$$t_{PETC} \geq \max_{i \in \mathcal{N}} \frac{||\delta_i(0)|| - \epsilon}{||\sum_{j \in N_i} \delta_j(0) - \delta_i(0)||} \tag{5-26}$$

Proof. See the proof of proposition 5.3. □

Since the derivation of this bound follows a similar approach as proposition 5.3, we can make the same observations as outlined in section 5-3.

Upper bound

Proposition 5.8. *Given Assumption 5.1, the maximum time required by the time-delay multi-agent system (5-2) under the periodic event-triggering condition in Theorem 3.6 to converge is*

$$t_{PETC} \leq \frac{1}{\omega} \ln \left[\frac{\frac{\epsilon}{M} - \frac{||L||\sigma|u^{max}|}{\omega}}{e^{\omega\tau}\sqrt{1+\tau}|\delta(0)|_{max} - \frac{||L||\sigma|u^{max}|}{\omega}} \right]^{-1} \tag{5-27}$$

Here M represents the overshoot which can be bounded by proposition 5.1, ω is associated with the convergence rate in Theorem 5.1, and $|\delta(0)|_{max}$ is the maximum absolute initial displacement $|\delta(0)|_{max} = \max_{r,t \in \mathcal{N}} |\delta_{rt}(0)|$.

Proof. The bound on the system's solution proposed in equation (5-14) also applies to the periodic event-triggered algorithm.

Given the triggering function in Theorem 3.6, we know that:

$$||e_i(t)|| \leq \sigma| \sum_{j \in \mathcal{N}_i} \hat{x}_i(t) - \hat{x}_j(t)| \leq \sigma|u^{max}| \quad \forall i \in \mathcal{N} \tag{5-28}$$

Combining this with equation (5-14) results in a bound on $||\delta(t)||_{max}$:

$$\begin{aligned} ||\delta(t)||_{max} &\leq Me^{-\omega(t-\tau)}(1+\tau)^{1/2}|\delta(0)|_{max} + \int_{\tau}^{t} Me^{-\omega(t-\tau)}||L||\sigma|u^{max}|d\tau \\ &= Me^{-\omega(t-\tau)}(1+\tau)^{1/2}|\delta(0)|_{max} + \frac{M||L||\sigma|u^{max}|}{\omega}(1-e^{-\omega t}) \end{aligned} \tag{5-29}$$

Solving the inequality, $||\delta(t)||_{max} \leq \epsilon$, for t provides the upper bound on the convergence time in equation (5-27). □

Similar to the ETC algorithm, having a high convergence rate, or indirectly, a high network connectivity, is of importance to reduce the convergence time.

Furthermore, it verifies that increasing the allowed error, ϵ, or increasing the initial displacement, consequently increasing $|\delta(0)|_{max}$, between agents will not improve the convergence time.

Additionally, we observe from inequality (5-29) that as $t \to \infty$ the system convergences to a ball with radius:

$$r_{PETC} = \lim_{t \to \infty} ||\delta(t)||_{max} \leq \frac{M||L||\sigma|u^{max}|}{\omega} \tag{5-30}$$

Therefore, the selection of the triggering parameter is important to ensure that convergence, as defined by Assumption 5.1.5, is guaranteed.

5-4-2 Triggering events

Lower bound

Proposition 5.9. *The time-delayed system (5-2) with N agents under the triggering condition given in Theorem 3.6 requires a minimum number of triggering events to convergence, that bound is:*

$$|C|_{PETC} \geq \sum_{i=0}^{N} \left(1 + \frac{||\delta_i(0)|| - \epsilon}{\sigma|u^{max}|}\right) \tag{5-31}$$

Proof. As imposed in the proof of proposition 5.5, moving in a straight line to the consensus point minimises the trajectory of each agent, to $||\delta_i(0)|| - \epsilon$, and therefore minimises the number of triggering events among the agents.

Each agent triggers once at initiation after which the triggering function ensures the maximum distance an agent can move before triggering is:

$$||e_i(t)|| \leq \sigma| \sum_{j \in \mathcal{N}_i} \hat{x}_i(t) - \hat{x}_j(t)| \leq \sigma|u^{max}| \tag{5-32}$$

Given the maximum distance between events, the initial disagreement, and the number of agents, the minimum number of triggering events is defined by equation (5-31).

□

The derivation of this bound follows a similar approach as that in proposition 5.5. As a result, the same observations as discussed in section 5-3 can be made.

Upper bound

Proposition 5.10. *The maximum number of triggering events required by a time-delayed system with N agents, as per the triggering condition mentioned in Theorem 3.6, is restricted to*

$$|C|_{PETC} \leq \sum_{i=1}^{N} \frac{t_{PETC}}{h} \tag{5-33}$$

Here t_{PETC} denotes the time required to converge, which is upper bounded by proposition 5.8.

Proof. The maximum number of triggering events occurs when the triggering function is satisfied every time it is evaluated. Since the triggering function is assessed with a sampling period equal to h and the maximum convergence time of the system is given by t_{PETC} in proposition 5.8, the maximum number of triggering events is given by equation (5-33)

□

Proposition 5.10 supports the intuition that decreasing h by increasing the frequency of checking the triggering function may result in an increase in the number of triggering events.

5-4-3 Scalability

The PETC algorithm requires the same amount of memory as the ETC algorithm. The consensus protocol and triggering function require each agent to keep track of the last broadcasted state of its neighbours and its own current and last broadcasted state:

$$\max_{i \in \mathcal{N}} N_i + 1 \tag{5-34}$$

5-5 Wave algorithms

The phase and Tarry algorithms execute in finite time with a finite number of triggering events. The number of triggering events and memory required for each algorithm has already been provided in Table 5-1.

5-5-1 Convergence time

Lower bound

Proposition 5.11. *The minimum time required for a wave algorithm to converge, given the closed-loop system (5-9) and Assumption 5.1, is given by:*

$$t_{wave} \geq t_{algorithm} + \frac{||\delta(0)||_{max} - \epsilon}{||u_{max}||} \tag{5-35}$$

Here $t_{algorithm}$ is the execution time of the wave algorithm, given in Table 5-1, and $||\delta(0)||_{max}$ refers to the maximum initial disagreement.

Proof. Once the algorithm has terminated, the time to converge is mainly limited by the agent the farthest from the average consensus point, the controller gain, and the maximum control input. By selecting controller gains, $k1$ and $k2$, of sufficient magnitude, the convergence time is restricted only by u_{max}, leading to the minimum limit given in equation (5-35).

□

Upper bound

Proposition 5.12. *The maximum time required for a wave algorithm to converge, given the closed-loop system (5-9) and Assumption 5.1 is given by:*

$$t_{wave} \leq t_{algorithm} + t^* + K^{-1} \ln \left[\frac{\max_{i \in \mathcal{N}}(||\delta_i(t^*)||)}{\epsilon} \right] \tag{5-36}$$

Here $t_{algorithm}$ refers to the execution time of the wave algorithm, given in Table 5-1. Furthermore t^* equals:

$$t^* = \max_{i \in \mathcal{N}} \left(\frac{|\delta_i(0)| - K^{-1}|u^{max}|}{|u^{max}|},\ 0 \right) \tag{5-37}$$

and:

$$\delta_i(t^*) = \begin{cases} \delta_i(0) - u^{max} t^* & \text{if } \delta_i(0) \geq 0 \\ \delta_i(0) + u^{max} t^* & \text{if } \delta_i(0) < 0 \end{cases}$$

Proof. Once the algorithms have terminated, the time to converge is limited by the controller of the agent the farthest away from the consensus point x_{ref}.

The system without saturated input takes the form of an ordinary differential equation, therefore the decay function of each agent would be expressed as:

$$\delta_i(t) \leq e^{-Kt} \delta_i(0) \tag{5-38}$$

However, the system's input is bounded by u^{max}. Consequently, when using small gain factors, there exists a time, denoted by t_i^*, during which the control input of agent i is restricted to u^{max}.

$$|u_i(t)| = |u^{max}| \quad \forall t \leq t_i^* \tag{5-39}$$

At time, $t = t_i^*$, the agent's control input is no longer constrained and it holds that $|u^{max}| = |K\delta_i(t^*)|$. Furthermore, the agent has travelled a distance of $|u^{max}|t^*$, or equivalently:

$$\begin{aligned} |\delta_i(0)| - |\delta_i(t^*)| &= |\delta_i(0)| - K^{-1}|u^{max}| \\ &= |u^{max}|t^* \end{aligned} \tag{5-40}$$

Therefore, the control input of agent i is saturated for the duration:

$$t_i^* = \max\left(\frac{|\delta_i(0)| - K^{-1}|u^{max}|}{|u^{max}|}, 0\right) \tag{5-41}$$

Given, $t^* = \max_{i\in\mathcal{N}} t_i^*$, the decay function in equation (5-38) with $\delta_i(0) = \delta(t^*)$, and the allowed error $||\delta_i(t)|| \leq \epsilon$, the convergence time of the system is:

$$\begin{aligned} t_{wave} &= t_{algorithm} + t^* + t_{controller} \\ &\leq t_{algorithm} + t^* + ||K||^{-1} \ln\left[\frac{\max_{i\in\mathcal{N}}(||\delta_i(t^*)||)}{\epsilon}\right] \end{aligned} \tag{5-42}$$

□

Propositions 5.11 and 5.12 demonstrate that by selecting sufficiently large controller gains, $k1$ and $k2$, the convergence time of the wave algorithms is only limited by the maximum control input and the execution time of the algorithms, as t^* approaches zero.

Upon comparing the execution time of the wave algorithms, we can observe that in most cases, the phase algorithm converges faster than Tarry's algorithm. Only if the diameter of the network is equal to the number of edges in the network, which is only the case for a line network, might it be beneficial to use Tarry's algorithm as it requires less memory.

5-6 Comparison

A comparison of the two wave algorithms shows that it is evident the phase algorithm outperforms Tarry's algorithm in terms of convergence time and number of triggering events. However, the phase algorithm requires a considerable amount of temporary storage since it needs to store and transmit the global positions of all the agents in the network. Therefore, if the network is large, this may become infeasible, and it would be better to use either the ETC or PETC algorithm.

From now on, we will assume that the memory capacity is large enough to support the phase algorithm. Therefore, the comparison will only focus on the execution time and the number of triggering events.

The bounds on the convergence time suggest that the performance of the ETC and PETC algorithms is significantly influenced by network connectivity. Higher connectivity will lead to a faster convergence, but it reduces the system's robustness to time delays and generally results in a smaller network diameter.

On the other hand, the convergence time of the wave algorithms improves with a smaller time delay. As the time delay approaches zero, the execution time of the algorithms reduces to zero and the algorithms converge at the same rate or even faster than the ETC and PETC algorithms.

Therefore, there is a trade-off between having high connectivity while still allowing a large enough time delay and network diameter to outperform the wave algorithms.

Finally, a comparison based on the number of triggering events shows that the phase algorithm requires a fixed number of triggering events. However, as the number of agents or the network diameter grows, the phase algorithm requires more triggering events. This is also the case for the ETC and PETC algorithms. As a result, the ETC and PETC algorithms may only require fewer triggering events if the agents are already close to the average consensus point. This is not common in practical applications, and therefore the phase algorithm will most likely outperform the others in terms of the number of transmissions.

To summarise the results of this chapter, Table 5-2 and 5-3, include the mathematical bounds of all four algorithms.

Metric	**ETC**	**PETC**
t_{lower}	$\max_{i\in\mathcal{N}} \frac{\|\|\delta_i(0)\|\|-\epsilon}{\|\|\sum_{j\in N_i}\delta_j(0)-\delta_i(0)\|\|}$	$\max_{i\in\mathcal{N}} \frac{\|\|\delta_i(0)\|\|-\epsilon}{\|\|\sum_{j\in N_i}\delta_j(0)-\delta_i(0)\|\|}$
t_{upper}	$\frac{1}{\alpha}\ln\left[e^{\omega\tau}(1+\tau)^{1/2}\|\delta(0)\|_{max}+\frac{\|L\|c_1\sqrt{N}}{\omega-\alpha}\right]-\frac{1}{\alpha}\ln\left[\frac{\epsilon}{M}-\frac{\|L\|c_0\sqrt{N}}{\omega}\right]$	$\frac{1}{\omega}\ln\left[\frac{\frac{\epsilon}{M}-\frac{\|L\|\sigma\|u^{max}\|}{\omega}}{e^{\omega\tau}\sqrt{1+\tau}\|\delta(0)\|_{max}-\frac{\|L\|\sigma\|u^{max}\|}{\omega}}\right]^{-1}$
$\|C\|_{lower}$	$\sum_{i=0}^{N}\left(1+\frac{\|\|\delta_i(0)\|\|-\epsilon}{(c_0+c_1)}\right)$	$\sum_{i=0}^{N}\left(1+\frac{\|\|\delta_i(0)\|\|-\epsilon}{\sigma\|u^{max}\|}\right)$
$\|C\|_{upper}$	$\sum_{i=0}^{N}\frac{t_{ETC}}{t_{MIET}}$	$\sum_{i=0}^{N}\frac{t_{PETC}}{h}$
Memory	$\max_{i\in\mathcal{N}} N_i+1$	$\max_{i\in\mathcal{N}} N_i+1$

Table 5-2: Mathematical bounds of the ETC and PETC algorithms

Metric	**Phase**	**Tarry**
t_{lower}	$2D\tau+\frac{\|\|\delta(0)\|\|_{max}-\epsilon}{\|u_{max}\|}$	$2\|E\|\tau+\frac{\|\|\delta(0)\|\|_{max}-\epsilon}{\|u_{max}\|}$
t_{upper}	$2D\tau+t^*+K^{-1}\ln\left[\frac{\max_{i\in\mathcal{N}}(\|\|\delta_i(t^*)\|\|)}{\epsilon}\right]$	$2\|E\|\tau+t^*+K^{-1}\ln\left[\frac{\max_{i\in\mathcal{N}}(\|\|\delta_i(t^*)\|\|)}{\epsilon}\right]$
$\|C\|_{lower}$	DN	$2\|E\|$
$\|C\|_{upper}$	DN	$2\|E\|$
Memory	N	1

Table 5-3: Mathematical bounds of the wave algorithms

Chapter 6

Simulation-based comparison

In this chapter, the previously computed bounds are verified by means of Monte-Carlo simulations. The implementation will first be discussed after which the simulation results are presented and evaluated.

6-1 Implementation

Similar to Chapter 5, we consider a multi-agent systems (MAS) consisting of N agents connected via network graph $\mathcal{G}$. We impose the same assumptions as in section 5-1, with the addition of:

Assumption 6.1.

1. *The memory capacity of agents is unlimited.*
2. *All agents have knowledge of the eigenvalues of the Laplacian matrix corresponding to the network graph $\mathcal{G}$.*
3. *All agents possess information about the network diameter.*

To summarise, these assumptions result in a group of agents that move within a two-dimensional space, are subjected to static communication delays, disregard collisions with each other, and achieve convergence when they are within a ϵ distance from the average consensus point.

The addition of Assumption 6.1.1 guarantees that the agents can always execute the phase algorithm, making Tarry's algorithm less effective. Therefore, the comparison in this chapter will focus on comparing the performance of the event-triggered consensus (ETC), periodic event-triggered consensus (PETC), and the phase algorithm.

Additionally, Assumption 6.1.2, allows for the optimal selection of the triggering parameters: α for the ETC algorithm and σ for the PETC algorithm.

Finally, Assumption 6.1.3 is only imposed for convenience. As discussed in Chapter 3, knowledge of the network diameter is not required for the execution of the phase algorithm. Instead, an estimate of the network diameter or the number of agents also suffices, which can be achieved through Assumption 6.1.

The implementation of all algorithms involves solving an initial value problem. To solve an initial value problems of ordinary differential equations, the *Python* programming language provides the *solve_ivp()* function from the *scipy.integrate* library [57]. As an additional feature it can also track the zero-crossing of an event function, which is required in order to detect the triggering events of the ETC and PETC algorithms. Once a zero-crossing of the triggering function is detected the solver will terminate, allowing for the updating of the broadcasted states.

6-1-1 Event-triggered consensus

The selection of the triggering parameters, c_0, c_1, and α have a significant impact on the convergence time and the number of triggering events. The triggering function ensures that the algorithm converges to a ball centred around the origin, this ball must be smaller than the error radius, ϵ. From proposition 5.4 we obtain that the algorithm converges to a ball with a radius:

$$r_{ETC} = \lim_{t\to\infty} ||\delta(t)|| \leq \frac{M||L||c_0\sqrt{N}}{\omega} \tag{6-1}$$

To ensure convergence, the value of c_0 must be smaller than:

$$c_0 \leq \frac{\epsilon\omega}{M||L||\sqrt{N}} \tag{6-2}$$

Additionally, according to Theorem 3.5, α must be less than the convergence rate ω. When selecting a value for α, there is a trade-off between reducing the number of communications and achieving faster convergence. Choosing a smaller α reduces the number of communications, but it can lead to a slower convergence. The same trade-off must be made when selecting a value for c_1.

Due to this trade-off, the selection of these parameters is decided heuristically, as presented in Appendix B-1.

6-1-2 Periodic Event-triggered consensus

The convergence guaranteed by Theorem 3.6 is limited to situations with small static time delays, where τ is less than the sampling period h. Larger time delays put more strain on the wave algorithms, making it more interesting to compare the performance under such circumstances. Therefore, Theorem 6.1 provides an extension to Theorem 3.6, demonstrating that the system can still achieve average consensus even when subjected to larger static time delays.

Theorem 6.1. *In the multi-agent system (3-4) with control law, $\dot{x}(t) = -L\hat{x}(t-\tau), \quad i \in \{1, ..., N\}$, in a connected undirected network $\mathcal{G}$, assume that the event-checking period h is shared by all agents, and the communication delay $\tau \in [(n-1)h, nh)$, with $n \in \mathbb{Z}^+$. If*

$$0 < \sigma < \frac{1}{\lambda_N}, \quad 0 < h < \frac{2(1-\lambda_N\sigma)}{n\lambda_N}, \quad 0 < \tau < \frac{1}{\lambda_N}\left(1 - \lambda_N\sigma - \frac{\lambda_N h}{2}\right) \tag{6-3}$$

with $\lambda_N = \lambda_{max}(L)$ and the event-triggering condition, with $t \in (0, h, 2h, ...]$:

$$||x_i(t) - \hat{x}_i(t)|| > \sigma|\sum_{j\in\mathcal{N}_i} \hat{x}_i(t) - \hat{x}_j(t)|_{min} \tag{6-4}$$

then all agents asymptotically converge to average consensus.

Proof. Proof in Appendix A. □

The implementation of the PETC algorithm is mostly influenced by the selection of the parameters, h and σ. It is important to choose values for these parameters that ensure convergence and allow the largest possible delay.

Theorem 6.1 indicates that a small sampling period allows for a larger time delay. In addition, a smaller sampling period can lead to faster convergence as the system receives more feedback. Considering a practical implementation with low-cost agents, it is reasonable to select a sampling period between 1 and 10 msec.

The selection of σ should guarantee stability. As per proposition 5.8, we know that the algorithm converges to a ball whose radius depends on the value of σ:

$$\lim_{t\to\infty} ||\delta(t)|| = \frac{M\sqrt{2}||L||\sigma|u^{max}|}{\omega} \tag{6-5}$$

Convergence is therefore always guaranteed when:

$$\sigma \leq \frac{\epsilon\omega}{\sqrt{2}M||L|||u^{max}|} \tag{6-6}$$

6-1-3 Phase algorithm

The implementation of the phase algorithm includes two states: initialisation and control. During the initialisation state, the algorithm computes the average of the initial positions using the pseudocode in Algorithm 1. Hereafter, the proportional controller (5-9) ensures the convergence of the agents.

In the simulations, we assume that the algorithm is initiated by a single agent. Adding more initiators will not affect the number of triggering events, but it may reduce the time required for the phase algorithm to execute. This is mainly because all agents start communicating at time $t = 0$, therefore they all receive the information earlier. In the best case scenario, all agents will initiate the algorithm at the same time, reducing the execution time to $D\tau$.

6-2 Monte Carlo simulation

As discussed in Chapter 5, the convergence time of the (periodic) event-triggered consensus algorithm is mainly influenced by the algebraic connectivity, the largest eigenvalue of the Laplacian, and the communication delay. The performance of the phase algorithm decreases when the network diameter, the number of agents, or the communication delay increases.

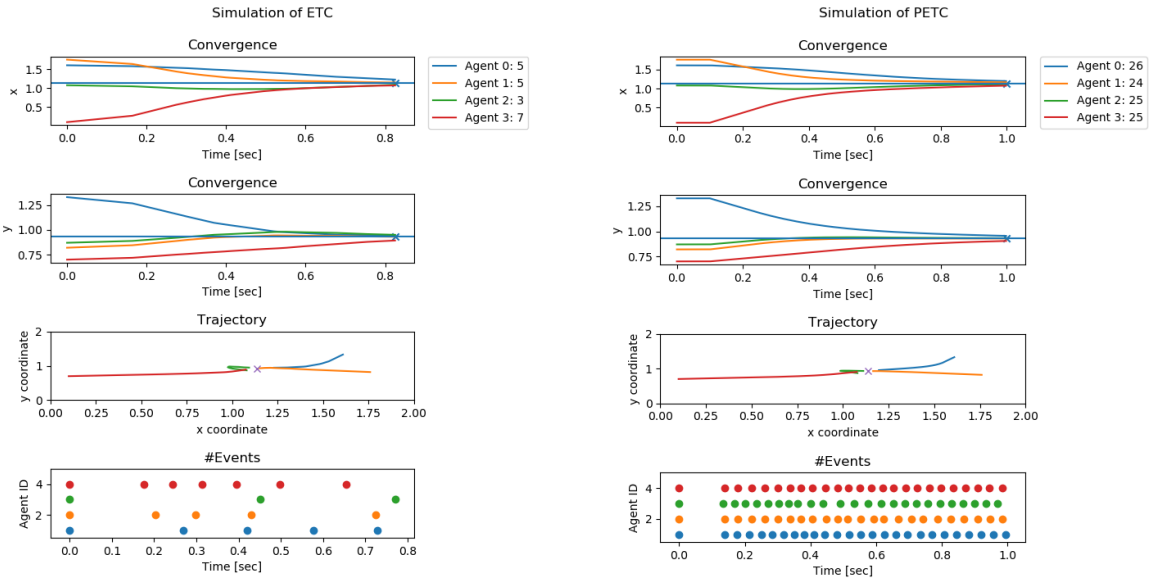
To verify the effects of these parameters and compare the performance of the three algorithms, simulations are performed with agents connected through different network graphs and subject to different static communication delays.

The initial position in combination with the network topology might also influence the performance of the ETC and PETC algorithms, therefore the experiments are conducted twice. One with randomly assigned initial positions. The other with the positions assigned according to the network topology.

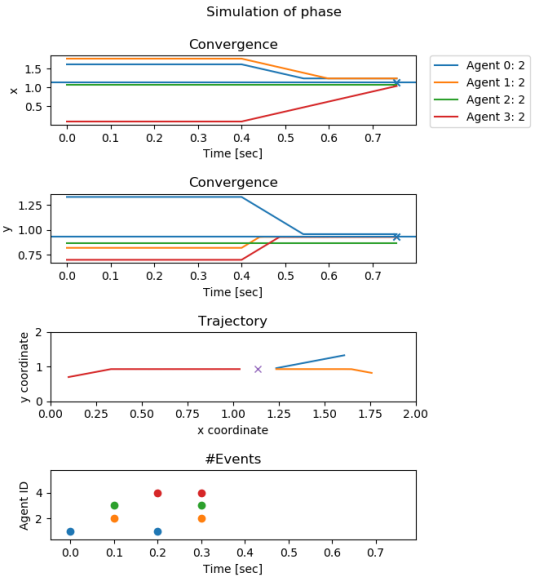
As a first result, Figure 6-1 shows the trajectory of 4 agents converging to the same position by using either of the three algorithms while experiencing a 10 msec communication delay.

The following observation can be made:

- The trajectory generated by both the ETC and PETC algorithms is the same, but the ETC algorithm requires fewer triggering events allowing it to converge faster.
- Both metrics indicate that the phase algorithm outperforms the ETC and PETC algorithms in this specific scenario.
- The phase algorithm ensures that agent 2, which is already located near the consensus point, does not move. On the other hand, in the event-triggered algorithms, agent 2 is influenced by its neighbours' feedback and moves away from the consensus point. Ultimately, it converges to a similar position, but this approach is not as efficient as the phase algorithm.
- The phase algorithm conserves more energy compared to the ETC and PETC algorithms. This is because it not only requires fewer triggering events, but it also preserves additional energy by disabling the communication system after $t = 0.4$sec and by having a more efficient trajectory. It is worth noting, that the performance metrics used to compare the algorithms in the subsequent sections only relate to the energy consumption due to the number of triggering events.



(a) ETC **(b)** PETC



(c) Phase

Figure 6-1: Trajectory and triggering events of 4 agents connected via a hypercube topology, where the number of triggering events per agent is given in the legend.

6-2-1 Network topology

The network topology that benefits the performance of the ETC and PETC algorithm is one with large algebraic connectivity, large diameter, and small cardinality of the nodes. Multiple network topologies were analysed in order to select a topology for the comparison, the results are summarised in Table 6-1.

Of these graphs, the line graph and the hypercube graph will be used in the comparison. The

line graph because of its large diameter and its robustness to communication delays.

The hypercube graph due to its large algebraic connectivity and relatively large diameter. The other graphs either have a quickly decreasing algebraic connectivity for larger networks and/or a small diameter.

Network graph	Diameter	Algebraic connectivity	Maximum degree
Line	N-1	$2(1-\cos\frac{\pi}{N})$ [23]	2
Cycle	N/2	$2(1-\cos\frac{\pi}{N})$ [23]	2
Hypercube (n-dimensional)	$\log_2 N$	2	$\log_2 N$
Watts–Strogatz (p)	$\log N$	< 1	
Caveman (n,k)	$4+\lfloor\frac{n}{2}\rfloor$	< 1	k
Star	2	1	N-1
Complete graph	1	N	N

Table 6-1: Analysis of network topologies

6-2-2 Randomised agent positions

The agents are randomly initialised in a 2x2 m square area. Once all agents are within a ϵ radius from the average consensus point the algorithms terminate. The value of ϵ throughout all simulations has been selected as 0.1 m, ensuring the agents are allowed to have a 10 cm deviation.

To obtain each result, 40 experiments were conducted and the average and standard deviation were computed over all runs.

Line graph

As previously discussed, for the best performance the triggering parameters need to be tuned for each network topology. The simulations results used for selecting the values are given in Appendix B-1-1, which resulted in the following values: $c_0 = 0.0003$, $c_1 = 0.25$, $\alpha = 0.5\omega$, $h = 0.002$ sec, and $\sigma = 0.04$.

Given these parameters, the PETC algorithm is stable if $\tau \leq 0.209$.

Figure 6-2 presents the simulation results, from which the following observations can be made:

- The phase algorithm is less affected by the low algebraic connectivity of the line graph, which allows it to outperform the ETC and PETC algorithms in both metrics. Only in the case of a small network do the ETC and PETC algorithms display comparable performance.
- The ETC algorithm triggers more efficiently compared to the PETC algorithm.
- The variation in the convergence time of the phase algorithm is minimal and unnoticeable in Figure 6-2. Nonetheless, the variation remains within the lower and upper bounds.

- The bounds as computed in Chapter 5 capture the trend of the metrics and the experimental results fall within the bounds. However, the bounds of the ETC and PETC algorithms are conservative and seem to cover situations that are unlikely to happen.
- The selection of the PETC parameter σ was made to limit the number of triggering events, rather than satisfying equation (6-6). Consequently, the upper bound on the convergence time is infinite as convergence to the ϵ-neighbourhood of the average consensus is no longer guaranteed. This is elaborated in Appendix B-1-1, where it is noted that experiments indicated the bound to be conservative and that larger values of σ can still achieve convergence.

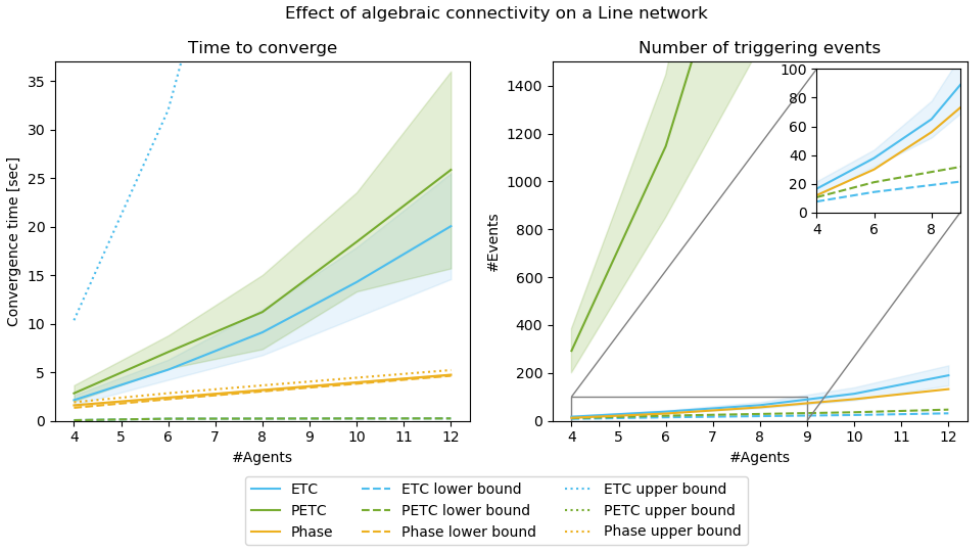


Figure 6-2: Agents connected via a line graph, with $\tau = 0.2$, $\tau_{max}^{ETC} = 0.39$, $\tau_{max}^{PETC} = 0.209$, and $\epsilon = 0.1$

Hypercube graph

The algebraic connectivity of a hypercube graph is higher than that of a line graph and does not decrease when the number of agents increases. However, it is not robust to time delays, and the diameter only slightly increases for larger networks.

We verify which of these factors influences the performance the most by running the algorithms on a hypercube topology with 16 agents subjected to different communication delays.

The triggering parameters for this simulation were selected based on the simulations given in Appendix B-1-2, which resulted in the following values: $c_0 = 0.0003$, $c_1 = 0.25$, $\alpha = 0.5\omega$, $h = 0.002$ sec, and $\sigma = 0.04$.

From the simulation results in Figure 6-3, the following can be concluded:

- When the communication delay nears the optimal value $\tau^* = 0.12$ sec (from Theorem 5.2), the ETC algorithm converges on average faster than the others. Hereafter, the performance degrades as the convergence rate decreases for larger communication delays.

- For small time delays, $\tau < 0.04$ sec, the ETC algorithm requires on average slightly fewer triggering events compared to the phase algorithm. However, as demonstrated in Figure 7-6, the energy consumption of the phase algorithm might still be lower as the communication system is only active during the execution of the phase algorithm, $t \leq 2D\tau$, which is less than half of the convergence time.
- Theorem 6.1 guarantees the convergence of the PETC algorithm only when the time delay, τ, is less than or equal to 0.084 sec. Nonetheless, the experiments demonstrate that the system remains stable even for larger delays.
- Interestingly enough, when the time delay is larger than 0.16 sec, the PETC algorithm performs better compared to the ETC algorithm. This can be explained by the triggering parameter α. For larger delays, the convergence rate of the system is lower, resulting in a smaller value of α. As a consequence, the triggering condition is not as strict, causing the agents to move more before triggering. As a result, the trajectory of the agents becomes less efficient, which leads to a longer convergence time and eventually an increase in the number of triggering events.
- The mathematical upper bound of the ETC algorithm still captures the trend of the convergence time. Furthermore, these results indicate that the upper bound on the convergence time is less conservative than previously assumed.

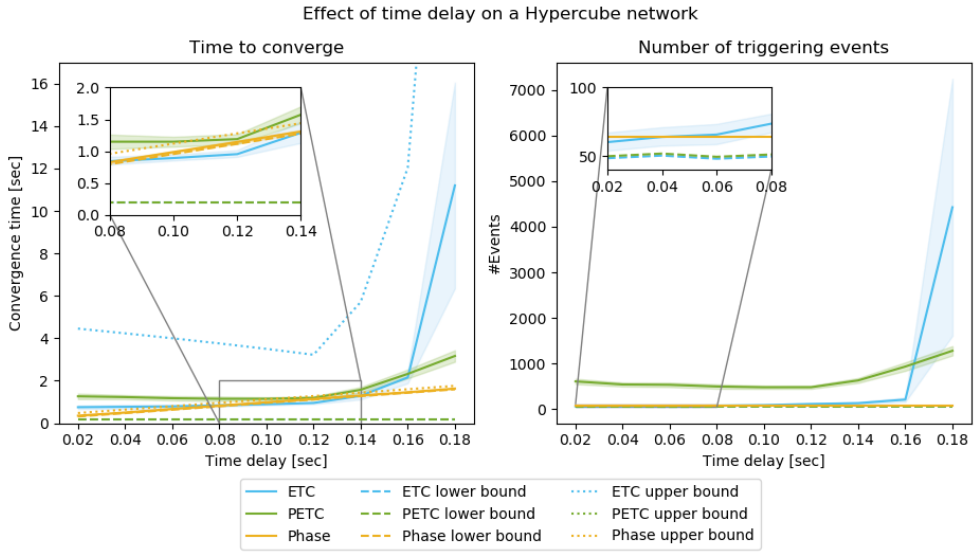


Figure 6-3: Agents connected via a Hypercube graph, with $N = 16$, $\tau_{max}^{ETC} = 0.19$, $\tau_{max}^{PETC} = 0.084$, and $\epsilon = 0.1$

These results already show that there are some configurations where using the ETC algorithm is beneficial, if fast convergence is required. To verify if this is still the case for systems with various sizes, similar simulations were performed with a varying network size. The results of these simulations are presented in Figure 6-4 and 6-5 and demonstrate:

- The number of triggering events required by the ETC algorithm is comparable to that of the phase algorithm when the swarm consists only of a few agents (less than 8).

However, one should still keep in mind that the total energy consumption of the ETC algorithm will be higher.

- All configurations have a range of communication delays, in which the ETC algorithm outperforms the other two algorithms with respect to the average convergence time. For instance, the ETC algorithm performs slightly better when a network of 64 agents experiences communication delays ranging from 0.08 to 0.1 sec.

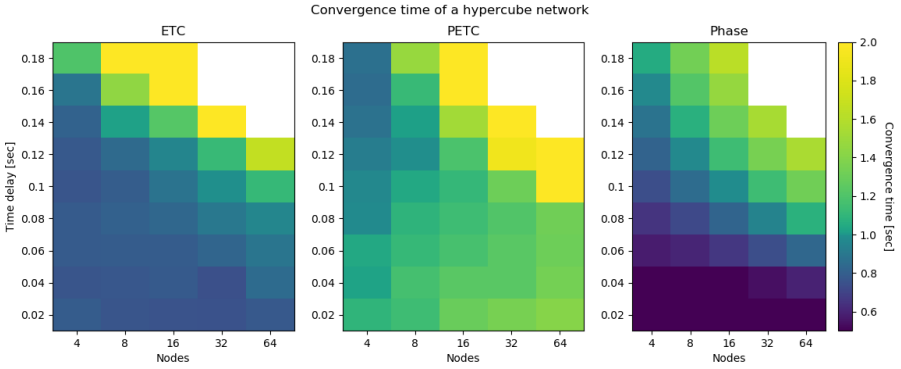


Figure 6-4: Average convergence time of agents connected via a hypercube graph, with $\epsilon = 0.1$

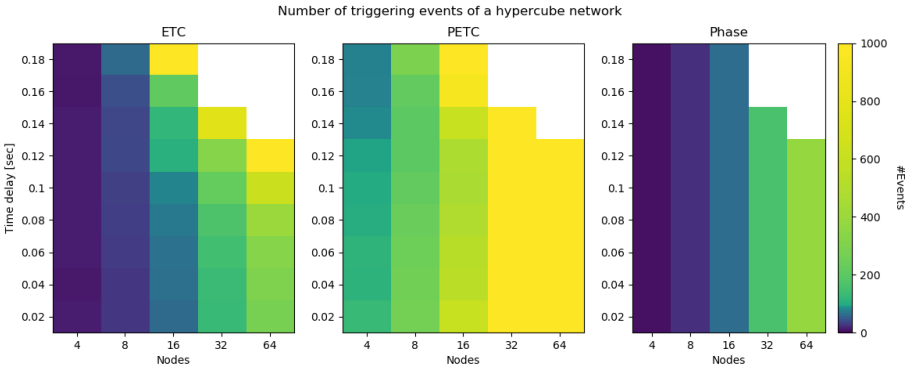


Figure 6-5: Average number of triggering events of agents connected via a hypercube graph, with $\epsilon = 0.1$

Impact of the initial area

The previous simulations have all been done in a fixed 2x2 m area with a fixed error radius ϵ. The size of the area and allowed error were selected to match the application discussed in Chapter 7.

However, a larger error radius, ϵ, and/or a smaller area should positively impact the performance of the ETC and PETC algorithms. On the other hand, it should have little impact on the performance of the phase algorithm.

We evaluated the impact of these parameters on the comparison by running the algorithms for 16 agents connected via a hypercube topology and subjected to a communication delay of $\tau = 0.1$ sec.

The simulation results are given in Appendix B (Figures B-5 and B-6). They confirm the relation as obtained from the upper and lower bounds in Chapter 5. Furthermore, they show that the performance of the phase algorithm is minimally impacted by a change in the area or error radius. Therefore, by increasing the allowed error or by decreasing the area there will always be a situation in which the ETC and PETC algorithms will perform better than the phase algorithm.

6-2-3 Network-based agent positions

As previously stated, the performance of the ETC and PETC algorithms may be influenced by the initial position and network topology. In robotic applications, agents will communicate with those nearest to their own location. Consequently, it would be interesting to compare results obtained when agents are positioned according to the network topology.

In terms of implementation, the placement of agents is achieved through the use of the *spring_layout()* function that is included in the *NetworkX* package [50]. This function applies the Fruchterman-Reingold force-directed algorithm to determine the positions of nodes within a graph. It treats the nodes in a graph as repelling objects and edges as springs that keep nodes close together.

Similar to section 6-2-2, the agents are initialised in a 2x2 m area and convergence is achieved when the agents are within a 10 cm radius of the consensus point.

Line graph

The simulation results, in Figure 6-6, confirm the observations made in section 6-2-2. Additionally, the results show that network-based positions do not improve the performance of the algorithms when agents are connected through a line topology. This can be explained by the fact that neighbouring agents are closer together and remain closer, leading to a smaller average control input. Consequently, it takes the agents more time to converge.

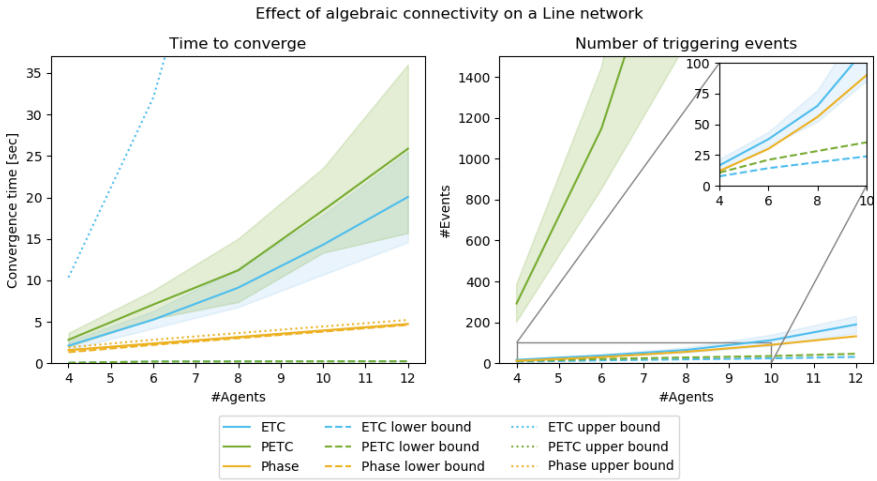


Figure 6-6: Agents connected via a line graph, with $\tau = 0.2$, $\tau_{max}^{ETC} = 0.39$, $\tau_{max}^{PETC} = 0.209$, and $\epsilon = 0.1$

Hypercube graph

The simulation results with agents positioned based on a hypercube topology, as shown in Figure 6-7, differ from the results obtained with randomly positioned agents. The following observations can be made:

- For both the ETC and PETC algorithms, increasing the time delay up to 0.16 sec is beneficial. This a result of the geometry of a hypercube graph and the lack of collision avoidance. The agents are positioned in such a way that they can move in a straight line to the average consensus point, which is the centre of the hypercube. Since the agents move in a straight line and never have to correct their trajectory, larger time delays cause the agents to adjust their control input later. Therefore, they move faster towards the consensus point.
- The execution time of the phase algorithm is not impacted by the initial position of the agents. Consequently, the convergence time of the phase algorithms is not impacted as much by the hypercube's geometry, resulting in the faster convergence of the ETC and PETC algorithms for a larger range of delays compared to the previous results in Figure 6-3.
- Compared to the phase algorithm, the ETC algorithm requires a similar amount of triggering events if the delay is smaller than 0.16 sec. However, as previously highlighted the ETC algorithm may still consume more energy.
- The mathematical bounds remain valid and capture the behaviour of the system. Additionally, the results indicate that the lower bound for the number of triggering events is not as conservative as previously assumed.

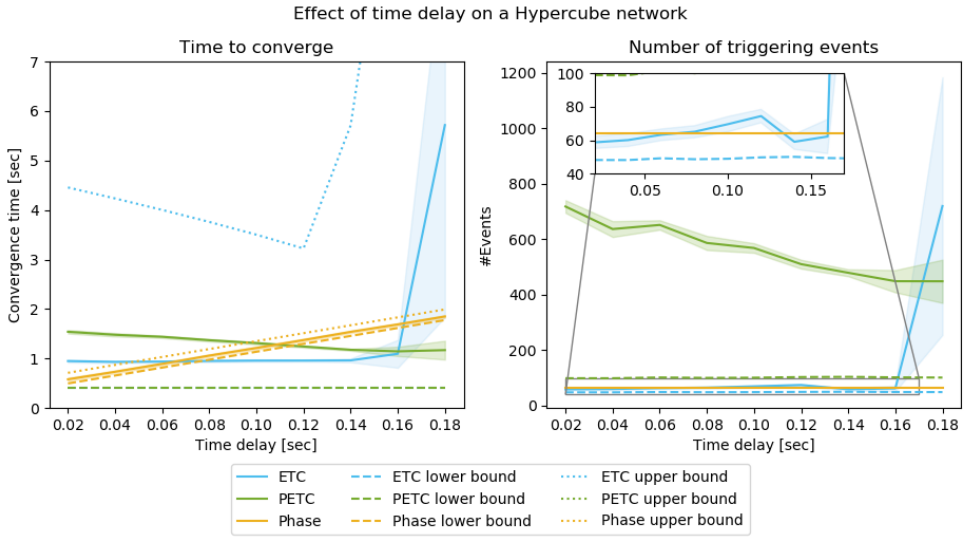


Figure 6-7: Agents connected via a Hypercube graph, with $\tau_{max}^{ETC} = 0.19$, $\tau_{max}^{PETC} = 0.114$, and $\epsilon = 0.1$

6-3 Discussion

The findings presented in this chapter show that the network topology, time delay, area, and error radius have a similar impact on the performance as the bounds computed in Chapter 5. We see that with our specific set-up, which initiates agents in a 2x2 m area and permits a 10cm deviation, the ETC algorithm may converge the fastest when the network has a large algebraic connectivity, a relatively large diameter, and a small maximum degree.

There are only a few network topologies that satisfy these criteria, the hypercube topology is one of the few. Simulation results show that even with such a communication network, the ETC algorithm is only slightly faster than the phase algorithm and only if the system is subjected to a particular range of communication delays. Performance is significantly improved when the initial placement of the agents corresponds with the network topology. However, this is a unique situation caused by the geometry of a hypercube and the lack of collision avoidance. Furthermore, in most cases, as demonstrated by simulations with a line network, the ETC and PETC algorithms will perform worse.

Comparing the PETC algorithm to the ETC algorithm confirms the intuition that periodically checking the triggering function does not benefit the performance unless the communication delay is larger than 160 msec. Therefore, if the agents are unable to continuously monitor the triggering function, the ETC algorithm is the better option.

On the other hand, the simulation results of the phase algorithm show that the convergence time is mainly dominated by the execution time of the algorithm, $t_{algorithm}$. Additionally, the execution time of the phase algorithm, and thus the convergence time, may reduce significantly if more agents initiate the algorithm. Therefore, it is worth considering using the phase algorithm if fast convergence is required.

Finally, the energy usage of the wireless communication system can be compared based on the number of triggering events required for each algorithm. In most situations, the phase algorithm requires fewer triggering events. Additionally, the phase has even more of an advantage over the other algorithms, due to the fact that agents find a more efficient trajectory and only require the communication system to be active for part of the convergence time. In contrast, the ETC and PETC algorithms often require more triggering events and transmissions occur throughout the entire trajectory.

Chapter 7

Application-based comparison

In this chapter the three algorithms are implemented on realistic agents that are required to assemble in a specific formation while avoiding collisions in a physical environment. We present experiments with the algorithms running on real Elisa-3 robots (GCTronic [44]), and a more extensive analysis with the realistic simulation environment *Webots* [77].

7-1 System requirements

In the previous chapters we have implicitly assumed the following:

- Agents have continuous access to their global position
- Agents have some measure of angular orientation
- Agents have enough memory to store scalars and vectors
- Agents have enough computational power to perform vector addition and multiplication operations
- Agents have the ability to send and receive signals with a minimum range of 2m
- Agents have the ability to detect obstacles
- Agents have the ability to rotate around their own axis

7-2 System Overview

The Elisa-3 robot has characteristics that satisfy most of these assumptions and has the benefit of having a realistic model within the *Webot* simulation environment [22]. Figure 7-1 gives an overview of the main hardware components.

- 8 infrared proximity sensors (PS), with a 6cm range
- 3 IR emitters
- RF radio communication, 2.4 GHz
- Atmel ATmega2560, 8MHz (8 MIPS)
- 256kB flash memory
- 3-axis accelerometer
- 2 DC motors
- An indicator LED
- Rotational selector with 16 positions

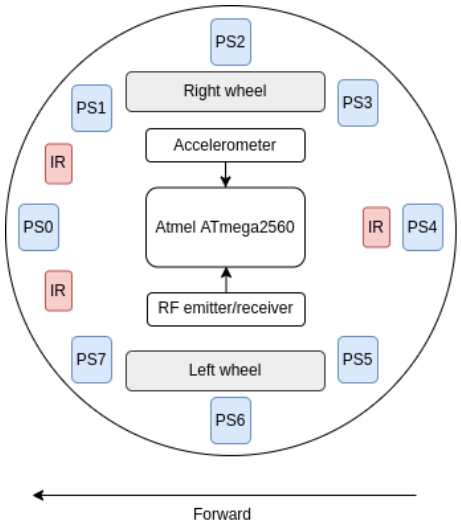


Figure 7-1: Elisa-3 hardware overview

Position and orientation

As the Elisa-3 robots lack a compass, gyroscope or GPS module, the only means of determining the position and orientation is through odometry data. However, odometry readings quickly accumulate an error, resulting in an incorrect estimation of the robot's location and orientation.

In the *Webot* simulation this problem can be resolved by including a GPS module in the robot model provided by GCTronic [22]. The GPS module allows the inclusion of Gaussian noise, which can be used to duplicate the inaccuracies of sensor measurements. Additionally, to obtain the orientation of the robots, a central unit, referred to as the "supervisor" is used. The supervisor keeps track of all the robots and has access to the robots' orientation.

Initial testing carried out with real Elisa-3 robots, as presented in Appendix B-4-1, confirmed the intuition that the odometry data is unreliable and contains a significant amount of accumulated error. Consequently, the position and orientation estimation in the practical implementation was improved by using the motion capture technology OptiTrack [2]. The work by Li, Y. [62], was used to merge the data obtained from the OptiTrack system with the odometry data from the Elisa-3 robots via a Multi-Rate Extended Kalman Filter (MR-EKF), resulting in a more accurate estimation of the position. The estimation of the position was also used to estimate the orientation of the robots.

Motor control

The movement of the robots is limited to either moving in a straight line or rotating around its own axis. By limiting the motion of the robots, we can now model it as a single-integrator and the consensus algorithms still apply.

Distance sensors

The Elisa-3 robots are equipped with 8 infrared proximity sensors, capable of detecting objects within a 6 cm range. Each sensor is positioned at a 45° angle from the others, allowing the

robot to detect any object.

In the *Webot* model, the infrared sensors have a very limited field of view, and are only able to detect objects in direct alignment with the sensors. Consequently, this makes it challenging to detect and navigate around smaller obstacles, such as other Elisa-3 robots. This may reflect in the results of the *Webot* simulation.

Wireless communication

The Elisa-3 robots are equipped with nRF24L01+ chip. This chip is a 2.4 GHz transceiver that enables wireless communication between the robots and a pre-programmed base-station. The base-station acts as a gateway between the robots and a PC, allowing information exchange between the robots and the OptiTrack system.

However, the chip does not allow direct communication between robots. Therefore, all the information exchange between robots will be handled through the PC.

The PC can only poll information from the base-station once per 4 milliseconds. Furthermore, due to the 64-byte buffer between the PC and the base-station, communication throughput is limited to 64 kBps. To overcome this limitation GCTronic has developed an optimised communication protocol [44], where each message contains information for 4 robots, resulting in an update frequency of 250Hz for 4 robots.

This protocol was adjusted to suit our application. Appendix C contains a detailed description of the adjusted communication protocol.

7-3 Software architecture

The software architecture discussed in this section is only related to the software running on the robots. Both the simulation and the real-world implementation use the same software architecture.

7-3-1 State machine

The software architecture for all three algorithms is centred around the Finite State Machine (FSM) in Figure 7-2. The three algorithms have four identical modes of operation; idle, angle_control, formation_control, and avoid_obstacle. The FSM used by the phase algorithm has been extended with two additional modes of operation; initialise and terminate.

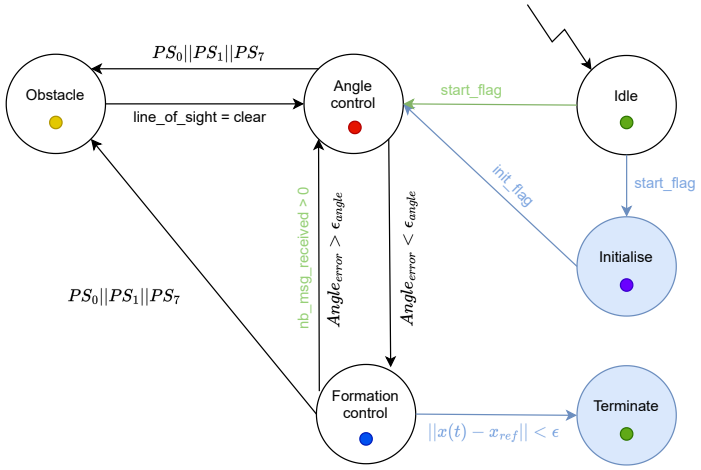


Figure 7-2: State diagram for Elisa-3, where green transitions are only included in the FSM of the event-triggered consensus (ETC) and periodic event-triggered consensus (PETC) algorithms, and blue states and transitions are only included in the FSM of the phase algorithm. The colored circles in states represent the color of the LED

The tasks executed in each state differ slightly based on the type of algorithm. To switch between each FSM, the rotational selector is used:

- Position 12 : PETC algorithm
- Position 13 : ETC algorithm
- Position 14 : Phase algorithm

In the rest of this section, we provide a short description of the tasks performed in some of the states. More detailed explanations of the obstacle and formation_control states can be found in sections 7-3-3 and 7-3-4, respectively.

Idle

The system always starts up in an idle state, with the motors turned off to ensure safety. Once the sensors have been initialised and calibrated, user input can be used to start the algorithm and either transition to the angle_control or initialise state.

Initialise

In this state, the phase algorithm is executed. Once a robot has received the positions of all the active robots, it computes the average and uses this value to determine the optimal formation.

Angle control

The FSM transitions to this state when the robot's current orientation does not align with the direction towards the reference position. A simple proportional controller is used to adjust the orientation.

Terminate

Once a robot is ϵ close to its position in the formation it will transition to this state. Here it will become idle and will wait until the other robots have reached their positions.

7-3-2 Task scheduling

The software executes all processes sequentially using Round Robin scheduling, without interrupts.

In the *Webot* simulation the loop frequency and therefore the control frequency is limited to 125 Hz. Increasing the loop frequency requires more processing power and significantly increases the real-time simulation duration.

In the setup with real Elisa-3 robots, the loop frequency is limited by the execution time of each task. To measure the loop frequency and thus the control frequency, timestamps were included in the firmware of the robots. Since the code is executed sequentially, measuring the difference between these timestamps corresponds with the execution time of specific tasks. The profiling results are given in Figure 7-3 and have been obtained by using a million measurements. From the results we can conclude the following:

- The maximum control frequency of the algorithms is: 175 Hz for the ETC algorithm, 185 Hz for the PETC algorithm, and 215Hz for the phase algorithm.
- The average computation time required by the triggering function of the ETC algorithm is more than that of the PETC algorithm, which is attributed to the time required to compute an exponential.
- The execution time of the angle controller is mainly a result of the computational complexity of the arctangent.

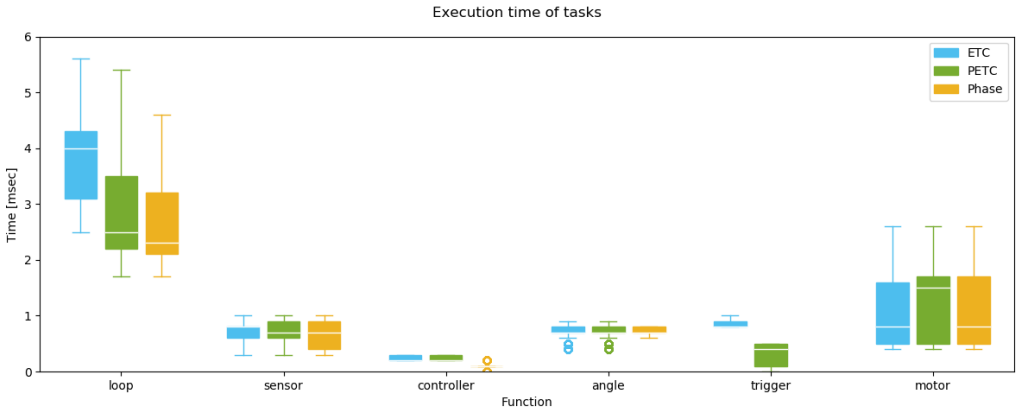


Figure 7-3: Response time of the Elisa-3 robots

7-3-3 Obstacle avoidance

In order to avoid collisions with other robots the implementation of an obstacle avoidance algorithm is essential. There are different ways to approach robot navigation when motion planning and obstacle avoidance pay a key role. The choice of a specific method depends on the task at hand and the available sensor data. Based on the analysis by Oroko and Nyakoe [91], the Bug algorithm best fits our application. The Bug algorithm is simple to implement, requires low computational power, and can be used with proximity sensors that have a limited field of view.

The Bug algorithms simply iterate over the following steps:

- Move in a straight line towards the reference point.
- If an obstacle is encountered orientate parallel (by rotating either left/right) to the obstacle surface.
- Keep following the obstacle.

There are multiple versions of the Bug algorithm which differ depending on the condition used to switch between obstacle-following and moving-to-goal behaviour. The Bug 0 algorithm involves following the obstacle until the robot can continue moving towards the goal. While this algorithm is considered greedy and may fail when navigating through a maze, it will suffice for our application since our only requirement is to avoid collisions with other robots.

Figure 7-4 represents the path generated by the Bug 0 algorithm when it encounters two obstacles.

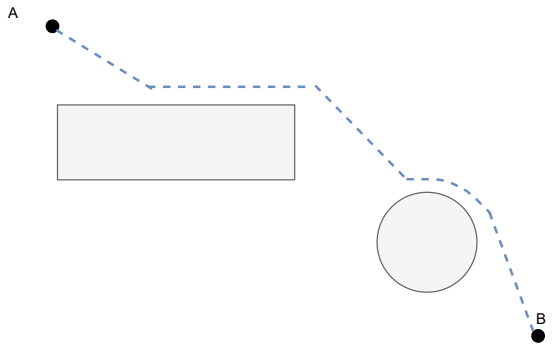


Figure 7-4: Path planning from A to B by the Bug 0 algorithm

Note that the Bug 0 algorithm is used for path planning, where it is assumed the obstacles are static. In our application, the robots are required to also avoid moving obstacles, namely the other robots in the swarm. In most cases, the Bug 0 algorithm will be able to handle avoiding moving obstacles. There is a single situation where a problem might occur. This situation is depicted in Figure 7-5.

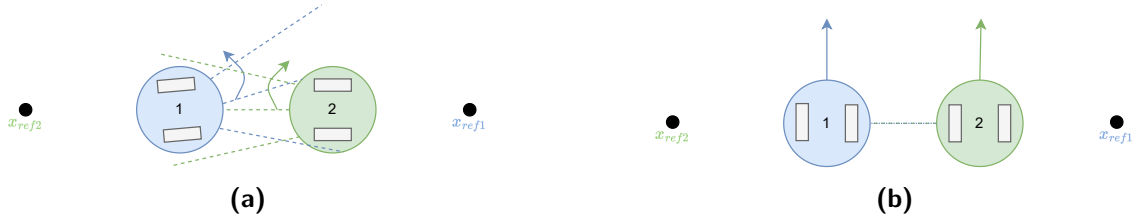


Figure 7-5: Collision avoidance. Figure 7-5a illustrates the moment the collision is detected. Figure 7-5b illustrates the movement of the robots to avoid the collision. The dashed lines represent the field of view of the proximity sensors.

In general, this situation will rarely occur since the robots will by default rotate clockwise if their left or right proximity sensors do not detect an object. However, as Figure 7-5a illustrates, robot 1 detects part of robot 2 with its right proximity sensor. Therefore, instead of rotating clockwise, it will rotate in the counter-clockwise direction. This leads to the situation depicted in Figure 7-5b. Once the robots are in this configuration they both will continue detecting a "wall" and move straight.

In practical situations, this issue can be solved by letting the robots communicate the direction they are turning. Since the robots are close proximity during an almost collision, they should be able to communicate with each other. For example, the Elisa-3 robot can use the IR emitters to transmit the turning direction.

7-3-4 Formation control

Assembling agents into a formation can be achieved by using a consensus algorithm to converge to the centre point of the formation and adding a bias term, b_i, to converge to the actual formation. Depending on the type of algorithm this bias represents a different quantity.

(Periodic) Event-triggered control

To achieve the formation, the consensus protocol needs to include a bias, b_{ij}, which represents the desired inter-agent relative distance between agent i and agent j in the formation:

$$u_i(t) = -\sum_{j \in \mathcal{N}_i} ((\hat{x}_j - \hat{x}_i(t)) + b_{ij}) \tag{7-1}$$

Phase algorithm

The phase algorithm uses a proportional controller to converge to a reference position x_{ref}, which is the average of the initial positions. To ensure the phase algorithm converges to the desired formation, each agent needs to update the reference position by including a bias, b_i, that represents the desired distance from the centre point of the formation:

$$u_i(t) = -K(x_i(t) - (x_{ref} + b_i)) \tag{7-2}$$

The performance of the phase algorithm may be further improved by allowing the robots to select the position within the formation that optimises the trajectories of all robots. A

simple method to accomplish this is to select the position that is closest to each agent. To prevent two agents from occupying the same position, the agent with the higher ID chooses the second-closest position.

A method to find a more efficient formation would be to use an optimisation algorithm, such as the Earth's movers distance algorithm [70]. However, these algorithms are computationally intensive and cannot be executed on the robots themselves.

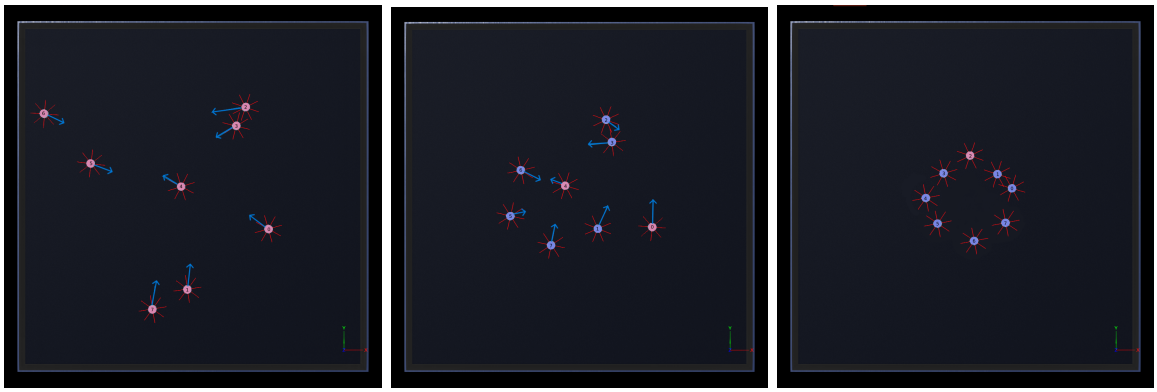
7-4 Simulation results

From the results of previous chapters, we know that the network topology, communication delay, and initial placement impact the performance. Furthermore, there are specific situations for which the ETC algorithm might converge faster. The use of realistic robots for formation control introduces additional factors that increase the convergence time. These factors include the additional time required to assemble in a specific formation, the need to avoid collisions with other robots, and the extra time required for rotation.

These additional effects, change the trajectory and will increase the minimal convergence time of all three algorithms. To verify the influence of these effects on the performance of the algorithms, three different simulations have been carried out. The results of these experiments have been obtained by using 15 Monte Carlo simulations, where the robots are placed in a 2x2 m area and have converged once they are within 10cm of the desired formation.

To ensure a fair comparison with the results in Chapter 6, the same triggering parameters have been used with the exception of the sampling period, h. This parameter is limited by the loop frequency and is therefore equivalent to 8ms.

As a first result, Figure 7-6 shows snapshots of the trajectory of 8 robots connected through a hypercube graph required to converge to a circle with a radius of 20 cm by using the ETC algorithm. In the figure, the blue arrows represent the direction in which the agents will move and the 8 red lines represent the field of view of the proximity sensors.



(a) $t = 0$sec **(b)** $t = 40$sec **(c)** $t = 70$sec

Figure 7-6: Trajectory of 8 agents connected via a hypercube topology

7-4-1 Randomised agent positions

In Chapter 6 we observed that if the agents are connected through a hypercube graph and are subjected to a communication delay larger than 80 msec, the ETC algorithm might converge faster than the phase algorithm. To verify if this is still the case in a realistic application, we subject the system to a 100 msec time delay and vary the number of agents. This delay was chosen as a larger communication delay might render the PETC algorithm unstable.

In all simulations, the robots are required to converge to a circular formation. This formation was selected as the bias terms, b_{ij}, can easily be computed for different networks with the *circular_layout()* function from the *NetworkX* library [50].

The result of the simulations has been depicted in Figure 7-7.

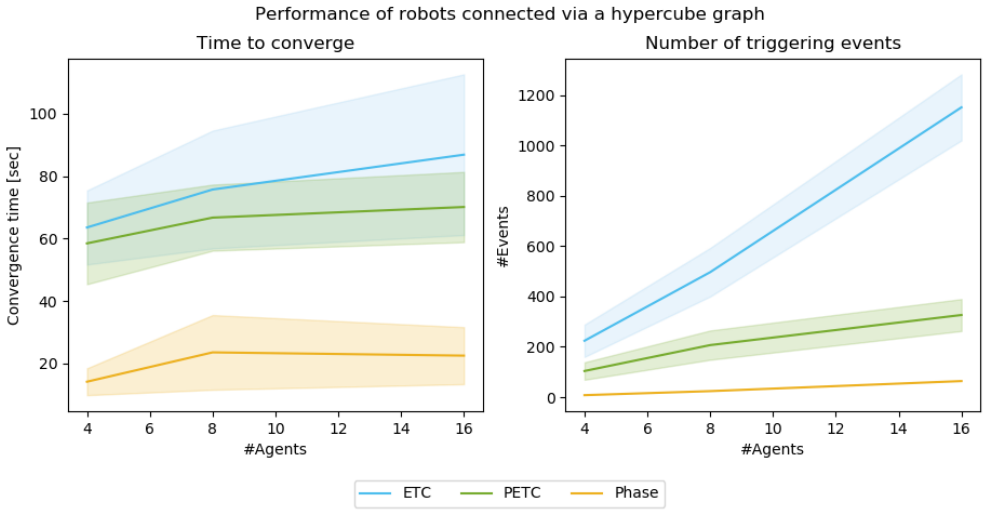


Figure 7-7: Agents connected via a Hypercube graph, with $\tau = 0.1$, and $\epsilon = 0.1$

From these results, we can observe the following:

- The time required for the execution of the phase algorithm, $t_{phase} = 2D\tau$, is insignificant compared to the entire convergence time.
- The trajectory of the phase algorithm is more efficient compared to the trajectory of the ETC and PETC algorithms.
- The ETC algorithm requires more time to converge than the PETC algorithm. This difference can be explained by two factors: firstly, the finite loop frequency of the ETC algorithm makes it impossible to continuously monitor the triggering function. Secondly, the ETC algorithm has more triggering events and therefore a lower average control input.
- In Chapter 6, the results showed that the PETC algorithm required significantly more triggering events than the ETC algorithm, which is opposite of the current observation. This is mainly due to the longer convergence time, which indicates that the triggering

parameter of the ETC algorithm should be adjusted according to the situation. Furthermore, the PETC algorithm has a larger sampling time causing it to trigger less compared to the results in Chapter 6.

- The mathematical bounds as computed in Chapter 5 no longer hold. The bounds do not take into account the time required for the angle controller to rotate in the right direction. Furthermore, the simulation now includes collision avoidance, which is also not taken into account by the bounds.

It should be noted that simulations with more agents will not offer new insights because the execution time of the phase algorithm is no longer a significant part of the overall convergence time.

Similar simulations were conducted with agents connected via a line topology, and the results are presented in Appendix B-3-1. The findings from these simulations are consistent with the previous observations and the conclusions drawn in Chapter 6.

7-4-2 Network-based agent positions

Using network-based initial positions greatly improved the performance of the ETC and PETC algorithms in Chapter 6. This was mainly due to the fact that the agents moved in a straight line to the average consensus position without avoiding other agents.

The results of the realistic simulation, in Figure 7-8, show that when collision avoidance is included, the performance of the ETC and PETC algorithm is on average similar to initiating the agents with random positions.

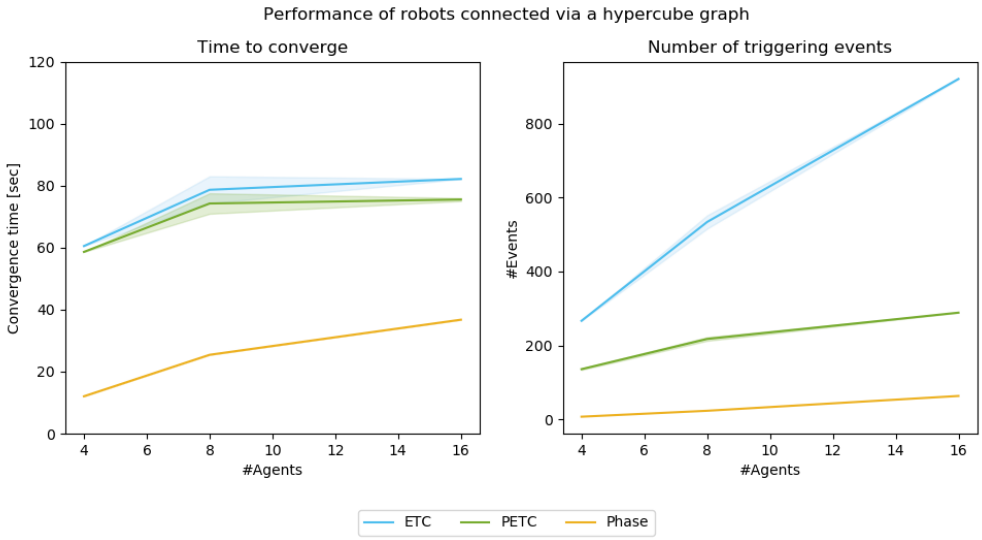


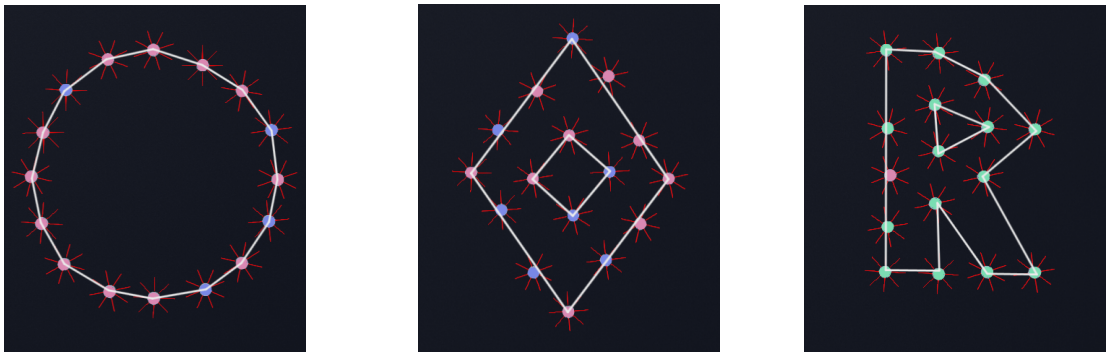
Figure 7-8: Agents connected via a Hypercube graph, with $\tau = 0.1$, and $\epsilon = 0.1$

7-4-3 Different formations

In the previous simulations, the robots were required to assemble in a circular formation. The type of formation influences the trajectory of the robots, and consequently can also influence

the convergence time and the number of triggering events.

To verify the effect a specific formation has on these metrics, three simulations were conducted where a group of 16 robots was instructed to assemble in one of three shapes depicted in Figure 7-9.



(a) Circle **(b)** Diamond **(c)** Letter R

Figure 7-9: Formation structures for a swarm of 16 robots

The results shown in Figure 7-10 demonstrate the following:

- The diamond and letter shapes are denser, which can result in more collisions compared to the circular formation. This is reflected in the convergence time required by the phase algorithm. However, the formation has less impact on the convergence time of the ETC and PETC algorithms.
- In all three formations, the phase algorithm still finds the most efficient trajectory and requires the least amount of triggering events.

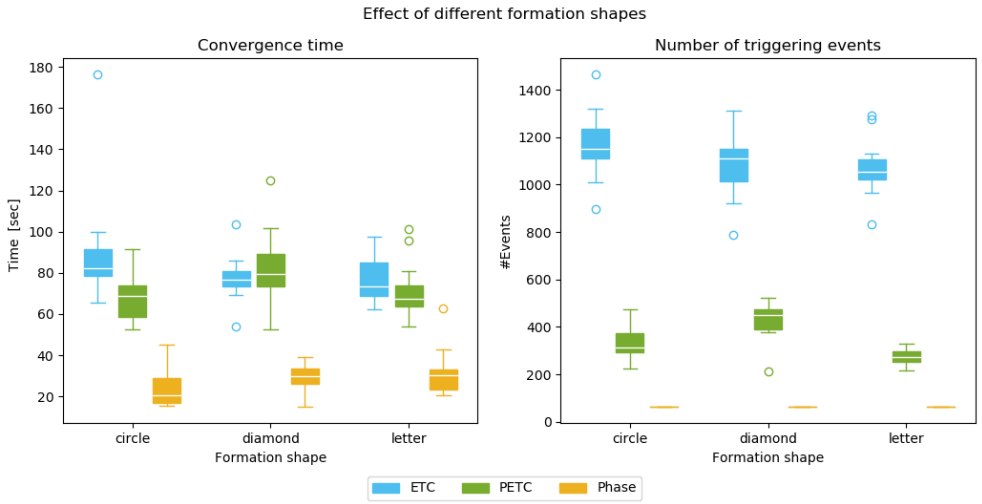


Figure 7-10: Performance of 16 agents connected via a Hypercube graph required to assemble in one of three formations (circle, diamond, or letter), with $\tau = 0.1$ and $\epsilon = 0.1$.

Similar simulations were carried out with 8 agents, and the results of these simulations, which are provided in Appendix B-3, lead to comparable observations. Therefore, we conclude that the type of formation does not have a significant effect on the comparison.

7-5 Real-world results

The algorithms were also implemented on real Elisa-3 robots. This implementation consists of two separate parts, one running on the PC and the other one on the robots. The software running on the robots is the same as used for the simulation and has already been discussed in section 7-3. In this section, we will first discuss the software used on the PC. Subsequently, we will discuss the experimental outcomes.

7-5-1 Software architecture on the PC

The software on the PC handles the communication between robots, as described in section 7-2, and merges the information from both the OptiTrack system and the robots. An overview of the software architecture is provided in Figure 7-11.

The architecture is divided into three components: the Elisa-3 ROS node, the Python API and the RViz visualisation. This division happened because GCTronics provided the framework for the Elisa3 node [45], while the Kalman filter was already implemented in Python by Li, Y. [62].

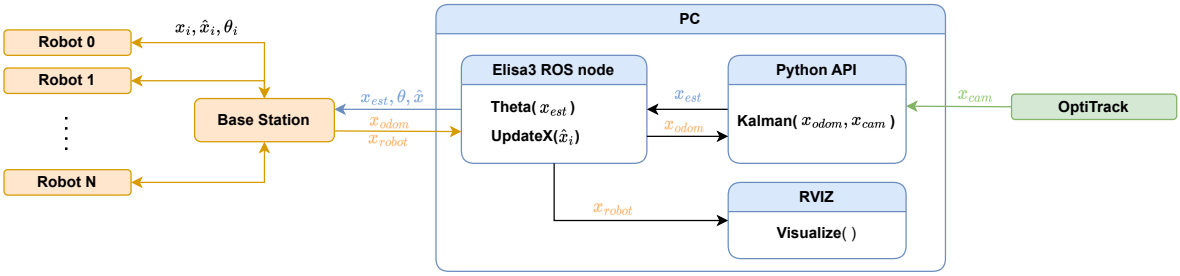


Figure 7-11: Software architecture on PC

The Elisa3 node was adjusted to fit our application. It uses both the past and current positions received from the Kalman filter to estimate the orientation of robots. These estimations are used to correct the odometry of the robot. Additionally, the node keeps track of the last broadcasted position, $\hat{x}_i(t)$, of all robots and sends updates to the neighbours of an agent when it triggers.

RViz is used to visualise the trajectory the robots believe they are following. This visualisation was mainly included for debugging reasons.

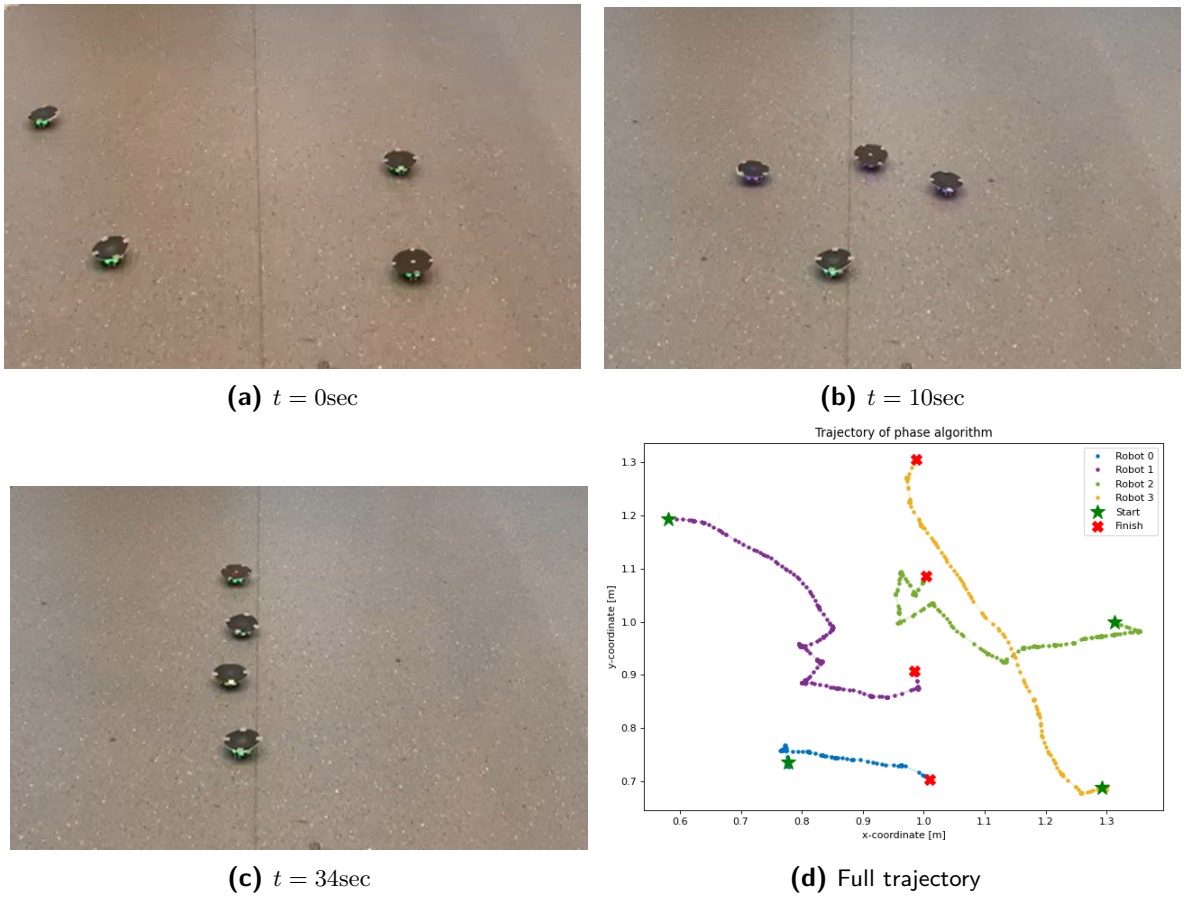
7-5-2 Experimental results

To verify the results obtained via the realistic simulation a swarm of 4 robots was deployed in a 2x2m area. The robots experience a communication delay of approximately 30 msec.

The estimation of the communication delay is based on the experimental results presented in Appendix B-4-2.

Two experiments were performed, the first requires the robots to converge within a 5cm proximity of a circular formation with a radius of 20cm, and the second required them to assemble in a vertical line with a 20cm inter-agent distance.

As a first result, Figure 7-12 shows snapshots of Elisa3 robots executing the phase algorithm to assemble in the line formation. The full trajectory of the robots was captured by the camera and is given in Figure 7-12d.



(a) $t = 0$sec **(b)** $t = 10$sec

(c) $t = 34$sec **(d)** Full trajectory

Figure 7-12: Swarm executing the phase algorithm to converge to a line formation

An overview of the outcomes from all experiments is presented in Table 7-1. It is important to note that these results are based on a single run per algorithm, the main reason for this is the limited time available for the experiments.

Furthermore, a video showcasing some of the experiments is available for reference alongside this overview. The link to the video can be found here: https://youtu.be/hGG9CEUyy_A

Network graph	Formation	ETC		PETC		Phase	
		t (sec)	C	t (sec)	C	t (sec)	C
Hypercube	Circle	28	134	30	280	27	8
	Line	27	126	32	440	42	8
Line	Circle	74	208	72	754	31	6
	Line	57	217	86	947	34	6

Table 7-1: Experimental results with a swarm of 4 robots

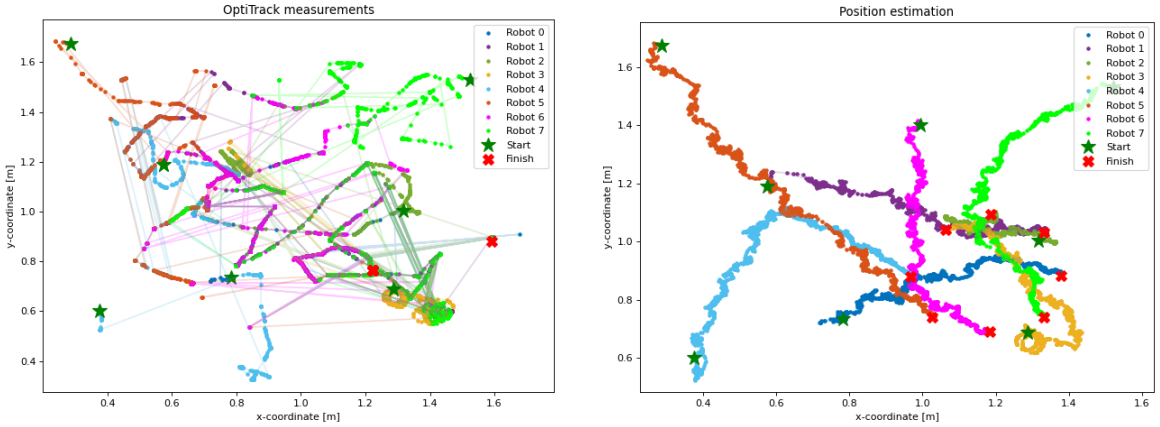
From these experiments, the following was observed:

- The position and orientation estimations provided by the OptiTrack system allow all three algorithms to converge to the correct formation.
- The network topology has a clear effect on the performance of the ETC and PETC algorithm, while it does not significantly effect the phase algorithm.
- The time required for the execution of the phase algorithm, $t_{phase} = 2D\tau$, is still insignificant compared to the entire convergence time.
- The convergence time of the ETC and PETC algorithms do not match the results of the *Webot* simulation. However, this can be attributed to the initial positions of the robots, as their starting configuration is restricted to one configuration within a 1x0.5m region, which differs from the 2x2m area utilised in the simulations. Furthermore, the *Webot* results include situations with collision avoidance, which requires additional time.
- The phase algorithm requires more time to converge compared to the *Webot* simulations and now performs similar to the ETC and PETC algorithms. The additional convergence time can be explained by the noise on the position and orientation of the robot. Due to the inaccuracy of the orientation and position, the robots no longer move in a straight line to their position in the formation, this is visible in the trajectory in Figure 7-12d.

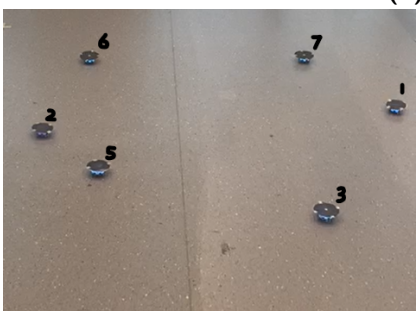
7-5-3 Experiments with a larger swarm size

More experiments were conducted with a larger swarm size to provide additional verification of the results obtained from the *Webot* simulation.

However, trials with 6 and 8 robots showed that the position estimations were not accurate enough to support a swarm with more than 4 robots. This observation is demonstrated in Figure 7-13, where a swarm of 8 robots connected via a hypercube network topology was required to assemble in a circular formation with a radius of 20cm using the ETC algorithm.



(a) OptiTrack measurements **(b)** Position estimation



(c) Actual final positions*

Figure 7-13: Formation control with a swarm of 8 robots using ETC

*Note that Figure 7-13c only includes the position of 6 robots, the reason for this is that robot 0 and 4 moved out of the field.

As Figure 7-13a shows, the OptiTrack system loses track of or switches between robots when they are in close proximity to each other. As a consequence, one of the two scenarios may occur; either the position estimation is based on the robots' odometry for the remainder of their trajectory, or the robots are assigned to the same OptiTrack data. Since the odometry of the robots is not that accurate, the robots fail to converge to the correct formation. The robots believe they have assembled in a circular formation, as indicated in Figure 7-13b, while in reality they are not even close. It follows that, as the swarm size increases, this problem becomes even more apparent.

To prevent losing track of the robots, a comparable experiment was conducted that required the robot to converge towards a circle with a larger radius. However, the robots inevitably crossed each other's paths, resulting in the OptiTrack system losing track.

The experiments conducted with 6 robots, which are presented in Appendix B-4-3, provides similar insights.

7-6 Discussion

Comparing the algorithm in a physical environment shows that the execution time of the phase algorithm is no longer significant and the convergence time is mainly dominated by the proportional controller. The phase algorithm finds in most situations a more efficient trajectory compared to the ETC and PETC algorithms. As a result, the phase algorithm converges faster and requires fewer triggering events. Therefore, it is worth considering the use of the phase algorithm over the ETC and PETC algorithms.

It is important to note, that testing with a larger swarm of real Elisa-3 robots could still provide more insights. To increase the size of the swarm, further adjustments to the position estimation are required. Possible improvements might include adding sensor data from the accelerometers to improve the odometry measurements, improving the tracking by using a single big marker on each robot, and ensuring the OptiTrack measurements are used once the tracker of a robot is recovered.

Finally, the experiments conducted so far assume that the environment is obstacle free. It is worth mentioning that the phase algorithm is less resilient to changes in the environment. This is due to the fact that the phase algorithm computes a formation before moving toward it. If the formation is centred around an area with many static obstacles, it is no longer possible to assemble in the precomputed formation. The agents will need a new round of communications to find a new position for the centre of the formation. This may reduce the efficiency of the phase algorithm.

Chapter 8

Conclusions and Future work

The aim of this thesis was to investigate whether the knowledge of global measurements undermines the benefit of using a (periodic) event-triggered algorithm over using a wave algorithm to solve the consensus problem for single-integrator multi-agent systems (MAS).

The motivation behind this goal was the observation that a considerable amount of research has been carried out over the years to reduce the number of transmissions in MAS using (periodic) event-triggered consensus. These algorithms guarantee convergence to the neighbourhood of a point or a formation, while keeping the number of transmissions to a minimum by only transmitting state measurements when necessary. In the case of single-integrator systems, that point is defined by the average of all initial states, whereas for a formation, an additional bias is included.

In addition to event-triggered consensus, another alternative exists: wave algorithms. Wave algorithms are message-passing schemes that can be used to distribute the state measurements of each agent to all others in the system. This information can be used by each agent to converge to the average consensus point or a formation. Each approach has its own strengths and weaknesses, which naturally raises the question whether one could perform better in certain applications.

To answer to this question, the three consensus algorithms were compared using three different methods. As a first indication, mathematical bounds were derived for three performance metrics; the convergence time, the required number of triggering events, and the required memory usage. This showed that the performance of the event-triggered consensus (ETC) and periodic event-triggered consensus (PETC) algorithms highly depend on the network topology. Whereas, the performance of the wave algorithm is more affected by the size of the communication delays and network diameter.

Given these results, the algorithms were further analysed by using a basic simulation. In these simulations, agents were placed in a two-dimensional plane, connected via a specific network topology, and subjected to a range of static communication delays. The simulations reveal that the ETC algorithm may achieve faster convergence than the wave algorithm under certain conditions. These conditions include a network with high algebraic connectivity, a

relatively large diameter, and a low maximum degree, as well as a communication delay that falls within a specific range.

However, there are only a few network topologies that satisfy these criteria, and even with such a communication network, the ETC algorithm achieves only slightly faster convergence at the expense of significantly more transmissions. It was also observed that the PETC algorithm is the least efficient algorithm among the three. This confirms the intuition that periodically checking the triggering function does not benefit the performance, if agents are capable of checking the triggering function continuously.

Finally, the third method involved implementing the algorithms on real robots to examine the impact of non-ideal conditions that are encountered in real applications. Additionally, to provide a more extensive analysis, the algorithms were also evaluated in the realistic simulation environment *Webots*. For these experiments, the two-wheeled Elisa-3 robots developed by GCTronic were used. The robots were required to assemble into a formation while avoiding collisions with other robots in the swarm.

The results obtained with *Webots* gave three main insights. Firstly, the execution time of the wave algorithm was no longer a significant factor of the total convergence time. Secondly, the wave algorithm performed better with respect to both metrics, indicating that the proportional controller in the wave algorithm finds a more efficient trajectory. Lastly, the ETC algorithm performed worse compared to the PETC algorithms, which can be attributed to the limited control frequency and the time-dependency of the triggering function causing more control updates and transmissions.

The experiments with the real Elisa-3 robots confirm that the wave algorithm is either more or equally efficient compared to the ETC and PETC algorithm. However, testing with a larger swarm may still provide more or different insights.

Through the results of the comparisons, this work has shown that it is worth considering the use of wave algorithms for solving the consensus problem in single-integrator MAS when agents have access to global measurements. The work in this thesis can be used to continue research into consensus problem in MAS. Some suggestions for possible directions include:

- Currently, the simulations consider static time-invariant communication delays. In practical applications, the communication delays might vary from message to message. These varying time delays might have a positive impact on the performance of the ETC and PETC algorithms.
- The study in this thesis was based on the assumption that agents are connected through a time-invariant connected network topology. However, when agents move the network topology might change over time. There are alternative (periodic) event-triggering algorithms that still converge as long as the union of the network graphs includes a spanning tree. The question arises whether the phase algorithm will also be more efficient if it is only guaranteed that the union of network graphs over time is connected.
- In most practical applications agents have more complex dynamics. Once more complex dynamical systems are considered, most (periodic) event-triggered algorithms no longer converge to the average of the initial states. On the other hand, the wave algorithm will still ensure convergence to the average. However, the average might no longer be

the most optimal solution. Therefore, it is interesting to further investigate whether the ETC and PETC algorithms find a more efficient solution when agents have complex dynamics.

- The formation control experiments conducted in this work assume that the environment in which the robots are dispatched is free of static obstacles, but in actual deployments this might not be the case. Since the wave algorithms precompute the formation any static obstacle could prevent one of the robots from reaching its position within the formation. Once this happens, the robots will need a new round of communications to adjust the formation. On the other hand, the ETC and PETC algorithms may be less affected by this problem. Therefore, extending the current research to different environments could reveal situations where the ETC and PETC algorithms are more efficient.

- The *Webot* results can be further validated by performing experiments with more real Elisa-3 robots. This requires optimising the position estimation and improving the tracking of the robots by the OptiTrack system.

- This study could be taken in a different direction by considering agents that only have access to range-based measurements obtained through Ultra-wideband ranging. We know of two approaches that achieve consensus in such scenarios. Either the relative position is obtained through some floating point operations [64]. This is however computationally complex and the complexity scales with the number of neighbouring agents. The other option would be to use a specific movement strategy [14, 55]. These strategies require continuous monitoring of the relative distance. The question that has not been answered is whether it is possible to use a movement strategy in combination with self-triggering consensus to limit the number of transmissions and whether this is more efficient than directly computing the relative position and applying an ETC, PETC, or wave algorithm.

Appendix A

Proof of Theorem 6.1

This proof is an extension of the proof of Theorem 3.6 by Wang [112], and includes communications delays $\tau \in [(n-1)h, nh)$ with $n \in \mathbb{Z}^+$

Proof. Let $\delta_i(t) = x_i(t) - 1/N \sum_{j \in \mathcal{N}} x_j = x_i(t) - \bar{x}_0$ and then $\delta(t) = [\delta_1(t), \delta_2(t), ..., \delta_N(t)] = x(t) - \bar{x}_0 \mathbf{1}_N$. Consider the Lyapunov function:

$$V(t) = \frac{1}{2}\delta^T(t)\delta(t) \geq 0 \tag{A-1}$$

The derivative of $V(t)$ is

$$\dot{V}(t) = \delta^T(t)\dot{\delta}(t) = (x(t) - \bar{x}_0 \mathbf{1}_N)^T \dot{x}(t) = -x^T(t)L\hat{x}(t-\tau) \tag{A-2}$$

We have, with $q \in \mathbb{Z}^+$

$$\begin{aligned} V((q+1)h+\tau) =& V(0) + \int_0^h \dot{V}(s)ds + \int_h^{2h} \dot{V}(s)ds + ...+ \\ & \int_{(n-1)h}^{\tau} \dot{V}(s)ds + \int_{\tau}^{nh} \dot{V}(s)ds + \int_{nh}^{h+\tau} \dot{V}(s)ds + ...+ \\ & \int_{qh+\tau}^{(q+n)h} \dot{V}(s)ds + \int_{(q+n)h}^{(q+1)h+\tau} \dot{V}(s)ds \end{aligned} \tag{A-3}$$

Note that for any value of q, if $t \in [qh+\tau, (q+1)h+\tau)$, we have $\hat{x}(t-\tau) = \hat{x}(qh)$. Therefore,

$$\begin{aligned} \int_{qh+\tau}^{(q+n)h} \dot{V}(s)ds &= \int_{qh+\tau}^{(q+n)h} x^T(s)L\hat{x}(qh)ds \\ \int_{(q+n)h}^{(q+1)h+\tau} \dot{V}(s)ds &= \int_{(q+n)h}^{(q+1)h+\tau} x^T(s)L\hat{x}(qh)ds \end{aligned} \tag{A-4}$$

Furthermore, we obtain that for $t \in [qh+\tau, (q+1)h+\tau)$,

$$x(t) = x(qh+\tau) + (t - (qh-\tau))(-L\hat{x}(qh)) \tag{A-5}$$

with

$$x(qh+\tau) = x(qh) - (\tau-(n-1)h)L\hat{x}((q-n)h) - hL\hat{x}((q-(n-1))h) + ... + hL\hat{x}((q-1)h) \tag{A-6}$$

Then

$$\int_{qh+\tau}^{(q+n)h} x^T(s)L\hat{x}(qh)ds = \int_{qh+\tau}^{(q+n)h} (x(qh+\tau) - (s-(qh-\tau))L\hat{x}(qh))^T L\hat{x}(qh)ds = \tag{A-7}$$
$$(nh-\tau)x^T(qh+\tau)L\hat{x}(qh) - \left(\frac{n^2h^2}{2} - nh\tau + \frac{\tau^2}{2}\right)\hat{x}^T(qh)L^TL\hat{x}(qh)$$

Given $x(qh+\tau)$ and $e(qh) = x(qh) - \hat{x}(qh)$, we have

$$\begin{aligned}
&(nh-\tau)x^T(qh+\tau)L\hat{x}(qh) - \left(\frac{n^2h^2}{2} - nh\tau + \frac{\tau^2}{2}\right)\hat{x}^T(qh)L^TL\hat{x}(qh) = \\
&(nh-\tau)\hat{x}^T(qh)L\hat{x}(qh) + (nh-\tau)e^T(qh)L\hat{x}(qh) \\
&-(nh-\tau)(\tau-(n-1)h)\hat{x}^T((q-n)h)L^TL\hat{x}(qh) \\
&-(nh-\tau)h\hat{x}^T((q-(n-1))h)L^TL\hat{x}(qh) + ... + \\
&-(nh-\tau)h\hat{x}^T((q-1)h)L^TL\hat{x}(qh) - \left(n^2h^2/2 - nh\tau + \tau^2/2\right)\hat{x}^T(qh)L^TL\hat{x}(qh)
\end{aligned} \tag{A-8}$$

Similarly, we have

$$\begin{aligned}
&\int_{(q+n)h}^{(q+1)h+\tau} x^T(s)L\hat{x}(qh)ds = (\tau+(1-n)h)\hat{x}^T(qh)L\hat{x}(qh) \\
&+(\tau+(1-n)h)e^T(qh)L\hat{x}(qh) - (\tau+(1-n)h)(\tau-(n-1)h)\hat{x}^T((q-n)h)L^TL\hat{x}(qh) \\
&-(\tau+(1-n)h)h\hat{x}^T((q-(n-1))h)L^TL\hat{x}(qh) + ... + \\
&-(\tau+(1-n)h)h\hat{x}^T((q-1)h)L^TL\hat{x}(qh) - \frac{1}{2}\left((1-n^2)h^2 + 2nh\tau - \tau^2\right)\hat{x}^T(qh)L^TL\hat{x}(qh)
\end{aligned} \tag{A-9}$$

Since the Laplacian L is symmetric and positive semi-definite, we have the following inequalities [112]:

$$\hat{x}^T(qh)L^TL\hat{x}(qh) \leq \lambda_N\hat{x}^T(qh)L\hat{x}(qh) \tag{A-10}$$

and for $m \in \mathbb{Z}^+$

$$\begin{aligned}
\hat{x}^T((q-m)h)L^TL\hat{x}(qh) &\leq \frac{1}{2}\left(\hat{x}^T((q-m)h)L^TL\hat{x}((q-m)h) + \hat{x}^T(qh)L^TL\hat{x}(qh)\right) \\
&\leq \frac{\lambda_N}{2}\left(\hat{x}^T((q-m)h)L\hat{x}((q-m)h) + \hat{x}^T(qh)L\hat{x}(qh)\right)
\end{aligned} \tag{A-11}$$

Using the event-triggering condition and inequality A-11, we have

$$e^T(qh)L\hat{x}(qh) \geq -|e^T(qh)L\hat{x}(qh)| \geq -\frac{\sigma\lambda_N}{2}\left(\hat{x}^T(qh)L\hat{x}(qh) + \hat{x}^T((q-1)h)L\hat{x}((q-1)h)\right) \tag{A-12}$$

Therefore, combining equations A-10, A-11 and A-12, we have

$$\int_{qh+\tau}^{(q+n)h} x^T(s)L\hat{x}(qh)ds \geq \tag{A-13}$$
$$(nh-\tau)\hat{x}^T(qh)L\hat{x}(qh) - (nh-\tau)\frac{\sigma\lambda_N}{2}\left(\hat{x}^T(qh)L\hat{x}(qh) + \hat{x}^T((q-1)h)L\hat{x}((q-1)h)\right)$$
$$-(nh-\tau)(\tau-(n-1)h)\frac{\lambda_N}{2}\left(\hat{x}^T((q-n)h)L\hat{x}((q-n)h) + \hat{x}^T(qh)L\hat{x}(qh)\right)$$
$$-(nh-\tau)h\frac{\lambda_N}{2}\left(\hat{x}^T((q-(n-1))h)L\hat{x}((q-(n-1))h) + \hat{x}^T(qh)L\hat{x}(qh)\right) + ...+$$
$$-(nh-\tau)h\frac{\lambda_N}{2}\left(\hat{x}^T((q-1)h)L\hat{x}((q-1)h) + \hat{x}^T(qh)L\hat{x}(qh)\right)$$
$$-\left(n^2h^2/2 - nh\tau + \tau^2/2\right)\lambda_N\hat{x}^T(qh)L\hat{x}(qh)$$

and

$$\int_{(q+n)h}^{(q+1)h+\tau} x^T(s)L\hat{x}(qh)ds \geq \tag{A-14}$$
$$(\tau+(1-n)h)\hat{x}^T(qh)L\hat{x}(qh) - (\tau+(1-n)h)\frac{\sigma\lambda_N}{2}\left(\hat{x}^T(qh)L\hat{x}(qh) + \hat{x}^T((q-1)h)L\hat{x}((q-1)h)\right)$$
$$-(\tau+(1-n)h)(\tau-(n-1)h)\frac{\lambda_N}{2}\left(\hat{x}^T((q-n)h)L\hat{x}((q-n)h) + \hat{x}^T(qh)L\hat{x}(qh)\right)$$
$$-(\tau+(1-n)h)h\frac{\lambda_N}{2}\left(\hat{x}^T((q-(n-1))h)L\hat{x}((q-(n-1))h) + \hat{x}^T(qh)L\hat{x}(qh)\right) + ...+$$
$$-(\tau+(1-n)h)h\frac{\lambda_N}{2}\left(\hat{x}^T((q-1)h)L\hat{x}((q-1)h) + \hat{x}^T(qh)L\hat{x}(qh)\right)$$
$$-\frac{1}{2}\left((1-n^2)h^2 + 2nh\tau - \tau^2\right)\lambda_N\hat{x}^T(qh)L\hat{x}(qh)$$

Combining equation A-3, A-13 and A-14, we have

$$V((q+1)h+\tau) \leq V(0) + \int_0^h \dot{V}(s)ds + \int_h^{2h}\dot{V}(s)ds + ... + \int_{(n-1)h}^{\tau}\dot{V}(s)ds + \tag{A-15}$$
$$-\left((n-1)h^2 - h\tau\right)\frac{\lambda_N}{2}\hat{x}^T(0)L\hat{x}(0)$$
$$-\sum_{i=1}^{n-2}\left((n-1)h^2 - h\tau - ih^2\right)\frac{\lambda_N}{2}\hat{x}^T(ih)L\hat{x}(ih)$$
$$+(\tau+\sigma)\,h\frac{\lambda_N}{2}\hat{x}^T((n-1)h)L\hat{x}((n-1)h)$$
$$-\sum_{j=n}^{q-n}\left(-h^2 - 2h\tau - 2h\sigma + \frac{2h}{\lambda_N}\right)\frac{\lambda_N}{2}\hat{x}^T((jh)L\hat{x}(jh)$$
$$-\sum_{k=0}^{n-2}\left(-(k+2)h^2 - h\tau - 2h\sigma + \frac{2h}{\lambda_N}\right)\frac{\lambda_N}{2}\hat{x}^T((q-k-1)h)L\hat{x}((q-k-1)h)$$
$$-\left(-h^2 - h\tau - h\sigma + \frac{2h}{\lambda_N}\right)\frac{\lambda_N}{2}\hat{x}^T(qh)L\hat{x}(qh)$$

By inequality 6-3 in Theorem 6.1, we know that in inequality A-15

$$\begin{cases} \left(-h^2 - 2h\tau - 2h\sigma + \frac{2h}{\lambda_N}\right) \frac{\lambda_N}{2} \hat{x}^T((jh)L\hat{x}(jh) \geq 0, & n \leq j \leq q-n \\ \left(-(k+2)h^2 - h\tau - 2h\sigma + \frac{2h}{\lambda_N}\right) \frac{\lambda_N}{2} \hat{x}^T((q-k-1)h)L\hat{x}((q-k-1)h) \geq 0, & n-2 \leq k \leq 0 \\ \left(-h^2 - h\tau - h\sigma + \frac{2h}{\lambda_N}\right) \frac{\lambda_N}{2} \hat{x}^T(qh)L\hat{x}(qh) \geq 0 \end{cases} \tag{A-16}$$

The remainder of the proof follows the same steps as the proof of Wang [112]. Given the fact that $V(t) \geq 0$, we have

$$\lim_{q \to +\infty} \hat{x}^T(qh)L\hat{x}(qh) = 0 \tag{A-17}$$

Using the properties of the Laplacian, we have

$$\lim_{q \to +\infty} L\hat{x}(qh) = 0 \tag{A-18}$$

Given equation (A-18) and the triggering function, we know that

$$\lim_{q \to +\infty} e(qh) = 0 \tag{A-19}$$

and then

$$\lim_{q \to +\infty} Lx(qh) = 0 \tag{A-20}$$

Furthermore, when $t \in [qh, (q-(n-1))h + \tau)$, we have

$$x(t) = x(qh) - (t - qh)L\hat{x}((q-n)h) \tag{A-21}$$

and when $t \in [(q-(n-1))h + \tau, (q+1)h))$,

$$x(t) = x(qh) - (\tau - (n-1)h)L\hat{x}((q-n)h) - (t - (q-(n-1)h + \tau))L\hat{x}((q-(n-1))h) \tag{A-22}$$

Therefore, we have

$$\lim_{t \to +\infty} Lx(t) = 0 \tag{A-23}$$

and thus

$$\lim_{t \to +\infty} L\delta(t) = \lim_{t \to +\infty} (Lx(t) - L\bar{x}_0 \mathbf{1}_N) = 0 \tag{A-24}$$

Since we have $\delta(t) \perp \mathbf{1}_N$ and $\delta^T(t)\delta(t) \leq \frac{1}{\lambda_2}\delta^T(t)L\delta(t)$ by Theorem 8 [89], it holds

$$\lim_{t \to +\infty} \delta(t) = 0 \tag{A-25}$$

which indicates that

$$\lim_{t \to +\infty} x(t) = \lim_{t \to +\infty} (\delta(t) + \bar{x}_0 \mathbf{1}_N) = \bar{x}_0 \mathbf{1}_N \tag{A-26}$$

This shows that the states of the agents converge to the average of the initial positions, $\bar{x}_0 \mathbf{1}_N$. □

Appendix B

Additional results

B-1 Triggering coefficients

B-1-1 Line graph

The line graph will have a maximum of 12 agents and is subjected to a time delay of $\tau = 0.2$.

ETC

Given this delay, the number of agents, equation (6-2) and Theorem 5.1, the maximum value of c_0 that still guarantees convergence is can be determined:

$$c_0 \leq \frac{1 * 0.06 * 0.1}{4 * \sqrt{12}} = 0.0004$$

Choosing the value $c_0 = 0.0003$ should suffice in all simulations. The values for c_1 and α can not be computed and are selected based on the results in Figure B-1. This shows that selecting $c1 = 0.25$ and $\alpha = 0.5\omega$ should give the best trade-off between the convergence time and the number of triggering events ratio.

PETC

Given equation (6-6), the value for σ that guarantees convergence can maximally be:

$$\sigma \leq \frac{1 * 0.1 * 0.06}{4 * |3.2|} = 0.00045$$

Simulations however show that a larger value for σ also guarantees convergence and limits the number of triggering events. Therefore, the value of σ has been selected based on the simulation results in Figure B-4. Furthermore, to ensure the system is still stable for large delays, σ must be smaller than 0.05.

Figure B-2, shows that a small value does significantly increase the number of triggering events. Therefore, $\sigma = 0.04$ was used for all simulations.

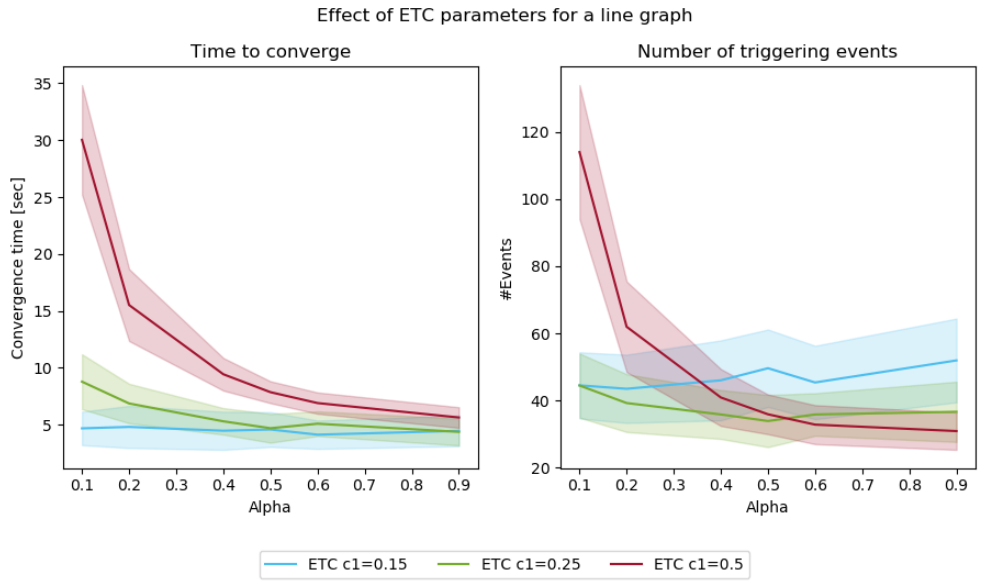


Figure B-1: ETC parameter selection for a line topology, with 6 agents, $c_0 = 0.0003$, $\tau = 0.2$, $\tau_{max} = 0.39$, $\epsilon = 0.1$, and initiated with random positions

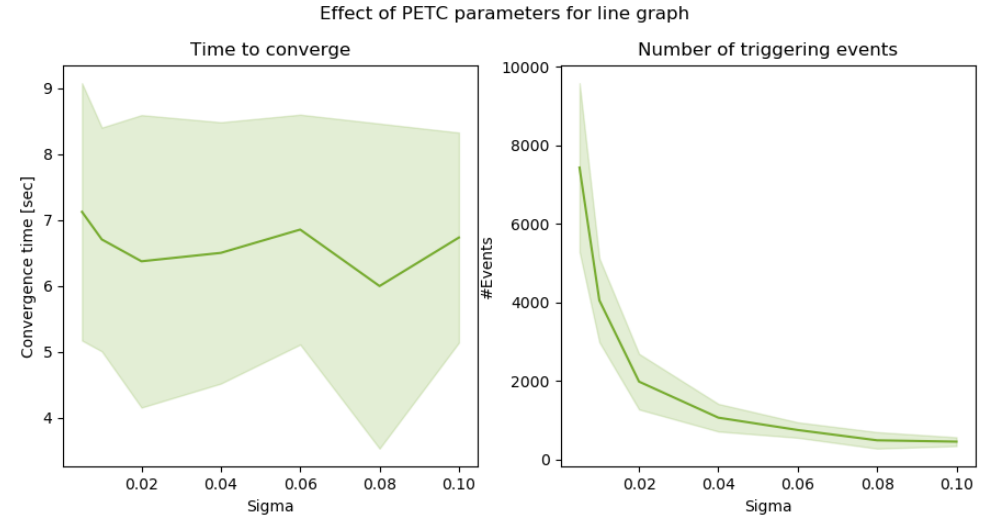


Figure B-2: PETC parameter selection for a line topology, with 6 agents, $\tau = 0.1$, $h = 0.002$, $\epsilon = 0.1$, and initiated with random positions

B-1-2 Hypercube graph

The simulations with a hypercube graph will consider at most 64 agents and are subjected to a maximum time delay of 0.18 sec.

ETC

Given this delay, the number of agents, equation (6-2) and Theorem 5.1, the maximum value of c_0 that still guarantees convergence is:

$$c_0 \leq \frac{1 * 2 * 0.1}{12 * \sqrt{64}} = 0.002$$

Keeping the value $c_0 = 0.0003$ from the line graph should therefore also suffice for the hypercube graph. The values for c_1 and α can not be computed and are selected based on the results in Figure B-3. This shows that selecting $c1 = 0.25$ and $\alpha = 0.5\omega$ should give the best trade-off between the convergence time and the number of triggering events.

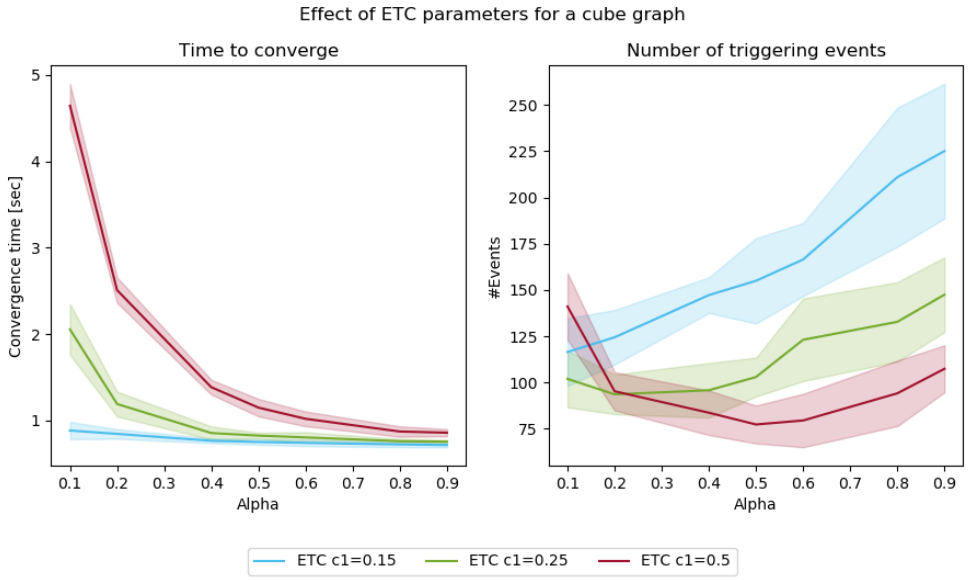


Figure B-3: ETC parameter selection for a hypercube topology, with 16 agents, $c_0 = 0.0003$, $\tau = 0.14$, $\tau_{max}^{ETC} = 0.19$, $\epsilon = 0.1$, and initiated with random positions

PETC

Similarly, as for the line graph, the value of σ has been selected based on the simulation results in Figure B-4.

The convergence time shows that a value lower than 0.04 does not improve the convergence time. Therefore, as a larger value for σ results in fewer triggering events, $\sigma = 0.04$, was used for all simulations.

B-2 Additional results for the simulation-based comparison

B-2-1 Influence of the initial area

The impact of the size of the initial area and the maximum allowable error is verified by executing the algorithms for 16 agents connected via a hypercube topology and subjected to a communication delay of $\tau = 0.1$ sec.

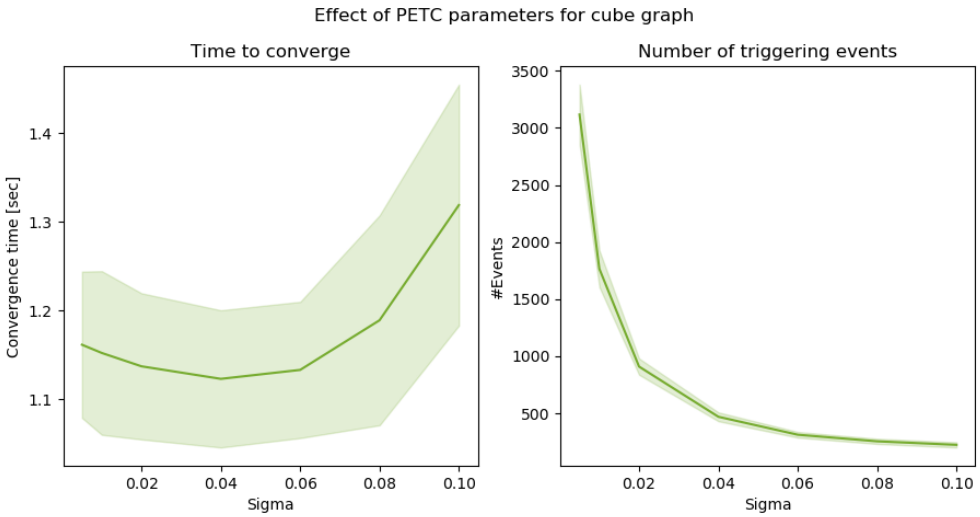


Figure B-4: PETC parameter selection for a hypercube topology, with 16 agents, $\tau = 0.1$, $h = 0.002$, $\epsilon = 0.1$, and initiated with random positions

The results in Figure B-5 and B-6 confirm the following intuitions:

- By increasing the initial area the event-triggered consensus (ETC) and periodic event-triggered consensus (PETC) algorithms require more time to converge as the trajectory is longer.

- The phase algorithm is not impacted by the size of the initial area, a larger area will lead to a larger maximum control input and therefore the convergence time will remain similar.

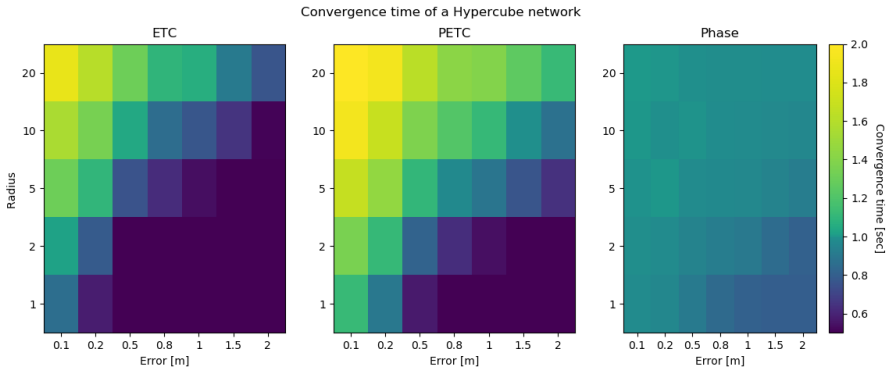


Figure B-5: Convergence time of agents connect via a hypercube graph, with $N = 16$, $\tau = 0.14$, and initiated with random positions

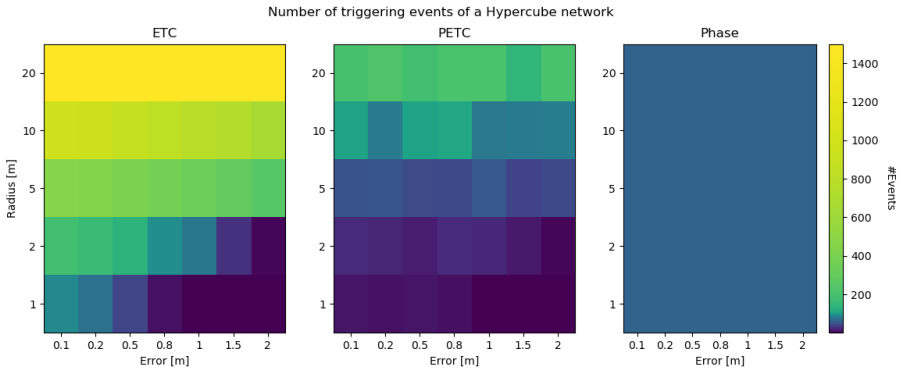


Figure B-6: Number of triggering events of agents connect via a hypercube graph, with $N = 16$, $\tau = 0.1$, and initiated with random positions

B-3 Additional results for the application-based simulation

B-3-1 Random initial positions

The robots are connected via a line network topology and are required to converge to a circular formation, with a radius of 20 cm. The simulation results are depicted in Figure B-7, from these results the following can be observed:

- The time required for the execution of the phase algorithm, is insignificant compared to the entire convergence time.
- The ETC and PETC algorithms require significantly more time to converge to the formation.
- The ETC and PETC algorithms require a comparable number of triggering events. The PETC algorithm requires a similar amount of triggering events as in Chapter 6, while the ETC triggers more often. This is mainly caused by the time-dependent triggering function of the ETC algorithm.

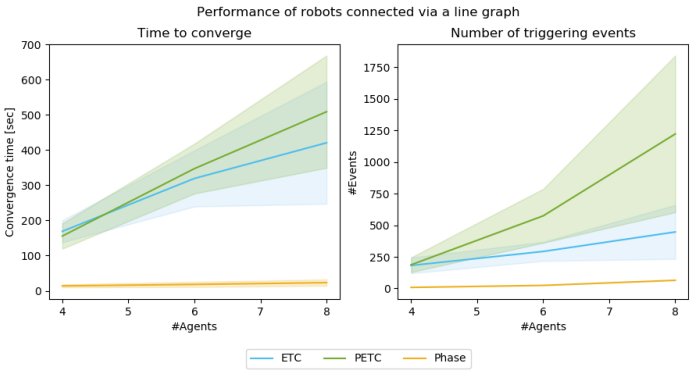
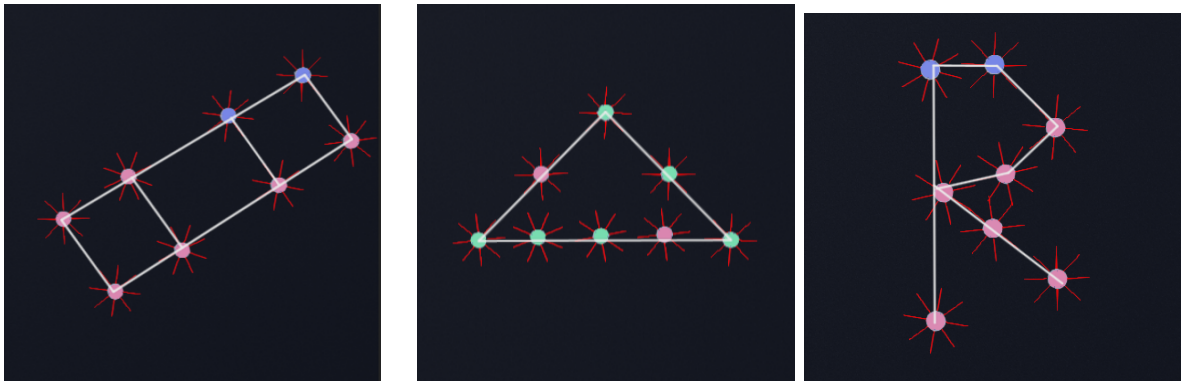


Figure B-7: Performance of agents connected via a line graph initialised at random positions, with $\tau = 0.1$, and $\epsilon = 0.1$

B-3-2 Influence of the formation shape

These simulations were carried out with 8 robots, subjected to a time delay of 0.1 sec. The robots are required to assemble in one of the three formations given in Figure B-8.



(a) Grid **(b)** Triangle **(c)** Letter R

Figure B-8: Formation structures with 8 robots

From the simulations results, in Figure B-9, we can observe the following:

- The algorithms' performances are only slightly influenced by the shape of the formation. If the shape is denser, the convergence time will be longer as the robots will be closer to each other, which will increase the need for collision avoidance.
- In all four formations, the phase algorithm still finds the most efficient trajectory and requires the least amount of triggering events.

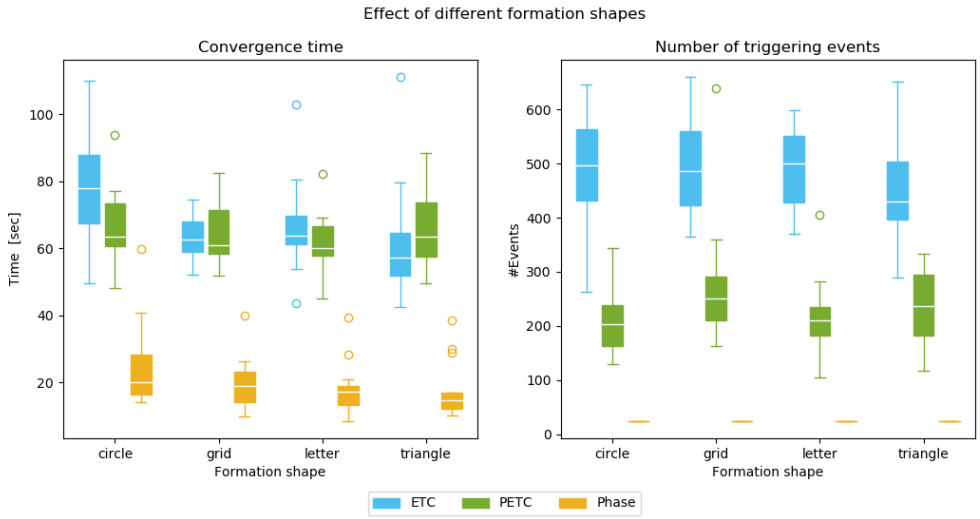


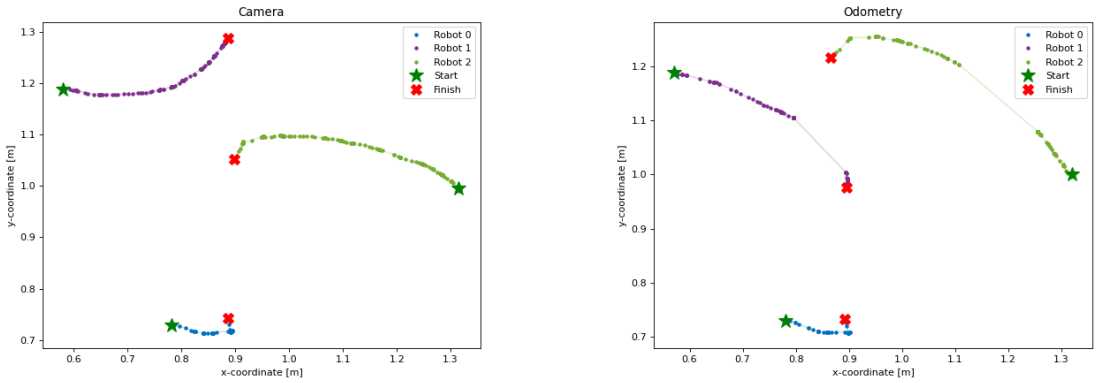
Figure B-9: Performance of 8 agents connected via a Hypercube graph required to assemble in one of three formations (circle, grid, letter, or triangle), with $\tau = 0.1$ and $\epsilon = 0.1$.

B-4 Additional results for the real-world comparison

B-4-1 Formation control without OptiTrack system

In this experiment, 3 robots were placed in a 2x1m area. The robots are connected via a line network topology and are all executing the phase algorithm to converge to a vertical line based only on the odometry data.

Figure B-10 shows that according to the odometry data the robots have reached the required line formation in the correct order from bottom to top; robot 0, 1, 2. However, the actual positions of the robots, as registered by OptiTrack, differ from the expected positions. Therefore, in reality the robots have not assembled in the correct formation.



(a) OptiTrack data **(b)** Odometry data

Figure B-10: Formation control without OptiTrack system

B-4-2 Communication delay with 4 robots

Given the 250Hz update frequency of the communication system and the structure of the communication protocol, we would expect that a swarm of 4 robots experiences a communication delay between 4-28msec.

The communication delay the system experiences can vary due to the fact that the robots can not directly communicate with each other. The message will first have to be received by the PC and will be forwarded to all neighbour robots once received. Essentially, the message is forwarded twice through the same channel instead of once.

Furthermore, the worst case expected delay is 28msec, which occurs when all robots trigger at the same time. Since the PC can only send updated state information of one robot at a time, it will require 16msec to send the updated state information of all robots. On top of this, it might happen that the robots trigger after it has just received a message from the PC. In this case we will have an additional delay of 4 msec. Now the remain 8msec is due to the fact that the robot sends two types of acknowledgement message, where one of these contains the state update and trigger flag.

To measure the actual delay of the communication channel, we have performed the following experiment:

1. The PC sends updated data to the robots and starts a timer.
2. The robots receive the data and immediately send it back in an one of the two acknowledgement packets.
3. Once the PC receives the updated data it stops the timer and measures the difference.

The results of the experiments are given in Figure B-11 and demonstrate the following:

- The average delay is higher than the expected worst case delay of 28msec. The additional delay can be caused by the fact that the robots run task sequentially and the limited loop frequency. Furthermore, the system experiences packet dropouts which also raises the average communication delay.

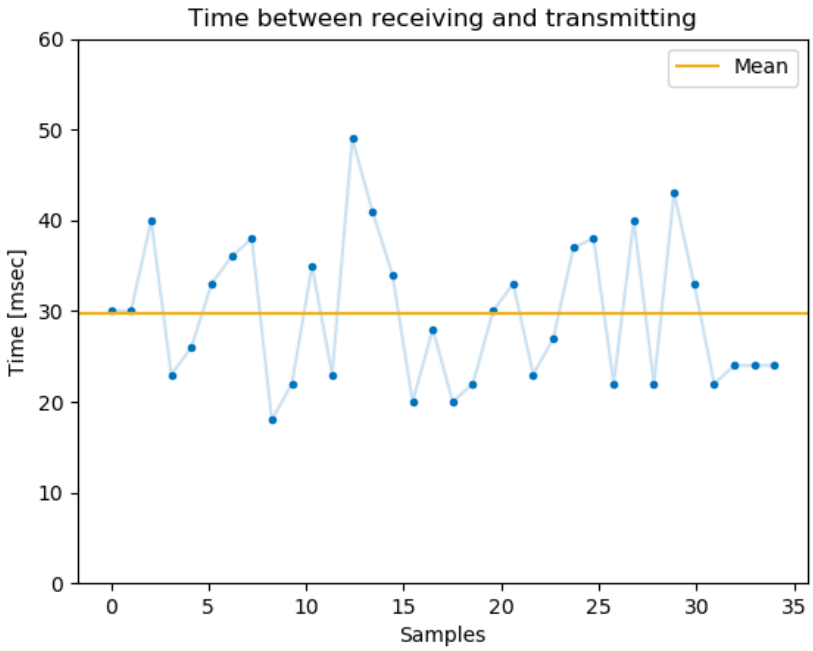


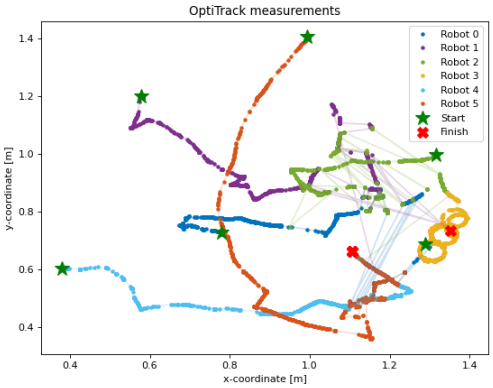
Figure B-11: Communication delay with 4 robots

B-4-3 Formation control with 6 robots

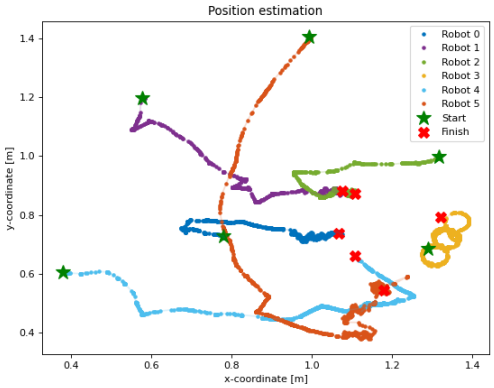
In this experiment a swarm of 6 robots connected via a line network topology was required to assemble in a circular formation with a radius of 20cm using the ETC algorithm.

From the results, in Figure B-12, we draw the same conclusions as in Chapter 7. Furthermore, we observe an additional problem that occurred often during trial runs. It shows that Robot

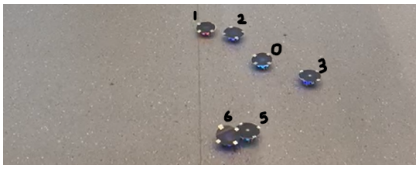
3 moves in a circular pattern around its initial position, this behaviour can be attributed to a wheel failure. Despite Robot 3's attempt to move in a straight line, one of its wheels experiences significant slip, causing it to move in circles. This problem can be solved by either re-calibrating the robot or replacing it with another robot.



(a) OptiTrack measurements



(b) Position estimation



(c) Actual final positions

Figure B-12: Formation control with a swarm of 6 robots using ETC

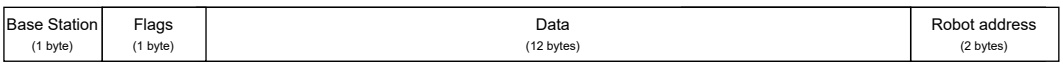
Appendix C

Communication Protocol

The protocol developed by GCTronic operates by sending packets to the base-station, which contains information for four robots. A similar approach is applied in reception, the base-station receives acknowledgement payloads from four robots (a total of 64 bytes) and forwards it to the computer. As part of the optimised protocol, the size of the packet transmitted to each robot is restricted to 16 bytes, of which two bytes are being used for the robot's address and another byte is used for updating the base-station.

Based on the algorithm that is being used these packets contain different information. Furthermore, during start-up a different type of packet structure is used than during operation. These initialisation messages contain information regarding the number of robots in the swarm, the network topology, the desired formation, their initial location, and orientation.

Each packet has the general structure as depicted in Figure C-1. The first byte in the packet serves to transmit flags, with the most significant bit (MSB) (init flag) indicating the type of message, either an initialisation packet or an operational packet. The 5th bit is reserved and used by the RF-module.



(a) General packet structure

(b) Structure of the Flag byte

Figure C-1: General packet structure of packets transmitted to the robots

Once a robot receives a packet from the PC, it responds by sending one of two types of acknowledgement messages. The information contained in these acknowledgement messages is the same for all three algorithms, the general structure is depicted in Figure C-2.

Type (1 byte)	#Events (2 bytes)	Orientation (2 bytes)	$\hat{x}_i$ - position (4 bytes)	$x_{odometry}$ (4 bytes)	Free (3 bytes)

(a) Type 0

Type (1 byte)	Speed (2 bytes)	Turning (2 bytes)	Accelerometer (4 bytes)	Timestaps (7 bytes)

(b) Type 1

Figure C-2: General packet structure of acknowledgement packets transmitted by robot i to the PC

C-1 (Periodic) Event-triggered algorithm

A schematic representation of an initialisation packet sent to activate a robot operating under the event-triggered consensus (ETC) or periodic event-triggered consensus (PETC) algorithm is presented in Figure C-5. The flag byte's 6th and 8th bits indicate the type of information being sent, while the four least significant bits (LSBs) are used to indicate the value of j. As each packet contains 5 elements of the array the value of j equals $Start_{ID} * 5$.

The first type, contains general information about the systems such as the number of robots in the network, its initial orientation and position, the maximum allowed speed u_{max} and the maximum allowed error radius ϵ.

The second, third, and fourth type of packet contains information regarding the Laplacian and the inter-agent distance.

Flags (1 byte)	ID (1 byte)	#Robots (1 byte)	Orientation (2 bytes)	x_i - position (4 bytes)	ϵ (2 bytes)	u_{max} (2 bytes)	Robot address (2 bytes)

(a) Initialisation packet structure - type 1

Flags (1 byte)	L_{ij}, Bx_{ij}, By_{ij} (2 bytes)	$L_{i(j+1)}$, $Bx_{i(j+1)}$, $By_{i(j+1)}$ (2 bytes)	...	$L_{i(j+5)}$, $Bx_{i(j+5)}$, $By_{i(j+5)}$ (2 bytes)	Robot address (2 bytes)

(b) Initialisation packet structure - type 2, 3, 4

Init (bit 7)	Type (bit 6)	RF (bit 5)	Type (bit 4)	Start ID (bit 3-0)

(c) Structure of the Flag byte

Figure C-3: Packet structure of initialisation packets transmitted to robot i for the ETC and PETC algorithms

Once initialised, the robots require continuous updates of their position and orientation from the OptiTrack system. Additionally, when a triggering event happens for one of the robot's neighbours, the PC must send an updated version of the neighbour's position x_j. The data bit in the flag byte is used to indicate the neighbour's position is being updated and the Robot ID byte indicates which neighbour.

Finally, the 6th bit in the Flag byte is used to start the algorithms based on input from the user.

Flags (1 byte)	Robot ID (1 byte)	Orientation (2 bytes)	Reset (1 byte)	x_i - position (4 bytes)	x_j - position (4 bytes)	Robot address (2 bytes)

(a) Operational packet structure

Init (bit 7)	Start (bit 6)	RF (bit 5)	Data (bit 4)	Free (bit 3-0)

(b) Structure of the Flag byte

Figure C-4: Packet structure of operational packets transmitted to robot i for the ETC and PETC algorithms

C-2 Wave algorithm

The initialisation packet structure used for robots operating under the wave algorithm is similar to the one used for the ETC and PETC algorithms. The main difference is that the wave algorithms do not require knowledge of the Laplacian, therefore there are only three types of packets.

Flags (1 byte)	ID (1 byte)	#Robots (1 byte)	Orientation (2 bytes)	x_i - position (4 bytes)	ϵ (2 bytes)	u_{max} (2 bytes)	Robot address (2 bytes)

(a) Initialisation packet structure - type 1

Flags (1 byte)	Bx_j, By_j (2 bytes)	$Bx_{(j+1)}$, $By_{(j+1)}$ (2 bytes)	$Bx_{(j+2)}$, $By_{(j+2)}$ (2 bytes)	...	$Bx_{(j+5)}$, $By_{(j+5)}$ (2 bytes)	Robot address (2 bytes)

(b) Initialisation packet structure - type 2, 3

Init (bit 7)	Type (bit 6)	RF (bit 5)	Type (bit 4)	Start ID (bit 3-0)

(c) Structure of the Flag byte

Figure C-5: Packet structure of initialisation packets transmitted to robot i for the ETC and PETC algorithms

Once the parameters of the robots have been transmitted the phase algorithm can be initiated by the user. Throughout the execution of the phase algorithm, the robots exchange the initial positions of all robots. As the optimised protocol permits the transmission of 12 data bytes, a maximum of three positions can be transmitted simultaneously once a triggering event takes place on the robot. Modifying the protocol to allow the transmission of more positions per packet can enhance efficiency. However, the optimised protocol is beneficial for the ETC and PETC algorithms and for transmitting information after the termination of the phase algorithm. Consequently, these changes have not been implemented.

Upon completion of the phase algorithm, there is no further communication between the robots. Nonetheless, the robots require continuous updates on their position and orientation.

The packet structure of these two types of messages is given in Figure C-6.

Flags (1 byte)	x_j - position (4 bytes)	x_{j+1} - position (4 bytes)	x_{j+2} - position (4 bytes)	Robot address (2 bytes)

(a) Packet structure of packets transmitted for the wave algorithm during execution

Flags (1 byte)	Robot ID (1 byte)	Orientation (2 bytes)	Reset (1 byte)	x_i - position (4 bytes)	Free (2 bytes)	Robot address (2 bytes)

(b) Packet structure of packets transmitted to robot i for the wave algorithm after termination

Init (bit 7)	Start (bit 6)	RF (bit 5)	Type (bit 4)	Start ID (bit 3-0)

(c) Structure of the Flag byte

Figure C-6: Packet structure of operational packets transmitted to robot i for the wave algorithm

Bibliography

[1] To sample or not to sample: Self-triggered control for nonlinear systems. *IEEE Transactions on Automatic Control*, 55(9):2030–2042, 2010.

[2] OptiTrack - Motion Capture Systems. https://optitrack.com/, 2021.

[3] Rawad Abdulghafor, S. Abdullah, Sherzod Turaev, and Mohamed Othman. An overview of the consensus problem in the control of multi-agent systems. *Automatika*, 59:143–157, 04 2018.

[4] Zahoor Ahmed, Muhammad Mansoor Khan, Muhammad Abid Saeed, and Weidong Zhang. Consensus control of multi-agent systems with input and communication delay: A frequency domain perspective. *ISA Transactions*, 101:69–77, 2020.

[5] Amos Albert and Robert Gmbh. Comparison of event-triggered and time-triggered concepts with regard to distributed control systems. *Embedded World*, 171902:235–252, 01 2004.

[6] Karolos Antoniadis, Antoine Desjardins, Vincent Gramoli, Rachid Guerraoui, and Igor Zablotchi. Leaderless consensus. In *2021 IEEE 41st International Conference on Distributed Computing Systems (ICDCS)*, pages 392–402, 2021.

[7] K.J. Astrom and B.M. Bernhardsson. Comparison of riemann and lebesgue sampling for first order stochastic systems. In *Proceedings of the 41st IEEE Conference on Decision and Control, 2002.*, volume 2, pages 2011–2016 vol.2, 2002.

[8] H. T. Banks and J. A. Burns. Hereditary control problems: Numerical methods based on averaging approximations. *SIAM Journal on Control and Optimization*, 16(2):169–40, 03 1978. Copyright - Copyright] © 1978 Society for Industrial and Applied Mathematics; Last updated - 2022-10-20.

[9] Pierre-Alexandre Bliman and Giancarlo Ferrari-Trecate. Average consensus problems in networks of agents with delayed communications. *Automatica*, 44(8):1985–1995, 2008.

[10] D. P. Borgers and W. P. M. H. Heemels. Event-separation properties of event-triggered control systems. *IEEE Transactions on Automatic Control*, 59(10):2644–2656, 2014.

[11] V. Borkar and P. Varaiya. Asymptotic agreement in distributed estimation. *IEEE Transactions on Automatic Control*, 27(3):650–655, 1982.

[12] Francesco Bullo, Jorge Cortés, and Sonia Martínez. 2009.

[13] Mengtao Cao, Feng Xiao, and Long Wang. Event-based second-order consensus control for multi-agent systems via synchronous periodic event detection. *IEEE Transactions on Automatic Control*, 60(9):2452–2457, 2015.

[14] Ming Cao, Changbin Yu, and Brian D.O. Anderson. Formation control using range-only measurements. *Automatica*, 47(4):776–781, 2011.

[15] Yongcan Cao, Wenwu Yu, Wei Ren, and Guanrong Chen. An overview of recent progress in the study of distributed multi-agent coordination. *IEEE Transactions on Industrial Informatics*, 9, 07 2012.

[16] Xia Chen, Fei Hao, and Mingyuan Shao. Event-triggered consensus of multi-agent systems under jointly connected topology. *IMA Journal of Mathematical Control and Information*, 32(3):537–556, 2015.

[17] Bin Cheng and Zhongkui Li. Fully distributed event-triggered protocols for linear multiagent networks. *IEEE Transactions on Automatic Control*, 64(4):1655–1662, 2019.

[18] Yi Cheng and Valery Ugrinovskii. Event-triggered leader-following tracking control for multivariable multi-agent systems. *Automatica*, 70:204–210, 2016.

[19] Fan Chung, Vance Faber, and Thomas Manteuffel. An upper bound on the diameter of a graph from eigenvalues associated with its laplacian. *SIAM J. Discrete Math.*, 7:443–457, 05 1994.

[20] J. Cortes, S. Martinez, and F. Bullo. Robust rendezvous for mobile autonomous agents via proximity graphs in arbitrary dimensions. *IEEE Transactions on Automatic Control*, 51(8):1289–1298, 2006.

[21] Jorge Cortés. Finite-time convergent gradient flows with applications to network consensus. *Automatica*, 42(11):1993–2000, 2006.

[22] Cyberbotics Ltd. Webots simulation environment for elisa-3 robot. https://cyberbotics.com/doc/guide/elisa3, accessed 2023.

[23] Nair Maria Maia de Abreu. Old and new results on algebraic connectivity of graphs. *Linear Algebra and its Applications*, 423(1):53–73, 2007. Special Issue devoted to papers presented at the Aveiro Workshop on Graph Spectra.

[24] Claudio De Persis and Romain Postoyan. A lyapunov redesign of coordination algorithms for cyber-physical systems. *IEEE Transactions on Automatic Control*, 62(2):808–823, 2017.

[25] J.P. Desai, J.P. Ostrowski, and V. Kumar. Modeling and control of formations of nonholonomic mobile robots. *IEEE Transactions on Robotics and Automation*, 17(6):905–908, 2001.

[26] Dimos Dimarogonas and Kostas Kyriakopoulos. A connection between formation infeasibility and velocity alignment in kinematic multi-agent systems. *Automatica*, 44:2648–2654, 10 2008.

[27] Dimos V. Dimarogonas and Emilio Frazzoli. Distributed event-triggered control strategies for multi-agent systems. In *2009 47th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 906–910, 2009.

[28] Dimos V. Dimarogonas, Emilio Frazzoli, and Karl H. Johansson. Distributed event-triggered control for multi-agent systems. *IEEE Transactions on Automatic Control*, 57(5):1291–1297, 2012.

[29] Dimos V. Dimarogonas and Karl H. Johansson. Event-triggered control for multi-agent systems. In *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*, pages 7131–7136, 2009.

[30] Dimos V. Dimarogonas and Kostas J. Kyriakopoulos. On the rendezvous problem for multiple nonholonomic agents. *IEEE Transactions on Automatic Control*, 52(5):916–922, 2007.

[31] Zhengtao Ding. Consensus output regulation of a class of heterogeneous nonlinear systems. *IEEE Transactions on Automatic Control*, 58(10):2648–2653, 2013.

[32] S.J. Dodds. Adaptive, high precision, satellite attitude control for microprocessor implementation. *Automatica*, 17(4):563–573, 1981.

[33] Ali Dorri, Salil S. Kanhere, and Raja Jurdak. Multi-agent systems: A survey. *IEEE Access*, 6:28573–28593, 2018.

[34] Gaopeng Duan, Feng Xiao, and Long Wang. Asynchronous periodic edge-event triggered control for double-integrator networks with communication time delays. *IEEE Transactions on Cybernetics*, 48(2):675–688, 2018.

[35] Shiming Duan, Jun Ni, and A. Ulsoy. Decay function estimation for linear time delay systems via the lambert w function. *Journal of Vibration and Control*, 18:1462–1473, 09 2012.

[36] Yuan Fan, Gang Feng, Yong Wang, and Cheng Song. Distributed event-triggered control of multi-agent systems with combinational measurements. *Automatica*, 49(2):671–675, 2013.

[37] Yuan Fan, Lu Liu, Gang Feng, and Yong Wang. Self-triggered consensus for multi-agent systems with zeno-free triggers. *IEEE Transactions on Automatic Control*, 60(10):2779–2784, 2015.

[38] J. Alexander Fax and Richard M. Murray. Graph laplacians and stabilization of vehicle formations. *IFAC Proceedings Volumes*, 35(1):55–60, 2002. 15th IFAC World Congress.

[39] J.A. Fax and R.M. Murray. Information flow and cooperative control of vehicle formations. *IEEE Transactions on Automatic Control*, 49(9):1465–1476, 2004.

[40] Xin-Lei Feng. Fast convergent consensus of high-order continuous-time multi-agent systems. In *2021 17th International Conference on Computational Intelligence and Security (CIS)*, pages 222–226, 2021.

[41] Miroslav Fiedler. Algebraic connectivity of graphs. *Czechoslovak Mathematical Journal*, 23:298–305, 01 1973.

[42] Eloy Garcia, Yongcan Cao, and David W. Casbeer. Periodic event-triggered synchronization of linear multi-agent systems with communication delays. *IEEE Transactions on Automatic Control*, 62(1):366–371, 2017.

[43] Eloy Garcia, Yongcan Cao, Han Yu, Panos Antsaklis, and David Casbeer. Decentralised event-triggered cooperative control with limited communication. *International Journal of Control*, 86(9):1479–1488, 2013.

[44] GCTronic. Elisa-3 robot. [Online]. Available: https://www.gctronic.com/doc/index.php/Elisa-3#Multiplatform_monitor, accessed 2023.

[45] GCTronic. Elisa3 Node C++. https://github.com/gctronic/elisa3_node_cpp.git, accessed 2023.

[46] J. S. Gibson. Linear-quadratic optimal control of hereditary differential systems: Infinite dimensional riccati equations and numerical approximations. *SIAM Journal on Control and Optimization*, 21(1):95–139, 1983.

[47] Antoine Girard. Dynamic triggering mechanisms for event-triggered control. *IEEE Transactions on Automatic Control*, 60(7):1992–1997, 2015.

[48] Robert Grone and Russell Merris. The laplacian spectrum of a graph ii. *SIAM Journal on Discrete Mathematics*, 7(2):221–229, 1994.

[49] Zhi-Hong Guan, Feng-Lan Sun, Yan-Wu Wang, and Tao Li. Finite-time consensus for leader-following second-order multi-agent networks. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 59(11):2646–2654, 2012.

[50] Aric Hagberg, Pieter Swart, and Daniel S Chult. Networkx. https://github.com/networkx/networkx, 2008. Accessed on May 3, 2023.

[51] Qingwei Han, Chuanhai Yang, Zhenxing Li, and Zhaodong Liu. Consensus of strict feedback multi-agent systems via event-triggered approach. In *2021 36th Youth Academic Annual Conference of Chinese Association of Automation (YAC)*, pages 761–764, 2021.

[52] Y. Hatano and M. Mesbahi. Agreement over random networks. *IEEE Transactions on Automatic Control*, 50(11):1867–1872, 2005.

[53] Pedram Hovareshti, John S. Baras, and Vijay Gupta. Average consensus over small world networks: A probabilistic framework. In *2008 47th IEEE Conference on Decision and Control*, pages 375–380, 2008.

[54] Wenfeng Hu, Lu Liu, and Gang Feng. Consensus of linear multi-agent systems by distributed event-triggered strategy. *IEEE Transactions on Cybernetics*, 46(1):148–157, 2016.

[55] Bomin Jiang, Mohammad Deghat, and Brian D. O. Anderson. Simultaneous velocity and position estimation via distance-only measurements with application to multi-agent system control. *IEEE Transactions on Automatic Control*, 62(2):869–875, 2017.

[56] Zhong-Ping Jiang, Iven M.Y. Mareels, and Yuan Wang. A lyapunov formulation of the nonlinear small-gain theorem for interconnected iss systems. *Automatica*, 32(8):1211–1215, 1996.

[57] E. Jones, T. Oliphant, and P. Peterson. Scipy: Open source scientific tools for python. https://www.scipy.org/, 2001. accessed 2023.

[58] Evgeny Kharisov, Xiaofeng Wang, and Naira Hovakimyan. *Distributed Event-Triggered Consensus Algorithm for uncertain Multi-Agent Systems*, pages 1–15. 2010.

[59] Yoonsoo Kim and M. Mesbahi. On maximizing the second smallest eigenvalue of a state-dependent graph laplacian. *IEEE Transactions on Automatic Control*, 51(1):116–120, 2006.

[60] Changbin Li, Yi He, and Aiguo Wu. Synchronization for a class of complex dynamical networks. In *Proceedings of the 10th World Congress on Intelligent Control and Automation*, pages 1010–1013, 2012.

[61] Xianwei Li, Yang Tang, and Hamid Reza Karimi. Consensus of multi-agent systems via fully distributed event-triggered control. *Automatica*, 116:108898, 2020.

[62] Yiting LI. Indoor localization with multi-rate extended kalman filter. Master's thesis, Delft University of Technology, 2023.

[63] Zhiyun Lin, Bruce Francis, and Manfredi Maggiore. State agreement for continuous-time coupled nonlinear systems. *SIAM J. Control and Optimization*, 46:288–307, 01 2007.

[64] Pfeiffer S. Liu, C. and G.C.H.E. de Croon. Cooperative relative localization in mav swarms with ultra-wideband ranging. 2022.

[65] Xiaoyang Liu, James Lam, Wenwu Yu, and Guanrong Chen. Finite-time consensus of multiagent systems with a switching protocol. *IEEE Transactions on Neural Networks and Learning Systems*, 27(4):853–862, 2016.

[66] Xiwei Liu, Wenlian Lu, and Tianping Chen. Consensus of multi-agent systems with unbounded time-varying delays. *IEEE Transactions on Automatic Control*, 55(10):2396–2401, 2010.

[67] Yaohua Liu, Cameron Nowzari, Zhi Tian, and Qing Ling. Asynchronous periodic event-triggered coordination of multi-agent systems. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 6696–6701, 2017.

[68] Zhongxin Liu, Zengqiang Chen, and Zhuzhi Yuan. Event-triggered average-consensus of multi-agent systems with weighted and direct topology. *Journal of Systems Science and Complexity*, 25, 10 2012.

[69] E. Lovisari and S. Zampieri. Performance metrics in the consensus problem: a survey *. *IFAC Proceedings Volumes*, 43(21):324–335, 2010. 4th IFAC Symposium on System Structure and Control.

[70] Yang T. Lui Y., Zhang T. and Tian Q. Research on a new algorithm for the robot tracking problem. *Machinery, Materials Science and Engineering Applications*, 37:25–34, 2017.

[71] Nancy A. Lynch. *Distributed Algorithms*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1996.

[72] Manuel Mazo, Adolfo Anta, and Paulo Tabuada. On self-triggered control for linear systems: Guarantees and complexity. In *2009 European Control Conference (ECC)*, pages 3767–3772, 2009.

[73] Deyuan Meng and Yingmin Jia. Robust consensus algorithms for multiscale coordination control of multivehicle systems with disturbances. *IEEE Transactions on Industrial Electronics*, 63(2):1107–1119, 2016.

[74] Xiangyu Meng, Lihua Xie, Yeng Chai Soh, Cameron Nowzari, and George J. Pappas. Periodic event-triggered average consensus over directed graphs. In *2015 54th IEEE Conference on Decision and Control (CDC)*, pages 4151–4156, 2015.

[75] Russell Merris. A note on laplacian graph eigenvalues. *Linear Algebra and its Applications*, 285(1):33–35, 1998.

[76] M. Mesbahi and F.Y. Hadaegh. Formation flying control of multiple spacecraft via graphs, matrix inequalities, and switching. In *Proceedings of the 1999 IEEE International Conference on Control Applications (Cat. No.99CH36328)*, volume 2, pages 1211–1216 vol. 2, 1999.

[77] O. Michel. Cyberbotics ltd. webots™: professional mobile robot simulation. In *International Journal of Advanced Robotic Systems*, volume 1, 2004.

[78] Rajiv Kumar Mishra and Hideaki Ishii. Average consensus in n-time multi-agent systems with distributed event-triggered control. In *2021 European Control Conference (ECC)*, pages 608–613, 2021.

[79] Hossein Moradian and Solmaz S. Kia. A study on rate of convergence increase due to time delay for a class of linear systems. In *2018 IEEE Conference on Decision and Control (CDC)*, pages 5433–5438, 2018.

[80] Hossein Moradian and Solmaz S. Kia. a study on accelerating average consensus algorithms using delayed feedback. *IEEE Transactions on Control of Network Systems*, pages 1–11, 2022.

[81] L. Moreau. Stability of multiagent systems with time-dependent communication links. *IEEE Transactions on Automatic Control*, 50(2):169–182, 2005.

[82] Ulrich Münz, Antonis Papachristodoulou, and Frank Allgöwer. Delay robustness in consensus problems. *Automatica*, 46(8):1252–1265, 2010.

[83] S. Mohammad Noorbakhsh and Jafar Ghaisari. Distributed event-triggered average consensus protocol for multi-agent systems. In *2015 23rd Iranian Conference on Electrical Engineering*, pages 840–845, 2015.

[84] Cameron Nowzari and Jorge Cortés. Self-triggered coordination of robotic networks for optimal deployment. In *Proceedings of the 2011 American Control Conference*, pages 1039–1044, 2011.

[85] Cameron Nowzari and Jorge Cortés. Distributed event-triggered coordination for average consensus on weight-balanced digraphs. *Automatica*, 68:237–244, 2016.

[86] Cameron Nowzari, Eloy Garcia, and Jorge Cortés. Event-triggered communication and control of networked systems for multi-agent consensus. *Automatica*, 105:1–27, 2019.

[87] R. Olfati-Saber. Ultrafast consensus in small-world networks. In *Proceedings of the 2005, American Control Conference, 2005.*, pages 2371–2378 vol. 4, 2005.

[88] R. Olfati-Saber. Flocking for multi-agent dynamic systems: algorithms and theory. *IEEE Transactions on Automatic Control*, 51(3):401–420, 2006.

[89] R. Olfati-Saber and R.M. Murray. Consensus problems in networks of agents with switching topology and time-delays. *IEEE Transactions on Automatic Control*, 49(9):1520–1533, 2004.

[90] Reza Olfati-Saber, J. Alex Fax, and Richard M. Murray. Consensus and cooperation in networked multi-agent systems. *Proceedings of the IEEE*, 95(1):215–233, 2007.

[91] J.A. Oroko and G.N. Nyakoe. Obstacle avoidance and path planning schemes for autonomous navigation of a mobile robot: A review. In *Sustainable Research and Innovation Proceedings 2012*, 2012.

[92] K.J. Åström and B. Bernhardsson. Comparison of periodic and event based sampling for first-order stochastic systems. *IFAC Proceedings Volumes*, 32(2):5006–5011, 1999. 14th IFAC World Congress 1999, Beijing, China.

[93] Wei Ren. On consensus algorithms for double-integrator dynamics. *IEEE Transactions on Automatic Control*, 53(6):1503–1509, 2008.

[94] Wei Ren and Ella Atkins. Distributed multi-vehicle coordinated control via local information exchange. *International Journal of Robust and Nonlinear Control*, 17, 07 2007.

[95] Wei Ren and R.W. Beard. Consensus seeking in multi-agent systems under dynamically changing interaction topologies. *IEEE Transactions on Automatic Control*, 50(5):655–661, 2005.

[96] Wei Ren and R.W. Beard. Consensus seeking in multiagent systems under dynamically changing interaction topologies. *IEEE Transactions on Automatic Control*, 50(5):655–661, 2005.

[97] Alain Sarlette, Rodolphe Sepulchre, and Naomi Ehrich Leonard. Autonomous rigid body attitude synchronization. In *2007 46th IEEE Conference on Decision and Control*, pages 2566–2571, 2007.

[98] Luca Scardovi and Rodolphe Sepulchre. Synchronization in networks of identical linear systems. *Automatica*, 45:2557–2562, 07 2009.

[99] Jin Seo, Hyungbo Shim, and Juhoon Back. Consensus of high-order linear systems using dynamic output feedback compensator: Low gain approach. *Automatica*, 45:2659–2664, 11 2009.

[100] Georg S. Seyboth, Dimos V. Dimarogonas, and Karl H. Johansson. Event-based broadcasting for multi-agent average consensus. *Automatica*, 49(1):245–252, 2013.

[101] Jiahui Shi and Wenfeng Hu. Consensus of second-order multi-agent systems by event-triggered control. In *2018 13th World Congress on Intelligent Control and Automation (WCICA)*, pages 269–273, 2018.

[102] Jiantao Shi. Event-triggered based consensus control for a type of multi-agent systems. In *2020 IEEE 9th Data Driven Control and Learning Systems Conference (DDCLS)*, pages 870–873, 2020.

[103] Youfeng Su and Jie Huang. Stability of a class of linear switching systems with applications to two consensus problems. *IEEE Transactions on Automatic Control*, 57(6):1420–1430, 2012.

[104] Yuan Gong Sun, Long Wang, and Guangming Xie. Average consensus in networks of dynamic agents with switching topologies and multiple time-varying delays. *Systems Control Letters*, 57(2):175–183, 2008.

[105] Paulo Tabuada. Event-triggered real-time scheduling of stabilizing control tasks. *IEEE Transactions on Automatic Control*, 52(9):1680–1685, 2007.

[106] Alireza Tahbaz-Salehi and Ali Jadbabaie. A necessary and sufficient condition for consensus over random networks. *IEEE Transactions on Automatic Control*, 53(3):791–795, 2008.

[107] Gerard Tel. *Introduction to Distributed Algorithms.* Cambridge University Press, USA, 2nd edition, 2001.

[108] Yu-Ping Tian and Cheng-Lin Liu. Consensus of multi-agent systems with diverse input and communication delays. *IEEE Transactions on Automatic Control*, 53(9):2122–2128, 2008.

[109] J. Tsitsiklis, D. Bertsekas, and M. Athans. Distributed asynchronous deterministic and stochastic gradient optimization algorithms. *IEEE Transactions on Automatic Control*, 31(9):803–812, 1986.

[110] John N. Tsitsiklis. *Problems in decentralized decision making and computation.* PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, 1984. ndltd.org (oai:dspace.mit.edu:1721.1/15254).

[111] S. Emre Tuna. Conditions for synchronizability in arrays of coupled linear systems. *IEEE Transactions on Automatic Control*, 54(10):2416–2420, 2009.

[112] Aiping Wang. Event-based consensus control for single-integrator networks with communication time delays. *Neurocomputing*, 173:1715–1719, 2016.

[113] Long Wang and Feng Xiao. Finite-time consensus problems for networks of dynamic agents. *IEEE Transactions on Automatic Control*, 55(4):950–955, 2010.

[114] Xiaofeng Wang and M.D. Lemmon. Self-triggered feedback control systems with finite-gain l(2) stability. *Automatic Control, IEEE Transactions on*, 54:452 – 467, 04 2009.

[115] M. Wooldridge. *An Introduction to Multiagent Systems*. John Wiley & Sons, Chichester, UK, 2nd edition, 2009.

[116] Chai Wah Wu. Synchronization and convergence of linear dynamics in random directed networks. *IEEE Transactions on Automatic Control*, 51(7):1207–1210, 2006.

[117] Feng Xiao and Long Wang. Asynchronous consensus in continuous-time multi-agent systems with switching topology and time-varying delays. *IEEE Transactions on Automatic Control*, 53(8):1804–1816, 2008.

[118] Feng Xiao, Long Wang, Jie Chen, and Yanping Gao. Finite-time formation control for multi-agent systems. *Automatica*, 45(11):2605–2611, 2009.

[119] Lin Xiao and S. Boyd. Fast linear iterations for distributed averaging. In *42nd IEEE International Conference on Decision and Control (IEEE Cat. No.03CH37475)*, volume 5, pages 4997–5002 Vol.5, 2003.

[120] Guangming Xie, Huiyang Liu, Long Wang, and Yingmin Jia. Consensus in networked multi-agent systems via sampled control: Switching topology case. In *2009 American Control Conference*, pages 4525–4530, 2009.

[121] Bo Yang and Huajing Fang. Forced consensus in networks of double integrator systems with delayed input. *Automatica*, 46(3):629–632, 2010.

[122] Dapeng Yang, Wei Ren, Xiangdong Liu, and Weisheng Chen. Decentralized event-triggered consensus for linear multi-agent systems under general directed graphs. *Automatica*, 69:242–249, 2016.

[123] Tao Yang, Sandip Roy, Yan Wan, and Ali Saberi. Constructing consensus controllers for networks with identical general linear agents. *International Journal of Robust and Nonlinear Control*, 21:1237 – 1256, 07 2011.

[124] Wenwu Yu, Jinde Cao, Guanrong Chen, Jinhu Lu, Jian Han, and Wei Wei. Local synchronization of a complex network model. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 39(1):230–241, 2009.

[125] Wenwu Yu, Guanrong Chen, and Ming Cao. Consensus in directed networks of agents with nonlinear dynamics. *IEEE Transactions on Automatic Control*, 56(6):1436–1441, 2011.

[126] Wenwu Yu, Guanrong Chen, Ming Cao, and Jürgen Kurths. Second-order consensus for multiagent systems with directed topologies and nonlinear dynamics. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 40(3):881–891, 2010.

[127] Huaipin Zhang, Dong Yue, Xiuxia Yin, Songlin Hu, and Chun xia Dou. Finite-time distributed event-triggered consensus control for multi-agent systems. *Information Sciences*, 339:132–142, 2016.

[128] Xiao-Dong Zhang and Rong Luo. The spectral radius of triangle-free graphs. *Australasian Journal of Combinatorics*, 26, 01 2002.

[129] Ya Zhang and Yu-Ping Tian. Consentability and protocol design of multi-agent systems with stochastic switching topology. *Automatica*, 45(5):1195–1201, 2009.

[130] Bin Zhou and Zongli Lin. Consensus of high-order multi-agent systems with input and communication delays-state feedback case. In *2013 American Control Conference*, pages 4027–4032, 2013.

[131] Feng Zhou, Zhiwu Huang, Yingze Yang, Jing Wang, Liran Li, and Jun Peng. Decentralized event-triggered cooperative control for multi-agent systems with uncertain dynamics using local estimators. *Neurocomputing*, 237:388–396, 2017.

[132] Quan Min Zhu, Jianan Wang, Chunyan Wang, Ming Xin, Zhengtao Ding, and Jiayuan Shan, editors. *Cooperative Control of Multi-Agent Systems*. Emerging Methodologies and Applications in Modelling. Academic Press, 2020.

[133] Karl-Erik Åarzén. A simple event-based pid controller. *IFAC Proceedings Volumes*, 32(2):8687–8692, 1999. 14th IFAC World Congress 1999, Beijing, China, 5-9 July.

Glossary

List of Acronyms

MAS	multi-agent systems
ETC	event-triggered consensus
PETC	periodic event-triggered consensus
ITD	input time delay
CTD	communication time delay
IoT	Internet of Things
CPS	cyber-physical systems
MIET	minimum inter-event time
FTC	finite-time consensus
ISpS	input-to-state practically stable
ISS	input-to-state stable
FSM	Finite State Machine
MSB	most significant bit
MR-EKF	Multi-Rate Extended Kalman Filter

List of Symbols

ϵ	Error radius
ω	Convergence rate of system (3-8) with $e(t) = 0$
τ	Communication delay
$\delta(t)$	Disagreement vector
$\hat{x}_i(t)$	Last broadcasted state of agent i
λ_N	Largest eigenvalue of the Laplacian

$\lambda_2(L)$	Algebraic connectivity of graph L
$\lambda_{max}(M)$	Maximum eigenvalue of matrix M
$\lambda_{min}(M)$	Minimum eigenvalue of matrix M
$\|C\|$	Number of triggering events
$\|E\|$	Number of edges
$\mathbb{N}$	Set of natural numbers
$\mathbb{R}$	Set of real numbers
$\mathbb{Z}$	Set of integer numbers
$\mathcal{E}$	Set of edges
$\mathcal{K}$	Kappa class function
$\mathcal{KL}$	Kappa-ell class function
$\mathcal{N}$	Set of all agents in the network
$\mathcal{N}_i$	Set of neighbours of a node i
$\mathcal{V}$	Set of vertices
$\overline{x}_0$	Average of the initial position x_0
D	Graph diameter
d_i	Degree of node i
$e(t)$	Error vector
h	Sampling period
L	Laplacian
L_2	The space of square Lebesgue integrable functions
M	Overshoot from the initial disagreement, $\max_{i\in\mathcal{N}}\|\delta_i(0)\|$
N	Number of agents in the network
t_{ETC}	Convergence time of the ETC algorithm
t_{PETC}	Convergence time of the PETC algorithm
u^{max}	Maximum control input
$u_i(t)$	Control input of agent i
$x_i(t)$	State of agent i
Z	Hilbert space
N	Number of nodes in a graph
$_{max}$	Maximum
$_{min}$	Minimum